Monte Carlo Integration in Bayesian Estimation

Avinash Kak

Purdue University

April 29, 2024

2:20pm

An RVL Tutorial Presentation

Originally presented in Spring 2009
Updated most recently in April 2024

CONTENTS

Back to TOC

1: Introduction

- The goal of this tutorial presentation is to focus on the Monte-Carlo methods for solving what are known as probabilistic integrals in Bayesian estimation. The integrand in a probabilistic integral consists of two parts, one a probability distribution and the rest. As a result, the value of the integral can be considered to be the expectation of the "rest" with respect to the probability distribution.

- The Monte-Carlo approach to solving a probabilistic integral consists of sampling the parameter space according to the probability distribution part of integrand and then merely summing up the values of the rest of the integrand at those points in the parameter space.

- When the probability distribution part of the integrand is simple — such as uniform or Gaussian — the method described above will generally work well. When that is not the case, you may have to resort to what's known as Importance Sampling of the parameter space.

- The first four sections of this tutorial are devoted to explaining in detail the ideas mentioned above. The last section, Section 5,

takes up the subject of time varying systems and their probabilistic modeling. As an example of a time-varying system, think of tracking objects and humans in videos. What you have here is time-sequenced data in the form of image frames and you want to track an object by estimating its position with respect to all its 6 degrees of freedom, 3 translational and 3 rotational. What you'd like to do is to create a data model for tracking a vector of 6 numbers in the presence of background noise.

- As I show in Section 5 of this tutorial, when the time evolution of whatever it is you are tracking is nonlinear and the observation noise non-Gaussian, you may again have to resort to Monte-Carlo based solutions for filtering out the noise and for making predictions.

- I should also mention that this tutorial is a continuation of my tutorial:

  "*ML, MAP, and Bayesian — The Holy Trinity of Parameter Estimation and Data Prediction*"

  that can be downloaded from:

  `https://engineering.purdue.edu/kak/Tutorials/Trinity.pdf`

Back to TOC

2: A Brief Review of ML, MAP, and Bayesian Estimation

- The main goal of this section is to refresh your memory about the basic vocabulary of *maximum likelihood* (ML), *maximum a-posteriori* (MAP), and *Bayesian* estimation. The subsections that follow are drawn from my "Holy Trinity" tutorial that I cited at the end of the previous section.

- Perhaps the most fundamental terms in this vocab are: *likelihood*, *log-likelihood*, *evidence*, *prior*, and *posterior*. To understand these terms, you must also be able to wrap your head around the terms and phrases like *observation model*, model parameters, *probability distribution over the model parameters*, *independent observations*, etc.

- As you will see in this section, constructing ML and MAP estimates is relatively easy. These estimates would be for the parameters of the model that you have conjured up for the observed data.

- On the other hand, Bayesian estimation is made difficult by the fact that it requires us to estimate not just a single numerical value for each model parameters, but a probability distribution

over all possible values for the model parameters.

- The challenges faced in constructing Bayesian estimates for the probability distribution for the model parameters provide the motivation for the Monte Carlo techniques presented in Section 3 of this tutorial.

Back to TOC

2.1: Estimation of Parameters and Prediction of Future Values from Evidence

- Let's say we have evidence $\mathcal{X}$ that consists of a set of independent observations:

$$\mathcal{X} \quad = \quad \{\mathbf{x}_i\}_{i=1}^{|\mathcal{X}|} \tag{1}$$

  where each $\mathbf{x}_i$ is a realization of a random variable $\mathbf{x}$. **Each observation $\mathbf{x}_i$ is, in general, a data point in a multidimensional space.**

- Let's say that a set Θ of probability distribution parameters can be used to explain the evidence $\mathcal{X}$.

- The manner in which the evidence $\mathcal{X}$ depends on the parameters Θ will be referred to as the **observation model**.

- **What can we do with this evidence?**

- We may wish to estimate the parameters Θ with the help of the Bayes' Rule

$$prob(\Theta|\mathcal{X}) \quad = \quad \frac{prob(\mathcal{X}|\Theta) \;\cdot\; prob(\Theta)}{prob(\mathcal{X})} \tag{2}$$

- Or, given a new observation $\tilde{\mathbf{x}}$, we may wish to compute the probability that the observation is supported by the evidence

$$prob(\tilde{\mathbf{x}}|\mathcal{X}) \tag{3}$$

- The former represents **parameter estimation** and the latter **data prediction** or **regression**.

- **In the rest of this section, I'll first focus on the estimation of the parameters Θ.** To that end, We can use the Bayes' Rule

$$prob(\Theta|\mathcal{X}) = \frac{prob(\mathcal{X}|\Theta) \cdot prob(\Theta)}{prob(\mathcal{X})} \tag{4}$$

  and interpret the rule as

$$posterior = \frac{likelihood \cdot prior}{evidence} \tag{5}$$

- Since we will be using these terms frequently in the rest of this tutorial, remember that *posterior* means $prob(\Theta|\mathcal{X})$, *likelihood* means $prob(\mathcal{X}|\Theta)$, and *evidence* means $prob(\mathcal{X})$.

- The likelihood distribution $prob(\mathcal{X}|\Theta)$ is the same thing as the **observation model** we talked about earlier.

- **In what follows, I'll now consider the Maximum Likelihood (ML) Estimation of Θ.**

- We seek that value for Θ which maximizes the likelihood **for the evidence actually recorded**. That is, we seek that value for Θ which gives largest value to

$$prob(\mathcal{X}|\Theta) \tag{6}$$

for the measured $\mathcal{X}$. Recognizing that the evidence $\mathcal{X}$ consists of independent observations $\{\mathbf{x}_1, \mathbf{x}_2, ...\}$, we can say that we seek that value Θ which maximizes

$$\prod_{\mathbf{x}_i \in \mathcal{X}} prob(\mathbf{x}_i|\Theta) \tag{7}$$

- Because of the product on the RHS, it is often simpler to use the logarithm of the product. What you get from the logarithm is commonly referred to as the log-likelihood. Using the symbol $\mathcal{L}$ to denote the log-likelihood, we can express it as:

$$\mathcal{L} \quad = \quad \sum_{\mathbf{x}_i \in \mathcal{X}} log\ prob(\mathbf{x}_i|\Theta) \tag{8}$$

and say that we seek that solution for the parameters that maximizes $\mathcal{L}$. That is,

$$\widehat{\Theta}_{ML} \quad = \quad \underset{\Theta}{\text{argmax}}\ \mathcal{L} \tag{9}$$

- The ML solution for the parameters is usually obtained by setting

$$\frac{\partial \mathcal{L}}{\partial \theta_i} \quad = \quad 0 \qquad \forall\ \theta_i \in \Theta \tag{10}$$

- **Special Note on ML versus MAP and Bayesian:**

> ML estimation is the easiest to implement because one usually expresses the **observation model** through equations that make explicit the analytical form of $p(\mathcal{X}|\Theta)$. This analytical form can then be manipulated to yield Θ that maximizes $p(\mathcal{X}|\Theta)$. See the Holy Trinity tutorial mentioned in the Introduction for an example.

- **I'll now consider Maximum a Posteriori (MAP) Estimation of Θ.**

- For constructing a maximum *a posteriori* (MAP) estimate, we first go back to our Bayes' Rule in Eq. (4):

$$prob(\Theta|\mathcal{X}) \quad = \quad \frac{prob(\mathcal{X}|\Theta) \;\cdot\; prob(\Theta)}{prob(\mathcal{X})} \tag{11}$$

- We now seek that value for Θ which maximizes the posterior $prob(\Theta|\mathcal{X})$.

- Therefore, our solution can now be stated as

$$\begin{aligned} \widehat{\Theta}_{MAP} \quad &= \quad \underset{\Theta}{\text{argmax}} \; prob(\Theta|\mathcal{X}) \\ &= \quad \underset{\Theta}{\text{argmax}} \; \frac{prob(\mathcal{X}|\Theta) \;\cdot\; prob(\Theta)}{prob(\mathcal{X})} \end{aligned} \tag{12}$$

- Being independent of Θ, the denominator can be ignored for finding the value of Θ where the numerator becomes maximum.

- So we can write for the solution:

$$\begin{aligned} \widehat{\Theta}_{MAP} &= \underset{\Theta}{\text{argmax}}\ prob(\mathcal{X}|\Theta)\ \cdot\ prob(\Theta) \\ &= \underset{\Theta}{\text{argmax}}\ \prod_{\mathbf{x}_i \in \mathcal{X}} prob(\mathbf{x}_i|\Theta)\ \cdot\ prob(\Theta) \end{aligned} \tag{13}$$

- As with the ML estimate, we can make this problem easier if we first take the logarithm of the posteriors. We can then write

$$\widehat{\Theta}_{MAP} = \underset{\Theta}{\text{argmax}} \left(\sum_{\mathbf{x}_i \in \mathcal{X}} log\ prob(\mathbf{x}_i|\Theta)\ +\ log\ prob(\Theta) \right) \tag{14}$$

- **Finding Θ where the above expression is maximized is just as easy as the maximization needed for ML if an analytical expression for the prior is available. Often people use parametric functions such as the beta, Dirichlet, etc., for $prob(\Theta)$.**

- **Finally, let's consider Bayesian Estimation.**

- Before taking up the case of Bayesian Estimation, note that, given the evidence $\mathcal{X}$, ML considers the parameter vector Θ to be a constant and seeks out that value for the constant that provides maximum support for the evidence. ML does NOT allow us to inject our prior beliefs about the likely values for Θ in the estimation calculations.

- MAP allows for the fact that the parameter vector Θ can take values from a distribution that expresses our prior beliefs regarding the parameters. MAP returns that value for Θ where the probability $prob(\Theta|\mathcal{X})$ is a maximum.

- **Both ML and MAP return only single and specific values for the parameter Θ.**

- **Bayesian estimation, by contrast, calculates fully the posterior distribution $prob(\Theta|\mathcal{X})$.**

- Of all the Θ values made possible by this distribution, it is our job to select a value that we consider best in some sense. For example, we may choose the expected value of Θ assuming its variance is small enough.

- The variance that we can calculate for the parameter Θ from its posterior distribution allows us to express our confidence in any specific value we may use as an estimate. If the variance is too large, we may declare that there does not exist a good estimate for Θ.

Back to TOC

2.2: What Makes Bayesian Estimation Difficult: A Theoretical Perspective

- Bayesian estimation is made complex by a variety of reasons, some theoretical and some practical. What's interesting is that some of the theoretical reasons that make Bayesian estimation difficult disappear in a practical implementation of the approach, but then other difficulties crop up when actually implementing the approach.

- Let's first focus on why a theoretician might consider Bayesian estimation difficult. Note that vis-a-vis MAP estimation, now the denominator in the Bayes' Rule

$$prob(\Theta|\mathcal{X}) \quad = \quad \frac{prob(\mathcal{X}|\Theta) \;\cdot\; prob(\Theta)}{prob(\mathcal{X})} \tag{15}$$

  cannot be ignored.

- The denominator in the above equation is known as the **probability of evidence** and is related to the other probabilities that make their appearance in the Bayes' Rule by

$$prob(\mathcal{X}) \;=\; \int_{\Theta} prob(\mathcal{X}|\Theta) \cdot prob(\Theta)\; d\Theta \tag{16}$$

- Bayesian estimation therefore calls on us to compute the following **posterior *as a distribution***:

$$prob(\Theta|\mathcal{X}) = \frac{prob(\mathcal{X}|\Theta) \cdot prob(\Theta)}{\int prob(\mathcal{X}|\Theta) \cdot prob(\Theta)\ d\Theta} \tag{17}$$

- If you want to be able to derive an algebraic form for the posterior, the most challenging part of Bayesian estimation is the integration in the denominator.

- This leads to the following thought critical to Bayesian estimation: **For a given observation model, if we have a choice regarding how we express our prior beliefs, we must use that form which allows us to carry out the integration in the denominator. It is this thought that leads to the notion of conjugate priors.**

- For a given algebraic form for the likelihood (**that is, for a given observation model, which is the same thing as the likelihood function**), the different forms for the prior $prob(\Theta)$ pose different levels of difficulty for the determination of the marginal in the denominator and, therefore, for the determination of the posterior.

- For a given algebraic form for the likelihood function $prob(\mathcal{X}|\Theta)$, a prior $prob(\Theta)$ is called a **conjugate prior** if the posterior $prob(\Theta|\mathcal{X})$ has the same algebraic form as the prior.

- Bayesian estimation and prediction become much easier should the engineering assumptions allow a conjugate prior to be chosen for the **applicable observation model**.

- When the **observation model** (meaning the likelihood distribution) can be assumed to be Gaussian, a Gaussian prior would constitute a conjugate prior because in this case the posterior would also be Gaussian.

- When data is generated by an experiment based on Bernoulli trials, the likelihood function is a binomial and the beta distribution constitutes a conjugate prior.

- When the likelihood function can be modeled as a multinomial, the conjugate prior is the Dirichlet distribution. (See my Holy Trinity tutorial mentioned in the Introduction section.)

Back to TOC

2.3: What Makes Bayesian Estimation Difficult: A Practical Perspective

- In the comment after Eq. (17), I mentioned that, from a theoretical perspective, the main difficulty with Bayesian estimation is the integration in the denominator on the right in Eqs. (15) and (17).

- **What is interesting is that, from an implementation perspective, the calculation of the denominator in Eqs. (15) and (17) may turn out to be the least of your problems.**

- **That is because the role played by the denominator is merely as a normalizer for the numerator.** So if you compute the Bayesian posterior distribution ignoring the denominator at a reasonably large number of sampling points in the parameter space spanned by Θ, just by summing the resulting estimates for the posterior distribution values you can find the normalization constant since the sum must add up to 1.

- **If one were to analyze the above mentioned simplification theoretically, you would see that it makes sense only when we can assume the prior distribution** $prob(\Theta)$ **to be uniform.**

- Even when it is possible to take advantage of the above mentioned simplification, a practical implementation of Bayesian estimation can throw up additional challenges that can be attributed to the **observation model** (meaning, the likelihood distribution defined in the previous section) that you might want to use in a given application.

- **From a practical perspective, there is also the issue of non-recursive versus recursive estimation.**

- Several of these challenges can be addressed by **importance sampling** and **Monte Carlo integration**, as you will see in the rest of this presentation.

- In the rest of this tutorial, I'll first take up the subject of the Monte Carlo technique for solving integrals that involve probability distributions. In that discussion, I'll also present importance sampling as a way to improve the performance of Monte Carlo based integration.

Back to TOC

3: Monte-Carlo Integration for Bayesian Estimation

- As mentioned in the previous section, Bayesian estimation, in general, requires us to solve the integral in the denominator of Eq. (17).

- The form shown for the denominator in Eq. (17) is **special** — in the sense that one of the terms in the denominator is a probability distribution.

- When an integrand can be expressed as a product of two functions, with one of the two being a probability distribution (as is the case with the denominator in Eq. (17)), you have what's known as a **probabilistic integral**.

- A probabilistic integral can be interpreted as finding the expectation of the values of the other function in the product of the two functions mentioned above. This is best done with the Monte-Carlo technique mentioned in the next section.

Back to TOC

3.1: Solving Probabilistic Integrals Numerically

- Integrals that involve probability distributions in the integrands are ideal for solution by Monte Carlo methods.

- Consider the following general example of such an integration:

$$E(g(\mathcal{X}, \Theta)) \;\; = \;\; \int g(\mathcal{X}, \Theta) \cdot prob(\Theta)\, d\Theta \tag{18}$$

- When $g(\mathcal{X}, \Theta) \equiv \Theta$, the integration amounts to finding the mean of the elements of the random vector Θ. In general, when $g(\mathcal{X}, \Theta) \equiv \Theta^n$, the integration amounts to finding the n^{th} moment of the random vector Θ.

- The Monte Carlo approach to solving the integration shown above is to draw samples from the probability distribution $prob(\Theta)$ and then to estimate the integral with the help of these samples.

- When $prob(\Theta)$ is simple, such as uniform or normal, it is trivial to draw such samples from the distribution $prob(\Theta)$ by making straightforward function calls to standard software libraries for generating random numbers.

- **Assuming that $prob(\Theta)$ is uniform or normal**, let $\{\Theta^1, \Theta^2, \ldots, \Theta^n\}$ be a sequence of independent samples supplied to us by a random number generator. Now you should be able to use the following summation as a good approximation to the integral shown above:

$$E(g(\mathcal{X}, \Theta)) \approx \frac{1}{n} \sum_{i=1}^{n} g(\mathcal{X}, \Theta^i) \tag{19}$$

- However, in general, especially in Bayesian estimation, $prob(\Theta)$ can be expected to be arbitrary.

- Now we are faced with the challenge of how to draw a set of samples that would correspond to our current best guess for $prob(\Theta)$. **Assume for a moment that we have somehow acquired the ability to draw such a set of samples.** (We could, for example, use the Metropolis-Hastings Algorithm described in Section 3.4 of this tutorial for that purpose.) In that case, one would think that the summation shown above should still work as an approximation to the integral. That is, we should still be able to estimate the integral by

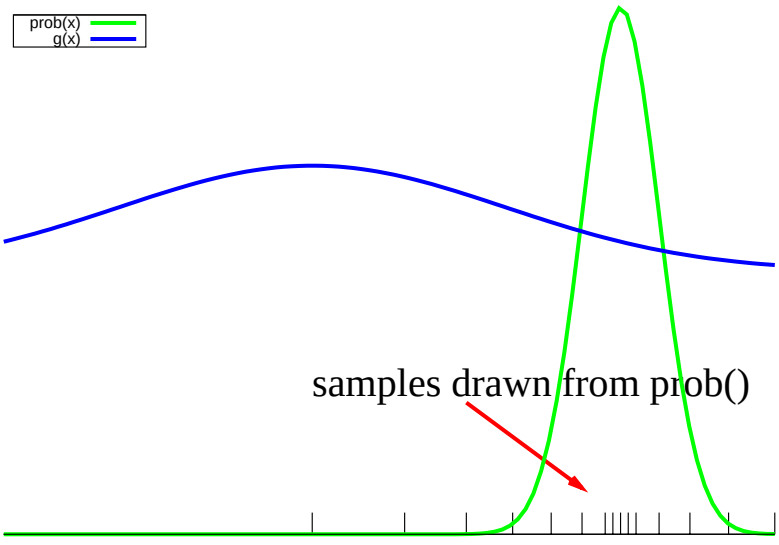
$$E(g(\mathcal{X}, \Theta)) \approx \frac{1}{n} \sum_{i} g(\mathcal{X}, \Theta^i) \tag{20}$$

  But, in practice, there is a deep practical flaw in this logic even if we have a great sampling algorithm, as you will soon see.

Back to TOC

3.2: A Pictorial Explanation of Why You Might Expect a Simple Summation of $g()$ to Work

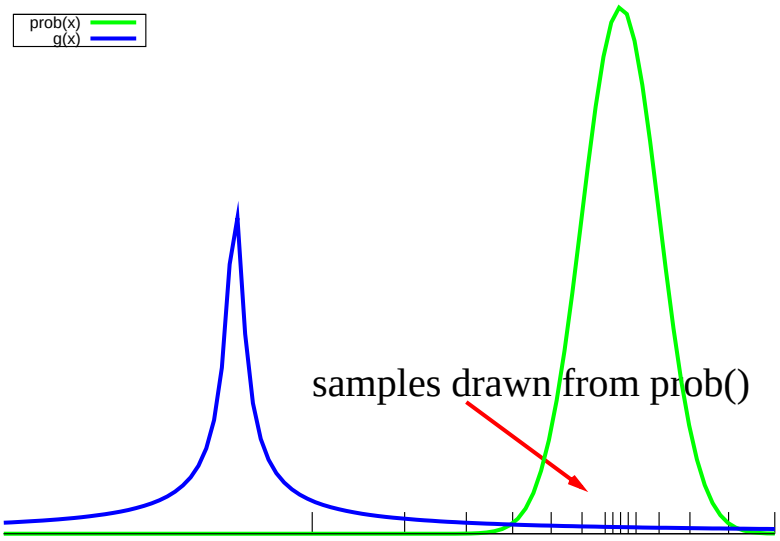
- As the following figure shows, if $g()$, shown by the blue curve in the figure, is a reasonably smooth function over its domain of definition, we can certainly expect a straightforward summation of $g()$ over the samples drawn from the probability distribution $prob()$, shown by the green curve, to approximate $\int g(\Theta) prob(\Theta)\, d\Theta$.



Back to TOC

3.3: A Pictorial Explanation of Why You Might NOT Expect a Simple Summation of $g()$ to Work

- However, if $g(\Theta)$ acquires its most significant values where $prob(\Theta)$ is expected to yield very few samples, as shown by the example in the figure below, a result obtained by $(1/n)\sum_i g(\Theta_i)$ over the samples drawn from the distribution $prob(\Theta)$ will certainly **NOT** yield a good approximation to the integral $\int g(\Theta)prob(\Theta)d\Theta$.



- This difficulty with the estimation of integrals involving probability distributions has led to the notion of **importance sampling**.

- For the purpose of solving the integral, we want to draw samples not taking into account only the distribution $prob(\Theta)$, but also where $g(\Theta)$ acquires significant values.

- We obviously want to think of $prob(\Theta)$ playing a probabilistic role vis-a-vis $g(\Theta)$. But, purely from the standpoint of the integration of the product $g(\Theta) \cdot prob(\Theta)$, the roles of $g()$ and $prob()$ are symmetric with respect to what gets weighted in a digital approximation to the integral.

Back to TOC

4: Importance Sampling

- By this time you should be comfortable with the fact that the Monte-Carlo method is a powerful technique for solving probabilistic integrals — that is, when the integrand is a product of two functions, with one of the two being a probability distribution.

- While in theory the Monte-Carlo approach sounds great, it does requires that the sampling points in space spanned by the parameter vector Θ be drawn in accordance with the probability distribution part of the integrand. **However, it is not so easy to draw usable sampling points from the parameter space should the the probability distribution NOT be compatible with the rest of the integrand as illustrated by the figures in Sections 3.2 and 3.3.**

- As explained in the subsections to follow, Importance Sampling based on a Proposal Distribution can be a solution to the dilemma presented above.

- When the probability distribution function is not compatible with the rest of the integrand, you try to come up with a Proposal Distribution that strikes a middle ground between the two.

- After you have decided that you want to use Importance Sampling based approach for solving a probabilistic integral, you are still faced with the following implementation issue: How exactly to draw samples from the probability distribution that you are interested in? The solutions to this problem are based on MCMC sampling. MCMC stands for Markov-Chain Monte-Carlo. In the last subsection of this section, I'll talk about the two commonly used MCMC samplers: Metropolis-Hastings and Gibbs.

Back to TOC

4.1: Definition of Importance Sampling

- Importance Sampling is an approach for a Monte-Carlo solution to integrals of the form

$$\int g(\mathcal{X}, \Theta) \cdot prob(\Theta)\, d\Theta \tag{21}$$

when we have no reason to believe that $g()$ is "compatible" with $prob()$.

- Importance sampling brings into play another distribution $q(\Theta)$, known as the **sampling distribution** or the **proposal distribution**, whose job is to help us do a better job of randomly sampling the values spanned by Θ.

- Before we show how $q(\Theta)$ can be used, let's rewrite the integral shown above in a way that accepts the interjection of this new distribution:

$$\frac{\int g(\mathcal{X}, \Theta)\, \frac{prob(\Theta)}{q(\Theta)}\, q(\Theta)\;\; d\Theta}{\int \frac{prob(\Theta)}{q(\Theta)}\, q(\Theta)\, d\Theta} \tag{22}$$

- You might wonder as to why we would want to gratuitously add a denominator in the above equation since none existed in the expression shown in Eq. (21). Consider this as a "remediation

guard" against the real possibility that at program execution time, the actual numerical values used for $\frac{prob(\Theta)}{q(\Theta)} \cdot q(\Theta)$ will depart from a true probability distribution.

- At least theoretically, Eq. (22) says that **our integral in Eq. (21) remains unchanged regardless of the choice of** $q(\Theta)$ (as long as dividing $prob()$ by $q()$ does NOT introduce any singularities in the integrand).

- **So we may now use** $q(\Theta)$ **for creating a random set of samples** $\{\Theta^1, \ldots, \Theta^n\}$ **for Monte-Carlo integration with the hope that these samples will be relevant to both** $g()$ **and** $prob()$**.**

- **That is, we can hope that the chosen** $q(\Theta)$ **would strike an acceptable middle ground between the needs of** $g()$ **and** $prob()$.

Back to TOC

4.2: Practical Implications of Using a "Designer" Proposal Distribution $q()$ for Monte-Carlo Integration

- Looking at Eq. (22), Importance Sampling tells us that we can use "any" proposal distribution $q(\Theta)$ to draw random samples from for the purpose of Monte Carlo integration provided we now think of $s(\Theta)$:

$$s(\Theta) \quad = \quad g(\mathcal{X}, \Theta) \, \frac{prob(\Theta)}{q(\Theta)} \tag{23}$$

  vis-a-vis $q(\Theta)$ the way we first thought of $g(\mathcal{X}, \Theta)$ vis-a-vis $prob(\Theta)$.

- Additionally, we must now also estimate the integration in the denominator of the form shown in Eq. (22). For this purpose, we must now estimate the integral $\int t(\Theta) q(\Theta) d\Theta$ with $t(\Theta) \; = \; prob(\Theta)/q(\Theta)$.

- The implication is that we must now first construct the weights at the random samples drawn according to the probability distribution $q(\Theta)$:

$$w^i \quad = \quad \frac{prob(\Theta^i)}{q(\Theta^i)} \tag{24}$$

  and then use the following estimation for our original integral

$$\frac{\frac{1}{n}\sum_{i=1}^{n} w^i \cdot g(\Theta^i)}{\frac{1}{n}\sum_{i=1}^{n} w^i} \tag{25}$$

- The weights w^i are known as the **importance weights**.

- But, of course, we are still faced with the question of how to choose the proposal distribution $q(\Theta)$.

- The Monte-Carlo integration formula shown above in Eq. (25) is used more often in the following form:

$$\sum_{i=1}^{n} W^i \cdot g(\Theta^i) \tag{26}$$

  where W^i are the **normalized versions of the importance weights** shown on the previous page:

$$W^i = \frac{w^i}{\sum_{j=1}^{n} w^j} \tag{27}$$

- It is in this form we will use the formula in Gaussian Particle Filtering.

Back to TOC

4.3: Is There a Way to Compare Different Proposal Distributions with Regard to Their Effectiveness?

- As mentioned in the previous subsection, a Monte-Carlo integration is an expectation of some entity $g()$:

$$\int g(\Theta) \cdot prob(\Theta)\, d\Theta \quad = \quad E(g(\Theta)) \quad \approx \quad \sum_{i=1}^{n} W^i \cdot g(\Theta^i) \tag{28}$$

- We may associate a variance with this estimate. We will call it the **Monte Carlo Variance**:

$$\int [g(\Theta) - E(g(\Theta))]^2 \cdot prob(\Theta)\, d\Theta \quad = \quad Var(g(\Theta)) \tag{29}$$

- By treating the left hand side in the same manner as the original integration problem, we can write a discrete approximation to the variance in terms of the random samples used in the Monte Carlo estimate.

- The goal of importance sampling is to choose the proposal distribution $q(\Theta)$ that minimizes the Monte-Carlo variance.

- It has been shown that the proposal distribution that minimizes the Monte-Carlo Variance is given by

$$q(\Theta) \quad \propto \quad |g(\Theta) \cdot prob(\Theta)| \tag{30}$$

  which makes intuitive sense.

- Although it is comforting to think of the above as a good strategy, it is not a complete solution to the choosing of the proposal distribution. The product $g(\Theta)prob(\Theta)$ may not sample $g(\Theta)$ properly because the former goes to zero where it should not.

- It is not uncommon to see people using the heuristic

$$q(\Theta) \quad \propto \quad |g(\Theta)| \tag{31}$$

  but now you run into the problem of ensuring that the weights w^i do not become ill conditioned at all the sampling points.

Back to TOC

4.4: We are Still Faced with the Problem of How to Draw Samples According to a Prescribed Distribution — MCMC Samplers

- It might be easier to carry out the Monte Carlo integration with the proposal distribution $q(\Theta)$ than it was with the original $prob(\Theta)$, but we are still faced with the problem of having to draw sampling point in the Θ space that would correspond to the $q(\Theta)$ distribution. How does one do that?

- As it turns out, how to draw samples whose probability distribution would correspond to a given proposal $q(\Theta)$ is an interesting challenge unto itself — especially if $q(\Theta)$ is a complicated multi-modal function.

- In the rest of this section, I'll use the notation $p(\mathbf{x})$ to denote the distribution whose samples we wish to draw from for the purpose of Monte Carlo integration.

- So the goal in this section is to estimate the integral $\int_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) f(\mathbf{x}) d\mathbf{x}$, where $p(\mathbf{x})$ is a probability distribution and where $f(\mathbf{x})$ is some arbitrary function. As mentioned earlier if $p(\mathbf{x})$ is simple, like a uniform or a Gaussian distribution, the N samples $\mathbf{x}_i$, as you would expect, can be drawn easily from $p(\mathbf{x})$ using

run-of-the-mill function calls to random-number generators. **With such samples, an unbiased estimate of the integral $\int_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x})f(\mathbf{x})d\mathbf{x}$ would be given by the summation** $(1/N)\sum_{i=1}^{N} f(\mathbf{x}_i)$. Unfortunately, this standard Monte-Carlo approach does not work when $p(\mathbf{x})$ is a complicated probability distribution — **simply because it is non-trivial to sample complicated probability distributions algorithmically**.

- Modern approaches to drawing samples from an arbitrary probability distribution $p(\mathbf{x})$ for the purpose of Monte-Carlo integration are based on MCMC sampling, where MCMC stands for *Markov-Chain Monte-Carlo*.

- **MCMC sampling is based on the following intuitions:**

  1. For the very first sample, $\mathbf{x}_1$, you accept any value that belongs to the domain of $p(\mathbf{x})$, that is, any randomly chosen value $\mathbf{x}$ where $p(\mathbf{x}) > 0$. At this point, any sample is as good as any other.

  2. For the next sample, you again randomly choose a value from the interval where $p(\mathbf{x}) > 0$ but now you must "reconcile" it with what you chose previously for $\mathbf{x}_1$. Let's denote the value you are now looking at as $\mathbf{x}_*$ and refer to it as our *candidate* for $\mathbf{x}_2$.

  3. By having to "reconcile" the candidate $\mathbf{x}_*$ with the previously selected $\mathbf{x}_1$ before accepting the candidate as the next sample, here is what I mean: For obvious reasons, your desire should be to select a large number of samples in the vicinity of the peaks in $p(\mathbf{x})$ and, relatively speaking, fewer samples where $p(\mathbf{x})$ is close to 0. You can capture this intuition by examining the ratio $a1 = \frac{p(\mathbf{x}_*)}{p(\mathbf{x}_1)}$. If $a1 > 1$, then accepting

$\mathbf{x}_*$ as $\mathbf{x}_2$ makes sense because your decision would be biased toward placing samples where the probabilities $p(\mathbf{x})$ are higher.

4. However, should $a1 < 1$, you need to exercise some caution in accepting $\mathbf{x}_*$ for $\mathbf{x}_2$, as explained on the next page.

5. While obviously any sample $\mathbf{x}_*$ where $p(\mathbf{x}_*) > 0$ is a legitimate sample, you nonetheless want to accept $\mathbf{x}_*$ as $\mathbf{x}_2$ with some hesitation when $a1 < 1$, your hesitation being greater the smaller the value of $a1$ in relation to unity. You capture this intuition by saying that let's accept $\mathbf{x}_*$ as $\mathbf{x}_2$ with probability $a1$.

6. In an algorithmic implementation of the above stated intuition, you fire up a random-number generator that returns floating-point numbers in the interval $(0, 1)$. Let's say the number returned by the random-number generator is u. You accept $\mathbf{x}_*$ as $\mathbf{x}_2$ if $u < a1$.

- **It is these intuitions that form the foundation of the original Metropolis algorithm for drawing samples from a given probability distribution.**

- Since each sample chosen in this manner depends on just the sample selected previously, a sequence of such samples forms a **Markov chain**.

- Since the sequence of samples generated in this manner forms a Markov chain, this approach to drawing samples from a distribution for the purpose of Monte-Carlo integration of complex integrands is commonly referred to as the **Markov-Chain**

Monte-Carlo sampler, or, more conveniently, as the **MCMC sampler.**

- To be precise, the Metropolis algorithm for MCMC sampling uses what is known as a *proposal distribution*[1] for MCMC sampling $q(\mathbf{x}_*|\mathbf{x}_{t-1})$ to return a candidate $\mathbf{x}_*$ for the current sample $\mathbf{x}_t$ given the previous sample $\mathbf{x}_{t-1}$ and requires that $q(.|.)$ be symmetric with respect to its two arguments if you want the theoretical guarantee that the first-order probability distribution of the samples of the Markov Chain converge to the desired density $p(\mathbf{x})$.

- This symmetry restriction on the proposal distribution **is removed** in a more general algorithm known as the Metropolis-Hastings (MH) algorithm.

- In the Metropolis-Hastings algorithm, however, the ratio that is tested for the acceptance of the candidate $\mathbf{x}_*$ is now given by the product $a = a1 \times a2$ where $a2 = \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_*)}{q(\mathbf{x}_*|\mathbf{x}_{t-1})}$ where $a1$ was defined earlier in Step 3 of the 6-Step original Metropolis algorithm. If $a \geq 1$, we accept the candidate $\mathbf{x}_*$ immediately for the next sample. Otherwise, we only accept it with probability a.

- If we think of $a1$ for the case of the Metropolis algorithm and of $a = a1 \times a2$ for the case of Metropolis-Hastings algorithm as the probability with which the candidate $\mathbf{x}_*$ is accepted, we can write for the Metropolis algorithm:

[1] Do not confuse the *proposal distribution* $q(\cdot|\cdot)$ here with the proposal distribution $q()$ in Section 4.1 where I was explaining the basic idea of importance sampling.

$$a1 \;=\; \min\Big(\frac{p(\mathbf{x}_*)}{p(\mathbf{x}_{t-1})},\;\; 1\Big) \tag{32}$$

and for the Metropolis-Hastings algorithm:

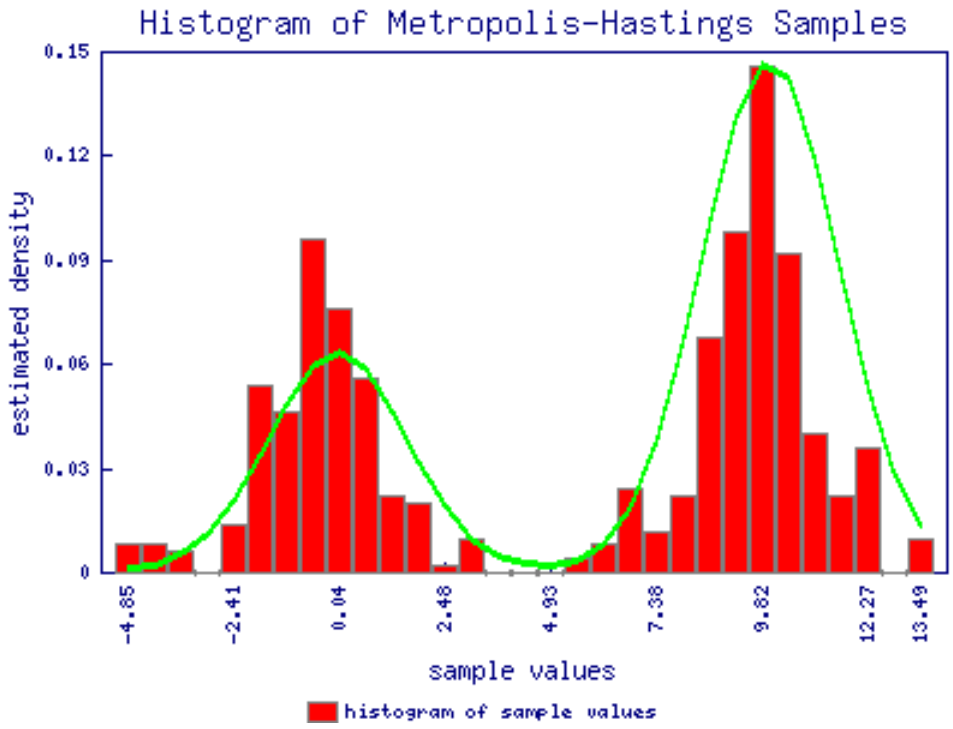
$$a \;=\; \min\Big(\frac{p(\mathbf{x}_*)q(\mathbf{x}_{t-1}|\mathbf{x}_*)}{p(\mathbf{x}_{t-1})q(\mathbf{x}_*|\mathbf{x}_{t-1})},\;\; 1\Big) \tag{33}$$

- Obviously, should you happen to use for the Metropolis-Hastings algorithm a $q(.|.)$ that is symmetric with respect to its two arguments, you'll have $a = a1$.

- **The Metropolis-Hastings (MH) algorithm is the most popular algorithm for MCMC sampling.** The algorithm is straightforward to implement. The reader may wish to check out my Perl implementation `Metropolis_Hastings.pl` in Section 26.7 of

  `https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture26.pdf`.

- That script generates MCMC samples for the target density function $p(\mathbf{x}) = 0.3 \cdot e^{-0.2\mathbf{x}^2} + 0.7 \cdot e^{-0.2(\mathbf{x}-10)^2}$ that is shown by the line plot in the figure on the next page. The histogram for the MCMC samples produced by the script is based on only the first 500 samples of the sequence. This histogram is shown as a bar graph in the same figure. [Ordinarily, it is best to discard several hundred samples at the beginning of such a sequence to eliminate the effects of initialization. After these initial samples are rejected, the rest of the sequence would follow even more closely the desired density.]



- As mentioned earlier, the MH algorithm requires us to specify a **proposal** density function $q(x|y)$.

- The proposal density function that I used in my Perl script is $q(x|y) = \mathcal{N}(y, 100)$, that is, it is a normal density **that is centered at the previous sample** with a standard deviation of 10. This standard-deviation was chosen keeping in mind the interval $(-5.0, 15.0)$ over which $p(\mathbf{x})$ is defined with values not too close to zero, as shown by the line plot in above figure.

- For some final observations related to the MH algorithm for MCMC sampling:
  - Commonly used proposal densities include uniform, normal (as in my

Perl script), and χ^2. With each of these choices, you also have to decide how "wide" a proposal density to use. In my Perl script, I used a standard deviation of 10 keeping in mind the width of the target density function $p(\mathbf{x})$.

- It is possible for the width of the proposal density to be either too small or too large. If too small, your MCMC chain may get stuck in just one of the modes of a multimodal target distribution. If too wide, your acceptance rate for the candidate samples may be too low.

- One final issue related to the samples yielded by the Metropolis-Hastings algorithm is the correlatedness of the adjacent samples. The very nature of the derivation of the samples implies that there can exist significant correlations between adjacent samples. **However, it is important to Monte Carlo integration that the samples be independent. In order to meet this constraint, it is common to take every n^{th} sample from the MCMC chain, with n chosen suitably to make the retained samples independent. This is referred to as the *thinning* of the MCMC samples.**

- On last topic I want to mention just in passing is the **Gibbs sampler** as a special case of the MH MCMC sampler.

- First note that the notation $\mathbf{x}$ in our discussion on MCMC sampling was meant to stand for a vector variable of an arbitrary number of dimensions.

- Assuming that $\mathbf{x}$ is n-variate, MCMC sampling with the MH algorithm directly gives us a sequence of samples in the

n-dimensional space spanned by $\mathbf{x}$. After the initialization effects have died down, this sequence can be expected to stabilize in a stationary distribution that is a good approximation to the n-variate joint distribution $p(\mathbf{x})$ over n variables.

- **The Gibbs MCMC sampler samples each dimension of x separately through the univariate conditional distribution along that dimension vis-a-vis the rest.** Before I explain why this makes sense, let me introduce some notation for such conditional distributions.

- I'll make the individual components of $\mathbf{x}$ explicit by writing $\mathbf{x} = (x_1, \ldots, x_n)^T$.

- I will also write $\mathbf{x}^{(-i)} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)^T$.

- Let's now focus on the n univariate conditional distributions: $p(x_i|\mathbf{x}^{(-i)})$ for $i = 1, \ldots, n$. Keep in mind the fact that a conditional distribution for the component scalar variable x_i makes sense only when the other $n-1$ variables in $\mathbf{x}^{(-i)}$ are given constant values.

- **Gibbs MCMC sampling is based on the observation that even when the joint distribution $p(\mathbf{x})$ is multimodal, the univariate conditional distribution for each x_i when all the other variables are held constant is likely to be approximable by a relatively easy unimodal distribution, such as uniform or normal.**

- **What that implies is that the individual scalar variables can be sampled through straightforward function calls to standard software packages dealing with the production of random numbers without resort to the notion of acceptance probabilities as in the Metropolis-Hastings algorithm.**

- Taking the above observations into account, one could set up a Gibbs MCMC sampler in the following manner:

  - For initialization, we choose random values for the variables x_2 through x_n. We denote these by $x_2^{(0)}, \ldots, x_n^{(0)}$. These are our initial samples for the $n-1$ scalar variables x_2 through x_n.

  - We next draw a sample for x_1 by

$$x_1^{(1)} \quad \sim \quad p\Big(x_1 \;\Big|\; \mathbf{x}^{(-1)} = (x_2^{(0)}, \ldots, x_n^{(0)})\Big) \tag{34}$$

    where $x_1^{(1)}$ means that it is a sample for the variable x_1 produced at the first iteration of the sampler.

  - We next draw a sample for x_2 by

$$x_2^{(1)} \quad \sim \quad p\Big(x_2 \;\Big|\; x_1 = x_1^{(1)}, \mathbf{x}^{(-1,-2)} = (x_3^{(0)}, \ldots, x_n^{(0)})\Big) \tag{35}$$

  - In the same manner, we draw samples for the scalars x_j, with $j = 3 \ldots n$, In the conditional probability for each x_j, we use the just drawn $x_i^{(1)}$ for $i = 1 \ldots j-1$ and the initialization values $x_i^{(0)}$ for $i = j+1, \ldots, n$.

- In this manner, we complete one "scan" through all the n dimensions of $\mathbf{x}$. In the next scan, we now use the previously calculated sample values for the conditioning variables and proceed in exactly the same manner as above.

- After K such scans through the component variables, we end up with K sampling points for vector variable $\mathbf{x}$.

Back to TOC

5: Application to Time-Varying Systems: Bayesian Estimation of the State

- A classic case of a time-varying system would be tracking an object or a human in a video. Imagine for a moment that you you have a segmentation algorithm that can isolate out a human in, say, the first frame of a video and now you wish to track that individual through all their motions in the subsequent frames.

- This problem is made very challenging by the fact that the object you are tracking might occasionally become partially or fully obscured by other objects in the scene.

- The algorithmic challenge here is how best to specify a state vector for representing the object being tracked and, then, how to update the state vector from frame to frame. As you will see in this section, you will need to make certain assumptions about the temporal evolution of the state vector in order to end up with a computationally feasible algorithm.

- It is beyond the scope of this tutorial to address the problem of how to patch up the disconnections in a track caused by object obscurations.

- If you are interested in the full problem of object tracking in the presence of obscurations, you might enjoy the research described in the following publication from Purdue Robot Vision Lab:

`https://engineering.purdue.edu/RVL/Publications/ITAE_AeschlimanParkKak_v2.pdf`

Back to TOC

5.1: Probabilistic Modeling of a Dynamic System

- There are two different but **equivalent** ways of looking at a dynamic system, View 1 and View 2 as described below.

- **View 1:** We may represent a dynamic system by an N-dimensional random vector $\mathbf{x}$ and talk about a sequence $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k, \ldots\}$ of the values of this random vector as a function of time k.

- For example, if we are tracking a human being in video imagery, we may think of $\mathbf{x}$ as the N values of the N parameters of the pose of the human being. All of these N values are likely to change with time k. We can use $p(\mathbf{x}_k = \mathbf{a})$ to denote the probability that the random vector $\mathbf{x}$ has acquired the specific value $\mathbf{a}$ at time k.

- We may now think of the conditional probability $p(\mathbf{x}_{k+1}|\mathbf{x}_k)$ as informing us how the values of the random vector $\mathbf{x}$ will transition from one time instant to the next.

- The above view is typical of how Kalman Filters and Particle Filters are developed.

- **View 2:** Or, we may conceive of the system being in one of N possible states at any given time instance k and think of $\mathbf{x}$ as representing a distribution of probabilities over all possible states.

- At any given time instant k, the system must be in exactly one of N states, except that we do not know which one. Our ignorance regarding the exact state the system is in at time k is reflected by the probability distribution $\mathbf{x}$ over all N states. We refer to this distribution as a **state vector**.

- Going back to the example of tracking a human being in video imagery, in View 2, we associate a discrete but **exhaustive** set of N pose states with the human. Assuming for a moment that $N = 4$, we expect the human to be in **exactly one** of those 4 states at all time instants.

- The time evolution of the state vector may now be represented by the changing distribution of probabilities stored in the state vector. That is, we now talk about a sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k, \ldots$ to represent the time varying distribution of the probabilities over the states.

- Obviously, in View 2, it must be the case that $\sum_{i=1}^{N} \mathbf{x}_{k,i} = 1$ at every time instant k since $\mathbf{x}_{k,i}$ is the probability of the system being in the i^{th} state in the state vector $\mathbf{x}_k$. (**Note that this normalization does not apply to View 1.**)

- In View 2, **the conditional probability $p(\mathbf{x}_{k+1}|\mathbf{x}_k)$ becomes an $N \times N$ matrix of state transition probabilities.**

- In the rest of this section, I'll assume View 1, but, in keeping with much of the literature, I'll call $\mathbf{x}$ the **state vector**. **One can indeed make the claim that the values for all of the N important variables of the system at time k represents the state of the system at that time instant.**

- An aside: In classical theoretical computer science, state spaces are associated with deterministic and nondeterministic finite state machines. But that discussion focuses on state transitions brought about by specific inputs and, in at least the classical treatment of finite state machines, there are no probabilities involved. Our current discussion does not deal with causal issues related to state transitions. In our current discussion, state transitions occur for reasons beyond our control. All we are interested in is in figuring out the probability distributions over the various states as we record the observables.

Back to TOC

5.2: Modeling the Time Evolution of the State Vector

- Regardless of which View you use, we assume that the state vector $\mathbf{x}$ (or the random data vector $\mathbf{x}$) is **unobservable** directly.

- We also assume that the time evolution of $\mathbf{x}$ can be represented by a possibly **nonlinear but known relationship** $f()$ between the state vector and a noise process $\mathbf{u}$ **whose distribution is known to us in advance**:

$$\mathbf{x}_{k+1} \quad = \quad f_k(\mathbf{x}_k, \mathbf{u}_k) \qquad k = 0, \ldots \tag{36}$$

- Theory does not require that the dimensionality of the state vector $\mathbf{x}$ and noise vector $\mathbf{u}$ be the same. So we can think of $f_k()$ as $f_k : \mathbb{R}^N \times \mathbb{R}^M \rightarrow \mathbb{R}^N$.

- **We will refer to Eq. (36) as our Process Model.**

- Our process model defines a **Markov process since, if we fix the value of $\mathbf{x}_k$, the only uncertainty in $\mathbf{x}_{k+1}$ is caused by the random contribution from $\mathbf{u}_k$ that is specific to time step k.** This fact translates into the following state transition probability distribution:

$$
\begin{aligned}
p(\mathbf{x}_{k+1}|\mathbf{x}_k) &= \int p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k) \cdot p(\mathbf{u}_k|\mathbf{x}_k)\, d\mathbf{u}_k \\
&= \int p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k) \cdot p(\mathbf{u}_k)\, d\mathbf{u}_k \\
&= \int \delta(\mathbf{x}_{k+1} - f_k(\mathbf{x}_k, \mathbf{u}_k)) \cdot p(\mathbf{u}_k)\, d\mathbf{u}_k \qquad (37)
\end{aligned}
$$

where the second equality follows from the fact that the process noise at each time step is independent of the state that corresponds to that time step. That is, $p(\mathbf{u}_k|\mathbf{x}_k) = p(\mathbf{u}_k)$. The last equality in which $\delta()$ is the Dirac delta function, follows from the fact that once we fix $\mathbf{x}_k$ and $\mathbf{u}_k$, finding $\mathbf{x}_{k+1}$ is a deterministic calculation involving $f_k()$.

Back to TOC

5.3: Relating the Observables to the State

- We further assume that what we observe is a sequence of vectors $\mathbf{y}_k$, $k = 1, 2, \ldots$ and that these observables are related to the unobservable state $\mathbf{x}_k$ by a **possibly nonlinear but known** relationship $h()$:

$$\mathbf{y}_k \quad = \quad h(\mathbf{x}_k, \mathbf{v}_k) \qquad k = 1, \ldots \tag{38}$$

  where $\mathbf{v}_k$ represents the observation noise random process that can nonlinearly affect the observed data $\mathbf{y}_k$.

- **It is assumed that the distribution for $\mathbf{v}$ is known.**

- Again, there is no requirement that the state $\mathbf{x}$ and the observation noise $\mathbf{v}$ be of the same dimension. The theory does not even require that the state $\mathbf{x}$ and the observation $\mathbf{y}$ be of the same dimension. So, in general, we can think of $h()$ as $h : \mathbb{R}^N \times \mathbb{R}^M \rightarrow \mathbb{R}^P$.

- Eq. (38) is called the **observation model**.

Back to TOC

5.4: Two Interdependent Problems

- Given the process model and the observation model, and given all of the observed data up to and including the current time instant, we can now address the following two problems:

  1. **The Filtering Distribution Problem:** We want to estimate recursively the distribution $p(\mathbf{x}_k|\mathbf{y}_{1:k})$.

  2. **The Predictive Distribution Problem:** We want to estimate recursively the distribution $p(\mathbf{x}_{k+1}|\mathbf{y}_{1:k})$.

- In stating the two problems, I have used the notation $\mathbf{y}_{1:k} = \{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$.

- We assume that we know the starting distribution $p(\mathbf{x}_0)$ at time $k = 0$ in order to get the recursive estimation started.

- The two problems listed above are not independent. As we will see shortly, they are highly interdependent in any approach to the recursive estimation of the state.

- Estimating the filtering distribution requires the prediction

provided by the estimation of the predictive distribution and vice versa.

Back to TOC

5.5: Fundamental Equation for the Recursive Estimation of the Filtering Distribution

- With multiple invocations of the chain rule, we can write the filtering distribution as

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{y}_{1:k}) &= \frac{p(\mathbf{y}_{1:k}, \mathbf{x}_k)}{p(\mathbf{y}_{1:k})} \\ &= \frac{p(\mathbf{y}_k|\mathbf{y}_{1:k-1}, \mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k-1})p(\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})p(\mathbf{y}_{1:k-1})} \\ &= \ldots \\ &= \frac{p(\mathbf{y}_k|\mathbf{x}_k) \cdot p(\mathbf{x}_k|\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})} \\ &= C_k \cdot p(\mathbf{y}_k|\mathbf{x}_k) \cdot p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) \end{aligned} \tag{39}$$

  where, in the last expression on the right, the normalization constant C_k is given by

$$\begin{aligned} C_k &= (p(\mathbf{y}_k|\mathbf{y}_{1:k-1}))^{-1} \\ &= \left(\int p(\mathbf{y}_k|\mathbf{x}_k) \cdot p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) \, d\mathbf{x}_k \right)^{-1} \end{aligned} \tag{40}$$

- In a recursive processing framework, the last term on the RHS in Eq. (39) can be interpreted **as providing prediction** at time step k and the second term inferred from the observation model. That

brings us to the estimation of predictive distribution discussed in the next subsection.

- Eq. (39) is our fundamental equation for the recursive estimation of the filtering distribution.

Back to TOC

5.6: Fundamental Equation for the Recursive Estimation of the Predictive Distribution

- We can write for the predictive distribution

$$\begin{aligned} p(\mathbf{x}_{k+1}|\mathbf{y}_{1:k}) &= \frac{p(\mathbf{x}_{k+1}, \mathbf{y}_{1:k})}{p(\mathbf{y}_{1:k})} \\ &= \frac{\int p(\mathbf{x}_{k+1}, \mathbf{x}_k, \mathbf{y}_{1:k})\, d\mathbf{x}_k}{p(\mathbf{y}_{1:k})} \\ &= \frac{\int p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{y}_{1:k}) \cdot p(\mathbf{x}_k, \mathbf{y}_{1:k})\, d\mathbf{x}_k}{p(\mathbf{y}_{1:k})} \\ &= \frac{\int p(\mathbf{x}_{k+1}|\mathbf{x}_k) \cdot p(\mathbf{x}_k, \mathbf{y}_{1:k})\, d\mathbf{x}_k}{p(\mathbf{y}_{1:k})} \\ &= \frac{\int p(\mathbf{x}_{k+1}|\mathbf{x}_k) \cdot p(\mathbf{x}_k|\mathbf{y}_{1:k}) \cdot p(\mathbf{y}_{1:k})\, d\mathbf{x}_k}{p(\mathbf{y}_{1:k})} \\ &= \int p(\mathbf{x}_{k+1}|\mathbf{x}_k) \cdot p(\mathbf{x}_k|\mathbf{y}_{1:k})\, d\mathbf{x}_k \end{aligned} \tag{41}$$

  where the fourth equality on the right follows from the Markov assumption regarding the state transitions.

- Of the two terms in the integrand in Equation (10), the second can be interpreted as being supplied by an estimate of the filtering distribution at time k and the first can be obtained from the process model.

- Eq. (41) constitutes the fundamental equation for the estimation

of the predictive distribution.

Back to TOC

5.7: Solving the Filtering and the Predictive Equations Simultaneously Under Different Assumptions

- **The Simplest Case:**

- When the relationships $f()$ and $h()$ can be assumed to be **linear**, and the prior $p(\mathbf{x}_0)$ and the noise processes $\mathbf{u}$ and $\mathbf{v}$ to be Gaussian, we can show that the filtering and the predictive distributions will always stay Gaussian as time progresses.

- In this case, the optimal solution is provided by the Kalman filter.

- The optimal Bayesian solution now consists of **propagating just the mean and the covariance** through time, which is what the **Kalman Filter** does.

- **The Not So Difficult Case:**

- When the process model and the observation model [that is, $f()$ and $h()$] are piecewise linear and the assumption of Gaussian for $p(\mathbf{x}_0)$, on the one hand, and for the noise processes u and v, on the other, hold, it may be sufficient to propagate only the mean

and the covariance through time. This can be done with an **Extended Kalman Filter**.

- **Moderately Difficult Case:**

- When the process model and/or the observation model are **nonlinear** [that is, $f()$ and $h()$ are nonlinear] but the assumption of the prior at time 0 and the noise processes being Gaussian still holds, we may be able to get away with propagating just the means and the covariances through time using the **Gaussian Particle Filter**.

- **The Most Difficult Case:**

- When the process model and the observation model are **nonlinear** and we cannot assume the prior $p(\mathbf{x}_0)$ and/or the noise processes to be Gaussian, then we must carry out full-blown Bayesian estimation at each step. This may require a **particle filter with resampling** at every time step.

Back to TOC

5.8: Gaussian Particle Filters

- Let's go back to the two fundamental (but interdependent) equations (39) and (41) that yield the filtering distribution and the predictive distribution:

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = C_k \cdot p(\mathbf{y}_k|\mathbf{x}_k) \cdot p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) \tag{42}$$

$$p(\mathbf{x}_{k+1}|\mathbf{y}_{1:k}) = \int p(\mathbf{x}_{k+1}|\mathbf{x}_k) \cdot p(\mathbf{x}_k|\mathbf{y}_{1:k})\, d\mathbf{x}_k \tag{43}$$

- The first equation above can be interpreted as saying that the current best filtering distribution is a product of the likelihood distribution (based on just the current hidden state) and the predictive distribution for the current state taking into account all the past observations. The second equation can be interpreted as saying that the prediction for the next time instant is a summation over all possibilities of the applicable transitions starting from the current filtering distribution.

- In Gaussian Particle Filtering, we assume that the filtering distribution on the left in Eq. (42) is a Gaussian given by $\mathcal{N}(\mathbf{x}_k;\ \mu_{\mathbf{x}_k}, \mathbf{\Sigma}_{\mathbf{x}_k})$.

- Similarly, we assume that the predictive distribution on the left in Eq. (43) is a Gaussian given by $\mathcal{N}(\mathbf{x}_{k+1};\ \widehat{\mu}_{\mathbf{x}_{k+1}}, \widehat{\mathbf{\Sigma}}_{\mathbf{x}_{k+1}})$.

- So we are faced with the following two problems:
  - How do we compute $(\mu_{\mathbf{x}_k}, \mathbf{\Sigma}_{\mathbf{x}_k})$ from the predicted $(\widehat{\mu}_{\mathbf{x}_k}, \widehat{\mathbf{\Sigma}}_{\mathbf{x}_k})$?
  - How do we estimate the prediction $(\widehat{\mu}_{\mathbf{x}_k}, \widehat{\mathbf{\Sigma}}_{\mathbf{x}_k})$?

Back to TOC

5.9: Estimation of $(\mu_{\mathbf{x}_k}, \mathbf{\Sigma}_{\mathbf{x}_k})$ in Kotecha-Djuric Gaussian Particle Filtering Algorithm

- In order to compute the filtering distribution parameters $(\mu_{\mathbf{x}_k}, \mathbf{\Sigma}_{\mathbf{x}_k})$ at time k, we note from Eq. (42):

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \;\approx\; C_k \cdot p(\mathbf{y}_k|\mathbf{x}_k) \cdot \mathcal{N}(\mathbf{x}_k;\; \widehat{\mu}_{\mathbf{x}_k}, \widehat{\mathbf{\Sigma}}_{\mathbf{x}_k}) \tag{44}$$

- Estimating the parameters $(\mu_{\mathbf{x}_k}, \mathbf{\Sigma}_{\mathbf{x}_k})$ of the filtering distribution on the left requires we take various moments of the product on the right. For example, to find $\mu_{\mathbf{x}_k}$ requires that we multiply the right hand side of the above equation by $\mathbf{x}_k$ and integrate over all possible values for $\mathbf{x}_k$.

- **This is where Monte-Carlo rears its head again.**

- We are obviously again faced with the computation of the following sort of integral

$$E_{some\_power\_of\_x_k} \quad = \quad \int g(\mathbf{x}_k) \cdot prob(\mathbf{x}_k)\, d\mathbf{x}_k \tag{45}$$

- So, as described in Section 4 on Importance Sampling, we first choose an **importance sampling distribution** $q(\mathbf{x}_k)$ and draw its samples

$$\{\mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)}, \ldots, \mathbf{x}_k^{(M)}\} \tag{46}$$

- We refer to these samples as **particles**.

- We now treat the entire right hand side in Eq. (44) as our $prob(\mathbf{x}_k)$ for the calculation of the **importance weights**:

$$w_k^{(j)} = \frac{p(\mathbf{y}_k|\mathbf{x}_k^{(j)}) \cdot \mathcal{N}(\mathbf{x}_k = \mathbf{x}_k^{(j)};\ \widehat{\mu}_{\mathbf{x}_k}, \widehat{\mathbf{\Sigma}}_{\mathbf{x}_k})}{q(\mathbf{x}_k^{(j)})} \tag{47}$$

  We can ignore the normalization constant C_k in Eq. (44) because it is going to drop out of our **normalized importance weights** anyway. These are shown below:

$$W_k^{(j)} = \frac{w_k^{(j)}}{\sum_{i=1}^{M} w_k^{(i)}} \tag{48}$$

- Note that the first term in the numerator of Eq. (14) is trivially computed because $\mathbf{y}_k$ is a specific observed value, $\mathbf{x}_k^{(j)}$ is a specific sample value for $\mathbf{x}_k$, and, even more importantly, because we assumed that $f()$ and $h()$ are **known** functions and the noise processes u and v with **known** distributions.

- Having calculated the normalized importance weights $W_k^{(j)}$, let's now go back to the goal of using Monte-Carlo integration to calculate the various moments of the filtering probability distribution shown in Equation (13).

- The Monte-Carlo formula says we can estimate the mean and the covariance of the filtering distribution at time step k by using the following weighted sums of the samples drawn from the importance sampling distribution:

$$\mu_k = \sum_{j=1}^{M} W_k^{(j)} \cdot \mathbf{x}_k^{(j)} \tag{49}$$

$$\mathbf{\Sigma}_k = \sum_{j=1}^{M} W_k^{(j)} \cdot (\mu_k - \mathbf{x}_k^{(j)})(\mu_k - \mathbf{x}_k^{(j)})^T \tag{50}$$

- We obviously can derive similar formulas for estimating the predicted parameters $\widehat{\mu}_{\mathbf{x}_k}, \widehat{\mathbf{\Sigma}}_{\mathbf{x}_k}$ that are needed in the importance weight calculations in Eq. (47).

- These can be obtained in the same manner as the parameters for the filtering distribution at time step k.

Back to TOC

5.10: How Does the Importance Sampling at Time k Relate to the Importance Sampling at Time $k+1$

- Let's say the particles at time k are given by

$$\{\mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)}, \ldots, \mathbf{x}_k^{(M)}\} \tag{51}$$

  To construct the set of M particles at time $k+1$, we propagate each particle through the process model. Therefore, the j^{th} particle at time $k+1$ is given by

$$\mathbf{x}_{k+1}^{(j)} \;=\; f(\mathbf{x}_k^{(j)}, \mathbf{u}_{k+1}^{(j)}) \qquad j = 1, \ldots, M \tag{52}$$

- where $\mathbf{u}_{k+1}^{(j)}$ is a randomly drawn j^{th} sample from the process model noise distribution $\mathcal{N}(\mathbf{u}; \mu_{\mathbf{u}}, \mathbf{\Sigma}_{\mathbf{u}})$.

- As we propagate the particles through successive time steps, they are only getting dithered by the process model noise.

- In more general approaches to particle filtering, as you calculate the posterior for the state vector at each time instant, you choose the importance sampling distribution that is adapted to that posterior in order to make the prediction for the next time

instant. **This is referred to particle filtering with resampling** or as **sequential importance sampling** (SIS).

Back to TOC

6: Acknowledgments

Henry Medeiros was kind enough to share with me his wonderful insights in recursive estimation and particle filtering. It gives me great pleasure to interrupt Henry when he least wants to be interrupted. It wouldn't be any fun otherwise. [This was written during the days when Henry was a Ph.D student in RVL.]

I also benefited a great deal from the paper "*Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation*," by Gordon, Salmond, and Smith, IEE Proceedings, April 1993. **It is an excellent paper — a must read.**

The paper "*Gaussian Particle Filtering*," by Kotecha and Djuric in IEEE Trans. Signal Processing, Oct 2003, was the source of the Gaussian Particle Filter based ideas in this presentation. **I would not have fully understood this paper without also reading the paper by Gordon et al.**

The tutorial was updated in February 2012 to fix the several typos discovered by German Holguin and to also incorporate German's suggestion that, at least from an implementation perspective, it makes more sense in Section 5 to start the observations $\mathbf{y}$ at index 1 while the state $\mathbf{x}$ starts at index 0. That eliminates the requirement that we have prior knowledge of the predictive distribution $p(\mathbf{x}_1|\mathbf{y}_0)$ in order to get the recursive estimation started. Another minor update of the tutorial was carried out in December 2012.