ASEN 5044, Fall 2018

Statistical Estimation for Dynamical Systems

Lecture 11 [Special Topic Lecture #1]: Simulated Sampling (Monte Carlo) and Expected Value Approximations

Prof. Nisar Ahmed (Nisar.Ahmed@Colorado.edu)

Fri 9/28/2018

University of Colorado Boulder

Aerospace Engineering

Today…

- How to numerically approximate analytically intractable expected values via direct Monte Carlo simulation?
- How to simulate sampling from basic probability distributions?
- How to represent and sample from mixture models of complex pdfs?

Recall: Expectation Operator

- What is the "expected value" of some arbitrary function g(x) of random var X?

<u>Discrete Case</u>

$$E[g(x)] = \sum_{i=1}^{N_x} g(x=i)P(x=i)$$

<u>Continuous Case</u>

$$E[g(x)] = \int_{-\infty}^{\infty} g(x)p(x)dx$$

Ideally, would like to find analytical/closed-form values, but…

- can have very large N_x (many possible discrete outcomes to sum over)
- can have g(x) that is not analytically tractable/nice for continuous integration

<u>Example</u>: $g(x) = \frac{e^{ax+b}}{1+e^{ax+b}}$, a & b some constants [logistic fxn]

in Machine learning apps: need to find $E_x[g(x)]$ for normalizing constant : $E[g(x)] = \int_{-\infty}^{\infty} \frac{e^{ax+b}}{1+e^{ax+b}} \cdot p(x)\,dx \rightarrow$ generally <u>NOT</u> closed form for typical pdfs p(x)!

Numerical Simulations/Approximations of E[.]

<u>Discrete Case</u>

$$E[g(x)] = \sum_{i=1}^{N_x} g(x=i)P(x=i)$$

<u>Continuous Case</u>

$$E[g(x)] = \int_{-\infty}^{\infty} g(x)p(x)dx$$

Recall: E[.] = shorthand for "take sample average of g(x) for infinite number of samples"

→ but if we instead had access to <u>finite</u> samples $x_1,x_2,\ldots,x_N$ ~ P(x) or p(x), could we still <u>approximate</u> E[g(x)] by a (finite) sample average of g(x) for a "big enough" finite N?

- <u>Yes we can! According to the (Weak) Law of Large Numbers:</u>

Let $x_1, x_2, \ldots x_N$ be II & identically distributed (iid) random variables such that $E[x_i] = \mu$ & $\mathrm{var}(x_i) = \sigma^2$ $\forall\, i = 1, \ldots, N$. Then we have for any $\varepsilon > 0$

Follows from Chebyshov Inequality

$$\mathbb{P}\left(\left| \underbrace{\frac{x_1 + x_2 + \cdots x_N}{N}}_{\bar{x}_{sample}\ \text{[sample average)}} - \mu \right| \geq \varepsilon \right) \leq \frac{\sigma^2}{N\varepsilon^2}$$

→ in particular, as $N \to \infty$, RHS → 0

Monte Carlo Approximations

- General idea: given function g(x) and set of i.i.d. samples from pmf P(X) or pdf p(x),

$$E_x[g(x)] = \begin{cases} \int_{-\infty}^{\infty} g(x)\,p(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} g(x_i) & \text{[cont.]} \\ \sum_{i=1}^{N_x} g(x=i)\,P(x=i) \approx \frac{1}{N}\sum_{i=1}^{N} g(x_i) & \end{cases}$$

where $x_i \underset{iid}{\sim} p(x)\ \forall i=1,\dots,N$

where $x_i \underset{iid}{\sim} P(x)\ \forall i=1,\dots,N$

$\hat{E}[g(x)]$: Monte Carlo approximations to $E_x[g(x)]$

Law of large Numbers says that $\hat{E}[g(x)] \xrightarrow[N\to\infty]{} E_x[g(x)]$

But How to Get Monte Carlo Samples in the First Place?

- Need sufficiently large number N of i.i.d. sample outcomes from P(x) / p(x)
- That is, need ability to get arbitrarily large number N of "random experiments" to simulate g(x) outcomes according to specified "target distribution"
- Ideally: "direct sampling" Monte Carlo: simulate outcomes directly from target distribution P(x) or p(x), if easy to do [e.g. nice/"standard" distributions]
- "Indirect sampling" Monte Carlo: set up another "proposal distribution" Q(x) that's easy to sample from, and compare samples point-wise against P(x) / p(x) [useful when P(x) / p(x) is highly complex/non-standard or not closed-form]
  - Markov chain Monte Carlo (MCMC)
  - Importance Sampling

Sampling from a Discrete Probability Table

- Finite discrete distributions often represented/encoded as probability tables
- Sometimes also called "multinomial distributions"
- Example: Discrete RV x w/ $N_x = 4$ possible outcomes

x outcomes j	$j=1$	$j=2$	$j=3$	$j=4$
$P(x=j)$	0.4	0.2	0.1	0.3

→ adds to 1 ✓

possible sample outcomes $\{x_1, \ldots, x_N\}$ for $N=6$ iid samples:

$\{1, 1, 1, 3, 4, 2\}$, $\{2, 3, 1, 1, 4, 2\}$, $\{2, 2, 3, 1, 1, 1\}$ ...

- How to simulate N i.i.d. discrete random samples from such a probability distribution for any $N_x >= 2$?

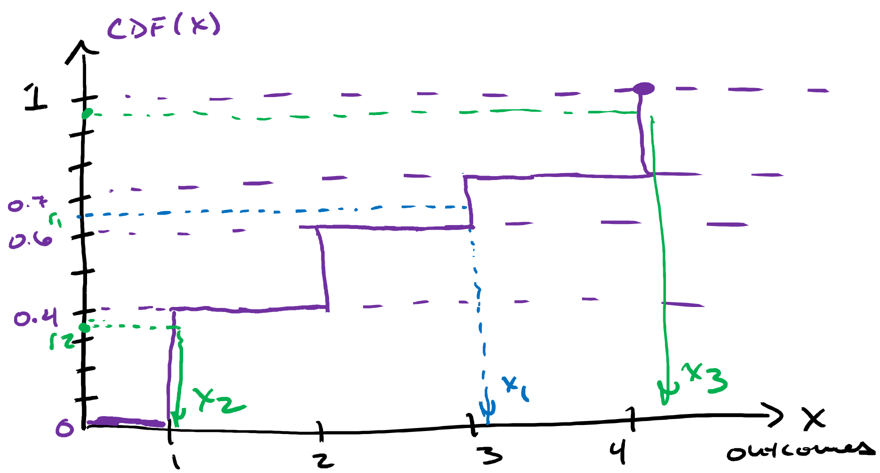
Sampling from a Discrete Probability Table

All we need is a uniform random number generator, and the cdf of the distribution!

Key idea: “go backwards” to generate a sample value from CDF(x):

- pick a probability value **r** between 0 and 1 uniformly at random (“rand” in Matlab)
- then use cdf to find x that has smallest CDF(x) such that **r** < CDF(x)

Example: cdf from previous prob. table:



ex. Sampling run : $N = 3$

Sample #1 : $r_1 \sim \mathcal{U}[0,1] \rightarrow r_1 = 0.65$ (r_1 = rand)

$\Rightarrow 0.65 > CDF(1) = 0.4$

$0.65 > CDF(2) = 0.6$

$0.65 < CDF(3) = 0.7 \rightarrow \boxed{x_1 = 3}$

Sample #2 : $r_2 \sim \mathcal{U}[0,1] \rightarrow r_2 = 0.31$

$\Rightarrow 0.31 < CDF(1) = 0.4 \rightarrow \boxed{x_2 = 1}$

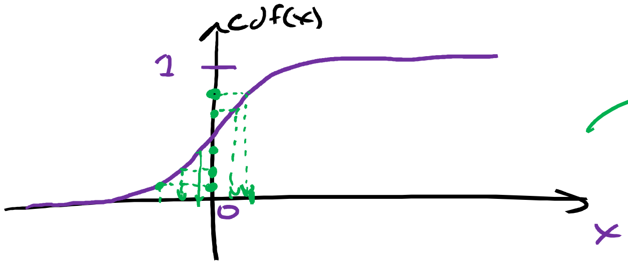
Sample #3 : $r_3 = 0.95 \; (\leftarrow \mathcal{U}[0,1]) \rightarrow \boxed{x_3 = 4}$

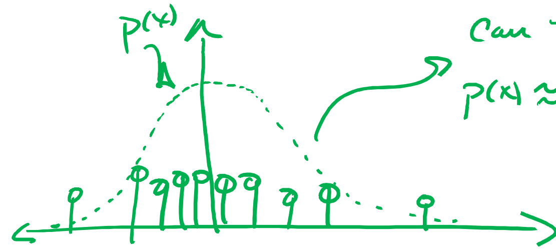
Basic Matlab Code for Sampling Discrete Probability Table

```
function [sampout] = discrete_sample(probTable)
%This produces a single sample sampout from a Nx by 1 discrete prob table (input:
probTable); sampout is an integer between 1 and Nx (Nx is # of possible discrete outcomes)

%set up discrete CDF: probTable input is size Nx by 1 array that sums to 1
%% Nx = size(probTable,1);
discrete_CDF = cumsum(probTable,1); %take cumulative sum along rows

%draw a random # from U(0,1):
r = rand;

%compare r to CDF:
tag = (r < discrete_CDF); %returns a logical Nx by 1 array
tag = single(tag); %convert to a numeric array
tagsums = cumsum(tag,1); %tally up # of times r is less than CDF

%extract the row where the first r<P(x) occurs in tag
[sampout,~] = find(tagsums==1);
```

Sampling from a "Nice" Continuous PDF

- Same idea can be extended to sampling pdfs, if corresponding cdf is available





Can think of
$$p(x) \approx \frac{1}{N}\sum_{i=1}^{N} \delta(x - x_i)$$
Dirac Deltas

But not necessarily most efficient way to draw continuous samples

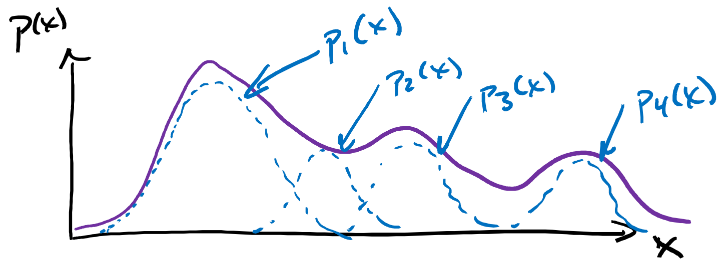
- Limited by cdf precision, cdf not always available, hard for multiple dimensions…

Standard/"nice" pdfs (Gaussian, exponential, gamma, etc.) have more efficient sampling methods that come built-in with most computing environments

- e.g. "randn" for normal/Gaussian distributions

"Mixture Model" Continuous PDF

- In many cases, can reasonably model/closely approximate complicated pdf p(x) as a weighted combination of nice/standard pdfs → **mixture model pdf**



$$\longrightarrow p(x) = \sum_{m=1}^{M} w_m \cdot p_m(x)$$

where $w_m \geq 0$ & $\sum_{m=1}^{M} w_m = 1$

& $\underbrace{\text{each } p_m(x) \text{ is a valid}}$ pdf [mixture components or "mixands"]

ie $\int_{-\infty}^{\infty} p_m(x)\,dx = 1$

⇒ Note: $\int_{-\infty}^{\infty} p(x)\,dx = \int_{-\infty}^{\infty} \sum_{m=1}^{M} w_m\, p_m(x)\,dx$

$$= \sum_{m=1}^{M} w_m \underbrace{\left[\int_{-\infty}^{\infty} p_m(x)\,dx\right]}_{=1} = \sum_{m=1}^{M} w_m \overset{?}{=} 1 \checkmark$$

Sampling "Mixture Model" Continuous PDFs

- Very easy to draw samples from mixture model pdfs, if we already know how to draw from discrete probability table and "nice"/standard component pdfs
- Key idea: two steps to getting continuous sample realization x:
- **Step 1:** sample a discrete component index **c** at random according to weights $w_1,\ldots,w_M$ using a discrete probability table (where c is one of 1,…,M)

i.e. $c \sim$

$m=1$	$m=2$	$\ldots$	$m=\underline{M}$
w_1	w_2	$\ldots$	$w_{\underline{M}}$

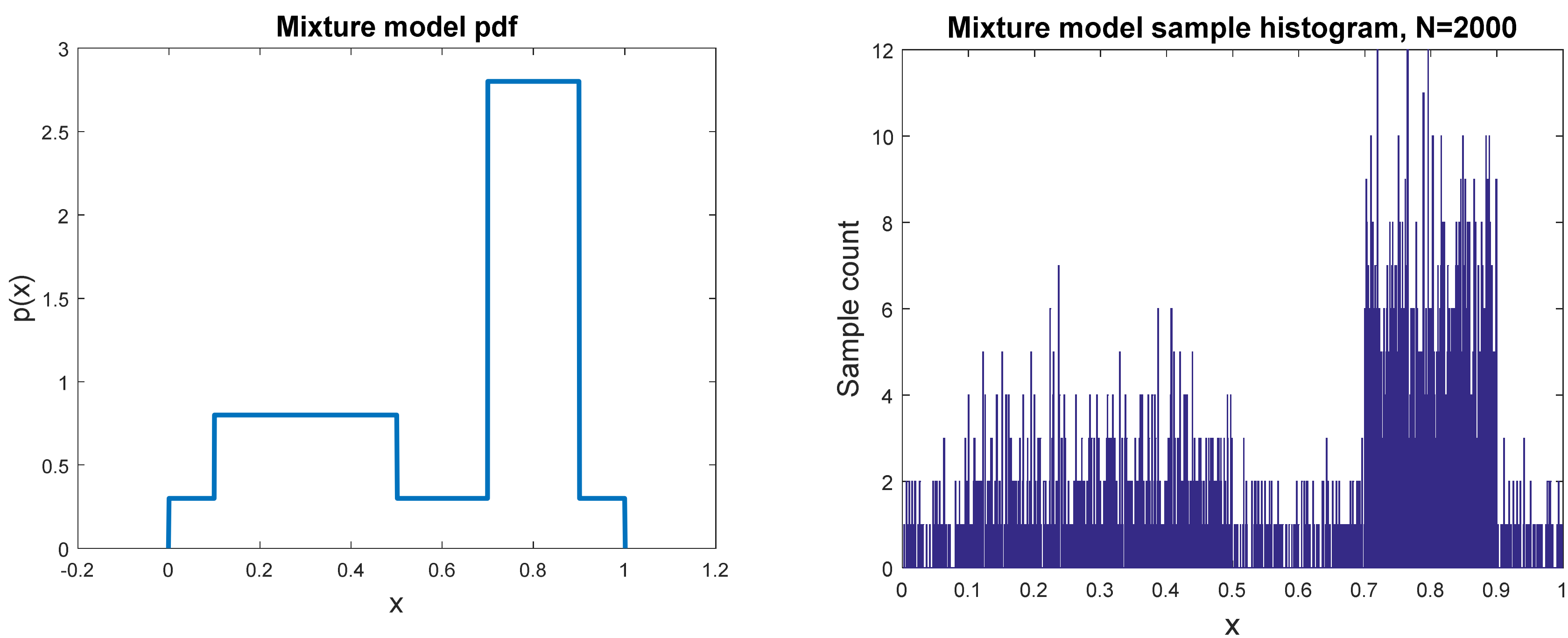
- **Step 2:** draw sample x from the component pdf $p_{\underline{m=\mathbf{c}}}(x)$ corresponding to the sample index **c** produced from Step 1

i.e. if $c=1$, then $x \sim p_1(x)$
$c=2$, " $x \sim p_2(x)$
$\vdots$
$c=M$, then $x \sim p_M(x)$

Example E[.] calculations using mixture model

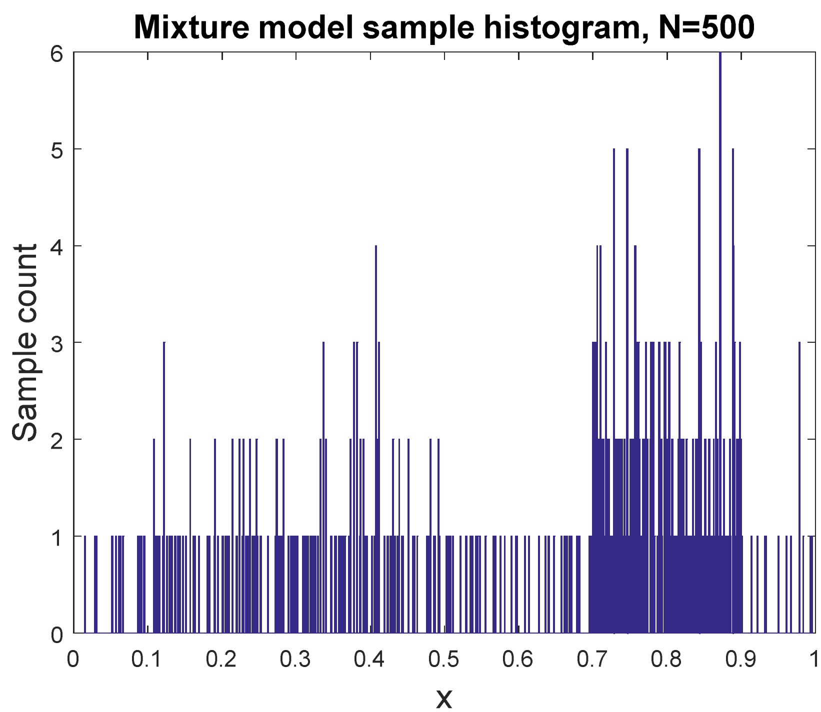
- Three component mixture of uniform pdfs

$$p(x) = 0.3 \cdot \mathcal{U}[0,1] + 0.2 \cdot \mathcal{U}[0.1, 0.5] + 0.5 \cdot \mathcal{U}[0.7, 0.9]$$



Example E[.] calculations using mixture model

- Compute mean and variance of x
- Monte Carlo vs. exact/grid-based values:



Exact grid (1000 grid points):

$E[x]$

μ = 0.6123 $\rightarrow \mu_G = \sum_{i=1}^{\#grid} x_i \cdot p(x_i) \cdot \Delta x$

length of grid

σ^2 = 0.0693 → σ = 0.2632

$E[(x-\bar{x})^2]$

$\sigma_G^2 = \sum_{i=1}^{\#grid} (x_i - \mu_G)^2 \cdot p(x_i) \cdot \Delta x$

Monte Carlo (500 samples):

$\hat{\mu}$ = 0.6073 $\rightarrow \frac{1}{N} \sum_{i=1}^{N} x_i$

$\hat{\sigma}^2$ = 0.0670 → σ = 0.2589 $\rightarrow \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})^2$