Intro to Sampling Methods

CSE586 Computer Vision II
Spring 2010, Penn State Univ
A Brief Overview of Sampling

Monte Carlo Integration

Sampling and Expected Values

Inverse Transform Sampling (CDF)

Rejection Sampling

Importance Sampling
The idea

What is the probability that a dart thrown uniformly at random will hit the red area? 

http://videolectures.net/mlss08au_freitas_asm/
Integration and Area

The idea

What is the probability that a dart thrown uniformly at random will hit the red area?

http://videolectures.net/mlss08au_freitas_asm/
The idea

What is the probability that a dart thrown uniformly at random will hit the red area?

http://videolectures.net/mlss08au_freitas_asm/
Integration and Area

The idea

What is the probability that a dart thrown uniformly at random will hit the red area?

http://videolectures.net/mlss08au_freitas_asm/
Integration and Area

- As we use more samples, our answer should get more and more accurate
- Doesn’t matter what the shape looks like

arbitrary region
(even disconnected)

area under curve
aka integration!
Monte Carlo Integration

Goal: compute definite integral of function $f(x)$ from $a$ to $b$

Generate $N$ uniform random samples in upper bound volume

Answer $= \frac{K}{N} \times \text{Area of upper bound volume}$

count the $K$ samples that fall below the $f(x)$ curve
Monte Carlo Integration

Sampling-based integration is useful for computing the normalizing constant that turns an arbitrary non-negative function $f(x)$ into a probability density function $p(x)$.

$$Z = \int f(x) \, dx$$

Compute this via sampling (Monte Carlo Integration). Then:

$$P(x) = \frac{1}{Z} f(x)$$

Note: for complicated, multidimensional functions, this is the ONLY way we can compute this normalizing constant.
Sampling: A Motivation

If we can generate random samples $x_i$ from a given distribution $P(x)$, then we can estimate expected values of functions under this distribution by summation, rather than integration.

That is, we can approximate:

$$E(f(x)) = \int f(x)P(x)dx$$

by first generating $N$ i.i.d. samples from $P(x)$ and then forming the empirical estimate:

$$\hat{E}(f(x)) = \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$
Inverse Transform Sampling

It is easy to sample from a discrete 1D distribution, using the cumulative distribution function.

\[ c(k) = \frac{\sum_{i=1}^{k} w_i}{\sum_{i=1}^{N} w_i} \]

The cumulative distribution function is

\[ F(x) = P(X \leq x) \]
Inverse Transform Sampling

It is easy to sample from a discrete 1D distribution, using the cumulative distribution function.

1) Generate uniform $u$ in the range $[0,1]$
2) Visualize a horizontal line intersecting bars
3) If index of intersected bar is $j$, output new sample $x_k = j$
Inverse Transform Sampling

Why it works:

cumulative distribution function

\[ F(x) = P(X \leq x) \]

inverse cumulative distribution function

\[ F^{-1}(t) = \min\{x : F(x) = t, 0 < t < 1\} \]

Claim: if \( U \) is a uniform random variable on (0,1) then \( X = F^{-1}(U) \) has distribution function \( F \).

Proof:

\[
P(F^{-1}(U) \leq x) \\
= P(\min\{x : F(x) = U\} \leq x) \quad \text{(def of } F^{-1}) \\
= P(U \leq F(x)) \quad \text{(applied } F \text{ to both sides)} \\
= F(x) \quad \text{(def of distribution function of } U) 
\]
Efficient Generating Many Samples

Basic idea: choose one initial small random number; deterministically sample the rest by “crawling” up the cdf function. This is $O(N)$.

Odd property: you generate the “random” numbers in sorted order...

Due to Arulampalam

Algorithm 2: Resampling Algorithm

\[
\{x_{k}^{i*}, w_{k}^{i}, i^{j}\}_{j=1}^{N_{s}} = \text{RESAMPLE} \left[\{x_{k}^{i}, w_{k}^{i}\}_{i=1}^{N_{s}}\right]
\]

- Initialize the CDF: $c_1 = 0$
- FOR $i = 2: N_s$
  - Construct CDF: $c_i = c_{i-1} + w_k^i$
  - END FOR
- Start at the bottom of the CDF: $i = 1$
- Draw a starting point: $u_1 \sim \mathcal{U}[0, N_s^{-1}]$
- FOR $j = 1: N_s$
  - Move along the CDF: $u_j = u_1 + N_s^{-1}(j-1)$
  - WHILE $u_j \leq c_i$
    - $i = i + 1$
    - END WHILE
  - Assign sample: $x_{k}^{j*} = x_{k}^{i}$
  - Assign weight: $w_{k}^{j} = N_s^{-1}$
  - Assign parent: $i^j = i$
- END FOR
A Brief Overview of Sampling

Inverse Transform Sampling (CDF)

Rejection Sampling

Importance Sampling

For these two, we can sample from continuous distributions, and they do not even need to be normalized.

That is, to sample from distribution $P$, we only need to know a function $P^*$, where $P = P^* / c$, for some normalization constant $c$. 
Rejection Sampling

Need a proposal density $Q(x)$ [e.g. uniform or Gaussian], and a constant $c$ such that $c(Qx)$ is an upper bound for $P^*(x)$

Example with $Q(x)$ uniform

- $cQ(x)$
- $P^*(x)$
- Upper bound
- Generate uniform random samples in upper bound volume
- Accept samples that fall below the $P^*(x)$ curve

The marginal density of the $x$ coordinates of the points is then proportional to $P^*(x)$

Note the relationship to Monte Carlo integration.
Rejection Sampling

More generally:
1) generate sample $x_i$ from a proposal density $Q(x)$
2) generate sample $u$ from uniform $[0, cQ(x_i)]$
3) if $u \leq P^*(x_i)$ accept $x_i$; else reject
Importance “Sampling”

Not for generating samples. It is a method to estimate the expected value of a function $f(x_i)$ directly

1) Generate $x_i$ from $Q(x)$

2) an empirical estimate of $E_Q(f(x))$, the expected value of $f(x)$ under distribution $Q(x)$, is then

$$\hat{E}_Q(f(x)) = \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

3) However, we want $E_P(f(x))$, which is the expected value of $f(x)$ under distribution $P(x) = P^*(x)/Z$
When we generate from $Q(x)$, values of $x$ where $Q(x)$ is greater than $P^*(x)$ are overrepresented, and values where $Q(x)$ is less than $P^*(x)$ are underrepresented.

To mitigate this effect, introduce a weighting term

$$w_i = \frac{P^*(x_i)}{Q(x_i)}$$
New procedure to estimate $E_P(f(x))$:

1) Generate $N$ samples $x_i$ from $Q(x)$

2) form importance weights

$$w_i = \frac{P^*(x_i)}{Q(x_i)}$$

3) compute empirical estimate of $E_P(f(x))$, the expected value of $f(x)$ under distribution $P(x)$, as

$$\hat{E}_P(f(x)) = \frac{\sum w_i f(x_i)}{\sum w_i}$$
Resampling

Note: We thus have a set of weighted samples \((x_i, w_i \mid i=1,\ldots,N)\)

If we really need random samples from \(P\), we can generate them by resampling such that the likelihood of choosing value \(x_i\) is proportional to its weight \(w_i\).

This would now involve now sampling from a discrete distribution of \(N\) possible values (the \(N\) values of \(x_i\)).

Therefore, regardless of the dimensionality of vector \(x\), we are resampling from a 1D distribution (we are essentially sampling from the indices 1...\(N\), in proportion to the importance weights \(w_i\)). So we can using the inverse transform sampling method we discussed earlier.
Note on Proposal Functions

Computational efficiency is best if the proposal distribution looks a lot like the desired distribution (area between curves is small).

These methods can fail badly when the proposal distribution has 0 density in a region where the desired distribution has non-negligeable density.

For this last reason, it is said that the proposal distribution should have heavy tails.
Sequential Monte Carlo Methods

Sequential Importance Sampling (SIS) and the closely related algorithm Sampling Importance Sampling (SIR) are known by various names in the literature:

- bootstrap filtering
- particle filtering
- Condensation algorithm
- survival of the fittest

General idea: Importance sampling on time series data, with samples and weights updated as each new data term is observed. Well-suited for simulating Markov chains and HMMs!
Markov-Chain Monte Carlo
References

Markov Chain Monte Carlo for Computer Vision

A tutorial at the 10th Intl Conf. on Computer Vision
October, 2005, Beijing

by
Song-Chun Zhu, UCLA
Frank Dellaert, Gatech
Zhuowen Tu, UCLA

INTRODUCTION TO MONTE CARLO METHODS
D.J.C. MACKAY
Department of Physics, Cambridge University.
Cavendish Laboratory, Madingley Road,
Cambridge, CB3 0HE. United Kingdom.

An Introduction to MCMC for Machine Learning
CHRISTOPHE ANDRIEU
C.Andrieu@bristol.ac.uk
Department of Mathematics, Statistics Group, University of Bristol, University Walk, Bristol BS8 1TW, UK

NANDO DE FREITAS
nando@cs.ubc.ca
Department of Computer Science, University of British Columbia, 2566 Main Mall, Vancouver, BC V6T 1Z4, Canada

ARNAUD DOUCET
doucet@cse.unsw.edu.au
Department of Electrical and Electronic Engineering, University of Melbourne, Parkville, Victoria 3052, Australia

MICHAEL I. JORDAN
mijordan@cs.berkeley.edu
Departments of Computer Science and Statistics, University of California at Berkeley, 387 Soda Hall, Berkeley, CA 94720-1772, USA
Problem

Sampling in High-dimensional Spaces

Standard methods fail:

• Rejection Sampling
  – Rejection rate increase with $N \to 100\%$

• Importance Sampling
  – Same problem: vast majority weights $\to 0$

Intuition: In high dimension problems, the “Typical Set” (volume of nonnegligible prob in state space) is a small fraction of the total space.
High Dimensional Spaces

Segmentation Example

• Binary Segmentation of image

each pixel has two states: on and off
Probability of a Segmentation

- Very high-dimensional
- $256 \times 256$ pixels $= 65536$ pixels
- Dimension of state space $N = 65536$ !!!!

- $\#$ binary segmentations $= \text{finite}$, but...
- $2^{65536} = 2 \times 10^{19728} \gg 10^{79} = \text{atoms in universe}$
Robert Collins
CSE

Representation $P(\text{Segmentation})$

- Histogram? No!
- Assume pixels independent?
  $P(x_1x_2x_3...)=P(x_1)P(x_2)P(x_3)...$
  ignores neighborhood structure of pixel lattice
  and empirical evidence
  that images are “smooth”

- Approximate solution: samples !!!
Brilliant Idea!

- Published June 1953
- Top 10 algorithm!
- Set up a Markov chain
- Run the chain until stationary
- All subsequent samples are from stationary distribution
Recall: Markov Chains

Markov Chain:

- A sequence of random variables $X_1, X_2, X_3, ...$
- Each variable is a distribution over a set of states (a, b, c...)
- Transition probability of going to next state only depends on the current state. e.g. $P(X_{n+1} = a \mid X_n = b)$

transition probs can be arranged in an NxN table of elements

$k_{ij} = P(X_{n+1} = j \mid X_n = i)$

where the rows sum to one
K = transpose of transition prob table \( \{ k_{ij} \} \) (cols sum to one. We do this for computational convenience (next slide).
Question:

Assume you start in some state, and then run the simulation for a large number of time steps. What percentage of time do you spend at $X_1$, $X_2$ and $X_3$?
four possible initial distributions

*: \[ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

initial distribution

distribution after one time step

\[
q_0 = K q_0 \\
q_1 = K q_1 = K^2 q_0 \\
q_2 = K q_2 = K^2 q_1 = K^3 q_0 \\
q_3 = K q_3 = \ldots K^{10} q_0 \\
q_{10} = K q_9 = \ldots K^{10} q_0
\]

all eventually end up with same distribution -- this is the stationary distribution!
Eigen-analysis

\[ K = \begin{array}{ccc}
0.1000 & 0.5000 & 0.6000 \\
0.6000 & 0.2000 & 0.3000 \\
0.3000 & 0.3000 & 0.1000 \\
\end{array} \]

\[ KE = ED \]

in matlab:
\[ [E,D] = \text{eigs}(K) \]

Eigenvalue \( v_1 \) always 1

Stationary distribution
\( \pi = e_1 / \text{sum}(e_1) \)
i.e. \( K \pi = \pi \)
The PageRank of a webpage as used by Google is defined by a Markov chain. It is the probability to be at page $i$ in the stationary distribution on the following Markov chain on all (known) webpages. If $N$ is the number of known webpages, and a page $i$ has $k_i$ links then it has transition probability $(1-q)/k_i + q/N$ for all pages that are linked to and $q/N$ for all pages that are not linked to. The parameter $q$ is taken to be about 0.15.
Google Pagerank

Pagerank == First Eigenvector of the Web Graph!

Computation assumes a 15% “random restart” probability

Sergey Brin and Lawrence Page, The anatomy of a large-scale hypertextual {Web} search engine, Computer Networks and ISDN Systems, 1998
Another Question:

Assume you want to spend a particular percentage of time at X1, X2 and X3. What should the transition probabilities be?

\[ P(x1) = 0.2 \]
\[ P(x2) = 0.3 \]
\[ P(x3) = 0.5 \]


we will discuss this next time