Lecture 4: Approximate Inference

Wray Buntine and Petri Myllymäki

Précis

- Approximations needed for scale and/or speed.
- Basic methods built on general approximation techniques from statistics.
- We compare some of these with counterparts in optimization.

Overview

- **Approximations:** some background.
- Gibbs sampling.
- Importance and Rejection sampling.

Approximations: Motivation

- General maximisation on the Almond tree is essentially an "all solutions" approach. What if we want just one solution?
- What if our graph is too big for Almond trees? Can an approximation be developed?
- What if a fast algorithm is needed inside a real-time system?

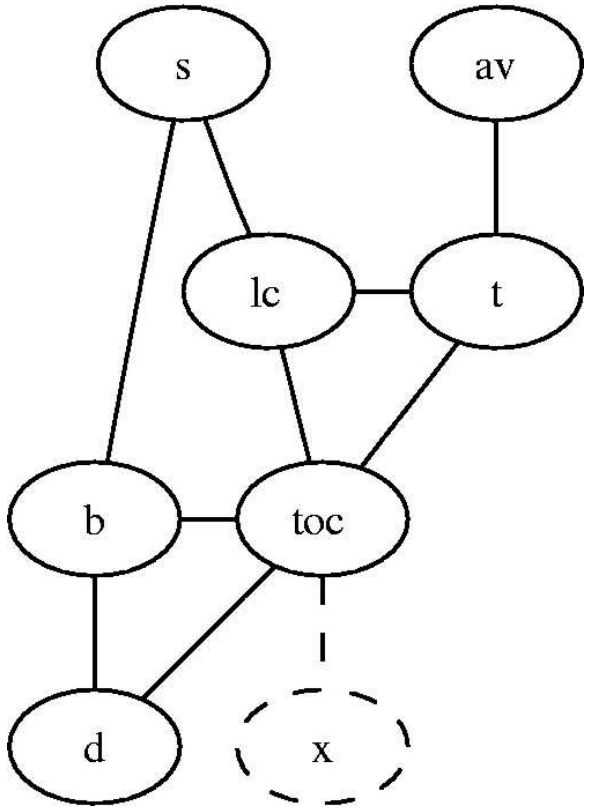
Local Search, example

Problem. Given a value for x, find an assignment to the remaining variables the maximises the probability $p(toc,d,b,t,lc,s,av|x)$.

Approximate solution.

1. Make an initial assignment to free variables.
2. Reassign toc by maximising $p(toc|d,b,t,lc,s,av,x)$.
3. Reassign d by maximising $p(d|toc,b,t,lc,s,av,x)$.
4. Reassign b by maximising $p(b|toc,d,t,lc,s,av,x)$.
5. ...

Continue with other variables, and repeat reassigning until bored or no more changes occur.



Local Search, background

- The initial algorithm tried for many search problems.
- Works pretty well on smaller problems, and some so-called random problems.
- For instance, 3-colouring, 3SAT, graph partitioning,

Local Search: Graph Colouring

Given an undirected graph G on variables X and C colours, find an assignment of colours to nodes so that no two neigbours have the same colour.

Note another version of the problem is *chromatic number*, which is to find the minimum C for which the graph is colourable, as above.

Create a probability distribution via

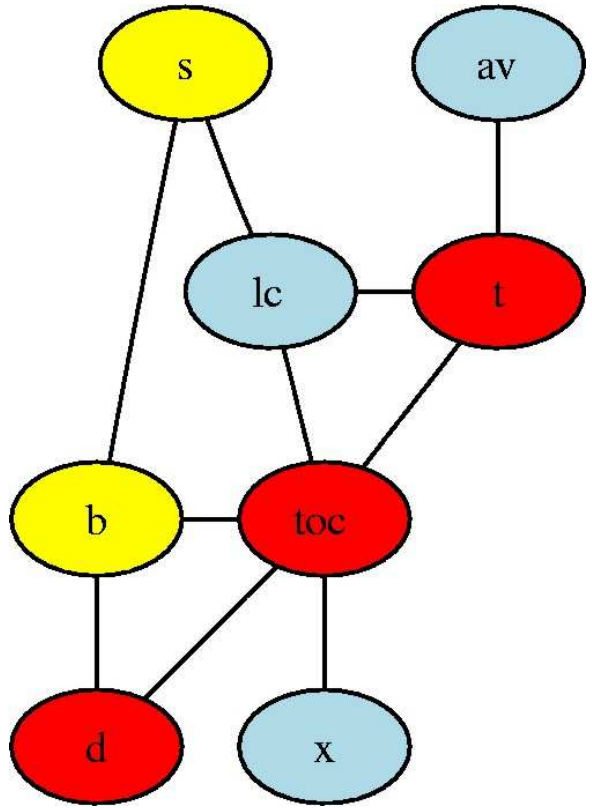
$$p(X) \;\propto\; e^{-T \sum_{(x,y)\in G} 1_{\text{colour}(x)=\text{colour}(y)}}$$

where $T \geq 0$ by physical analogy is referred to as temperature.

- $T = 0$ then everything equally likely.
- $T = 10^{20}$ then all solutions equally likely, and everything else effectively impossible.
- To find maximal probability assignment, all that matters is $T > 0$.

Local Search, example

node	cnflts	toc yellow	b blue
x	0	0	0
toc	2	1	0
d	1	0	0
b	1	2	0
t	1	0	0
lc	0	0	0
av	0	0	0
s	1	1	0



$$\text{conflicts}(x) = \sum_{y:(x,y)\in G} 1_{\text{colour}(x)=\text{colour}(y)}$$

Graph Colouring Algorithms

Two algorithms, the second is an improvement in the search, using a heuristic to select k at each step. Each assume some initialisation X_0 (itself a hard and important algorithm).

Repeat until bored.

1. *Choose* dimension k randomly or by cycling through.
2. *Reassign* x_k to a colour such that conflicts(x_k) becomes a minimum.

Repeat until bored.

1. *Choose* dimension k such that Δconflicts(x_k) is a maximum.
2. *Reassign* x_k to a colour such that conflicts(x_k) becomes a minimum.

Local Search Algorithm

Used to search for a maximal probability assignment for a distribution, $\text{argmax}_X p(X)$.

Input: $X = X_0$, conditional distributions
Repeat until bored.

1. *Choose* a dimension k.
2. *Reassign* x_k, breaking ties arbitrarily.

$$x_k \leftarrow \text{argmax}_{x_k} p(x_k \,|\, X/\{x_k\})$$

- Sometimes called coordinate-wise hill climbing.
- No guarantees that it will find a maximal. It will find a *local maximum*, but no good guarantees about time.
- May run many times from different random restarts and keep the best.
- Reassigning x_k even when the probability is unchanged is called *ridge running* and is a critical part of the search process.

Local Search, example

The Grand Canyon (from Wikipedia.org).

Local Search Algorithm with Dimension Optimisation

Input: $X = X_0$, conditional distributions.
Initiatise heap of values

$$\text{score}(k) = \max_{v \in \text{dom}(x_k)} p\left(x_k = v \,|\, X/\{x_k\}\right) \Big/ p(x_k \,|\, X/\{x_k\})$$

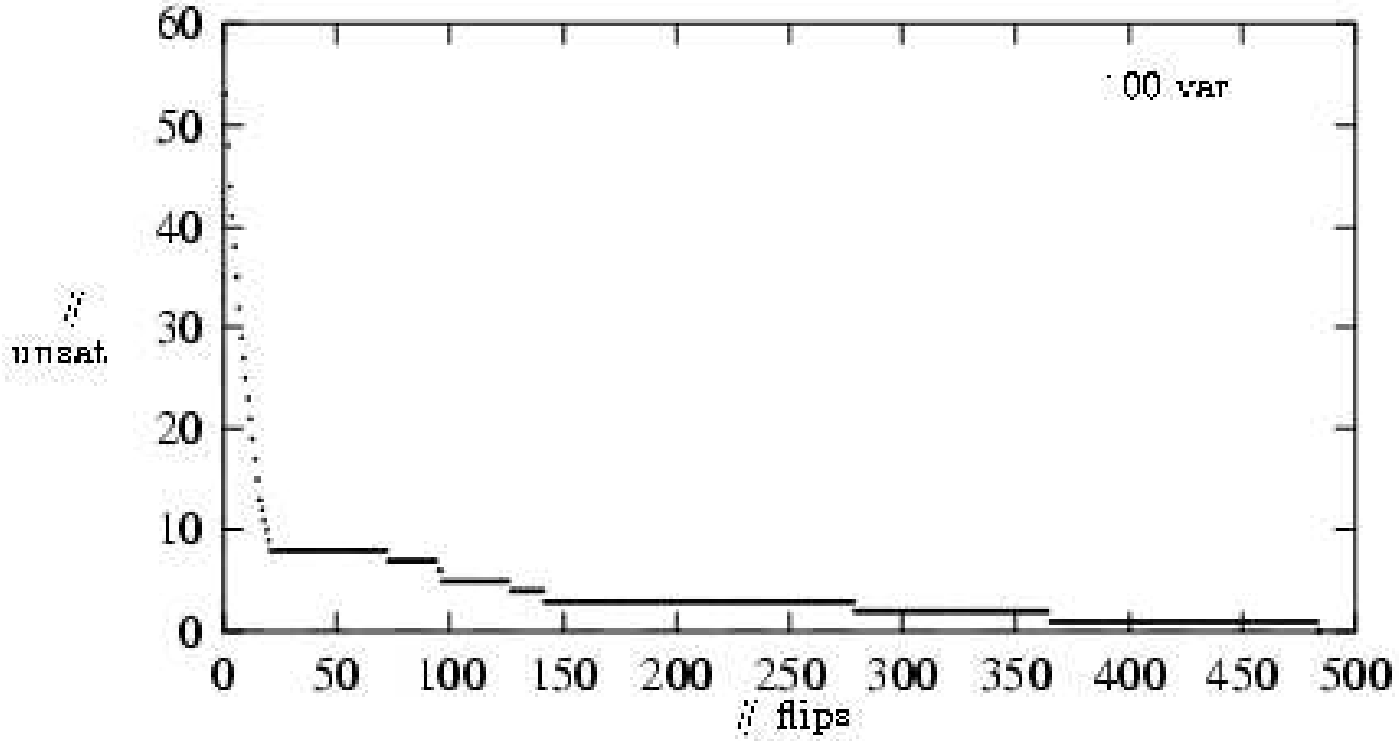
Repeat until bored.

1. *Choose* dimension k from the top of the heap.
2. *Reassign* x_k, breaking ties arbitrarily.

$$x_k \leftarrow \text{argmax}_{x_k} p(x_k \,|\, X/\{x_k\})$$

3. *Update heap* for all dimensions l such that $x_l \in \text{nbrs}(x_k)$.

Local Search, example

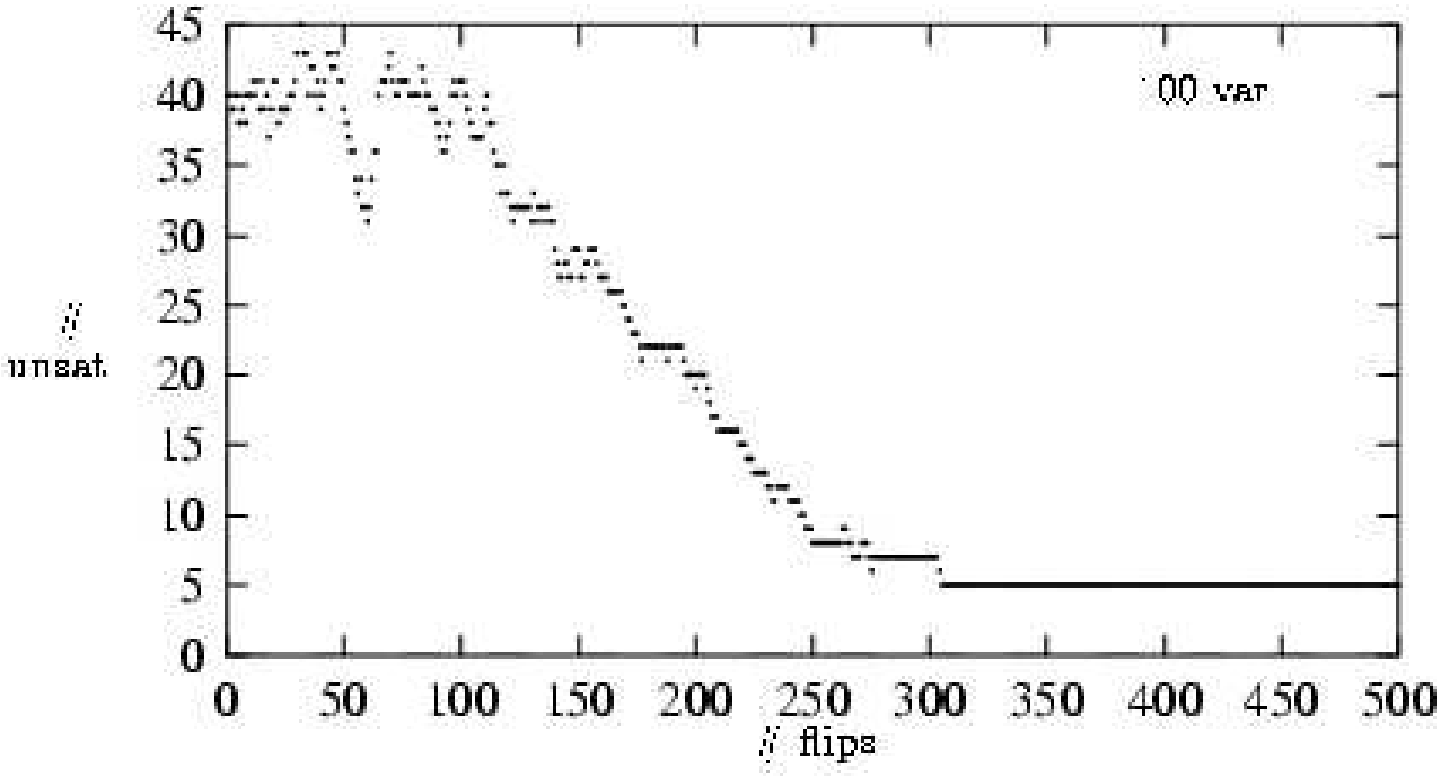


Here we have a 100 variable 3SAT problem done in local search.

Overview

- Approximations.
- **Gibbs sampling:** statistical counterpart to local search.
- Importance and Rejection sampling.

Gibbs sampling, example



Here we have a 100 variable 3SAT problem done with "Gibbs" search, with $T > 0$ in use. Now we sample variables at each stage instead of assigning to the value maximising local probability.

Gibbs Algorithm

Input: X_0, conditional distributions, function $f(X)$
Repeat until bored.

1. *Choose* dimension k (in a way that all dimensions will eventually get chosen in some fixed finite number C of cycles).
2. *Resample* x_k in X according to

$$x_k \sim p(x_k | X/\{x_k\})$$

3. *Update* the sample mean $\widehat{f(X)}$ of $f(X)$ from the new X.

Theorem. If $p(X) > 0$ for all X, then (1) the value of X from above algorithm converges in the limit to a sample from $p(X)$ and (2) the sample mean $\widehat{f(X)}$ converges in the limit to the true mean $\overline{f} = \mathsf{E}_{X \sim p(X)}(f(X))$.

Gibbs Sampling Theory, cont.

Proof outline for (1). Consider the stochastic matrix formed from the single cycle update, $p(X_{i+1}|X_i)$. This is a square matrix with side domain$(|X|)$, call it $\mathbf{Q}$. Its columns normalise to 1 since they represent the space of output values for X_{i+1}, even though for any cycle only one x_k will change. Since $p(X) > 0$ for all X, and all dimensions get updated at least once in C cycles, $\mathbf{Q}^C$ is positive for every entry, and is also a stochastic matrix. By the Stochastic Matrix Lemma (next), for large N, and any initial distribution $q(X)$ represented as a column vector $\vec{q}$, $\mathbf{Q}^{CN}\vec{q}$ converges to a unique value. Since $p(X)$ is a fixed point of $p(X_{i+1}|X_i)$. it follows that it is the unique value of convergence.

Stochastic Matrix Lemma

Definition. A *positive stochastic matrix* is a square matrix, all whose entries are positive, and whose columns all normalise to 1.

Lemma. Let $\mathbf{Q}$ be a positive stochastic matrix. Then for any non-zero column vector $\vec{q}$, $\lim_{N\to\infty}\mathbf{Q}^N\vec{q}$ exists and is given by a factor of the unique column vector $\vec{r}$ satisfying $\vec{r}=\mathbf{Q}\vec{r}$.

NB. Proof on the next page uses fairly rudimentary knowledge of eigensystems (spectral analysis of matrices). The standard proof for convergence of Gibbs sampling relies on *ergodicity and aperiodicity* and then refers to a more detailed result in Markov chain theory. For us, this is unnecessary complexity. Positive stochastic matrices have a simple bounding argument to yield the result.

Stochastic Matrix Lemma, cont

Proof Outline. The lemma is equivalent to: $\mathbf{Q}$ has a single eigenvector with eigenvalue 1 and all others have absolute value less than 1. It is sufficient to show this for left (row) eigenvectors and not right (column) eigenvectors.

Suppose $\vec{s}$ is a left eigenvector, so $\vec{s}\mathbf{Q} = \lambda\vec{s}$. Then taking the k-th entry,

$$|\lambda s_k| = |\lambda||s_k| = \left|\sum_j A_{j,k} s_j\right| \leq \sum_j A_{j,k}|s_j| \leq \left(\sum_j A_{j,k}\right) \max_j |s_j| = \max_j |s_j|$$

Let $k = \text{argmax}_j |s_j|$, then this implies $|\lambda| \max_j |s_j| \leq \max_j |s_j|$ and thus $|\lambda| \leq 1$. When $|\lambda| = 1$. this implies equality across the equations. Since $A_{j,k} > 0$, it follows that $|s_j| = \max_j |s_j|$ for all j, and all s_j have the same sign. Thus the only left eigenvector with eigenvalue 1 is the unit vector.

Gibbs Sampling Theory, cont.

Proof outline for (2). $\mathsf{E}_{p(X_1,\ldots X_N|X_0)}\left(\left|\frac{1}{N}\sum_{i=1}^{N} f(X_i) - \overline{f}\right|\right)$ can be bounded by

$$\mathsf{E}_{p(X_1,\ldots X_N|X_0)}\left(\left|\frac{1}{N}\sum_{i=1}^{N}\left(1-\frac{p(X_i)}{p(X_i|X_0)}\right)f(X_i)\right|\right) + \mathsf{E}_{p(X_1,\ldots X_N|X_0)}\left(\left|\frac{1}{N}\sum_{i=1}^{N}\frac{p(X_i)}{p(X_i|X_0)}f(X_i) - \overline{f}\right|\right)$$

If $|f(X)| < F$ for all X, then the first term can be bounded by

$$F\frac{1}{N}\sum_{i=1}^{N}\mathsf{E}_{p(X_1,\ldots X_N|X_0)}\left(\frac{|p(X_i|X_0) - p(X_i)|}{p(X_i|X_0)}\right) \;=\; F\frac{1}{N}\sum_{i=1}^{N} D_i$$

where $D_i = \sum_{v\in \mathrm{dom}(X_i)} |p(X_i = v|X_0) - p(X_i = v)|$. This approaches 0 for large N since $p(X_i|X_0)$ approaches $p(X_i)$ linearly, that is $|p(X_i|X_0) - p(X_i)| < A\lambda^i$ for some A and $\lambda < 1$. The second term in the original bound corresponds to an adaptive importance sampling estimate of $\overline{f}$ and thus approaches 0 for large N as well, though this is a bit harder to show.

Gibbs Algorithm Details

- In Gibbs, the only requirement is the ability to sample from individual conditional distributions along each dimension.
- Theory extends to mixed real and discrete domains using extensions of linear algegra beyond real vector spaces of finite dimension.
- Gibbs notoriously slow, like plain local search. The dimensional optimization of local search can also be used as long as it is used probabilistically so that each dimension is possible. *e.g.* like

$$p(\text{choose } k) = \frac{\text{score}(k)}{\sum_k \text{score}(k)}$$

- Algorithm usually has a *burn-in* when no recording is done, followed by a *recording* when samples are taken at intervals of C cycles.

Automating Gibbs

The software *Bayesian Inference Using Gibbs Sampling* (BUGS) automates Gibbs style analysis with the following elements:

- A *specification language* for graphical models over a suitable class of distributions.
- A *compiler* that compiles models into primitive statements for a Gibbs abstract machine (a runtime interpreter).
- A general purpose conditional sampler called *adaptive rejection sampling* that can handle all instantiations of the conditional distributions compiled by the previous step.
- A nice *interface* supporting statistical input, output, burnin, recording, and diagnostics.

Overview

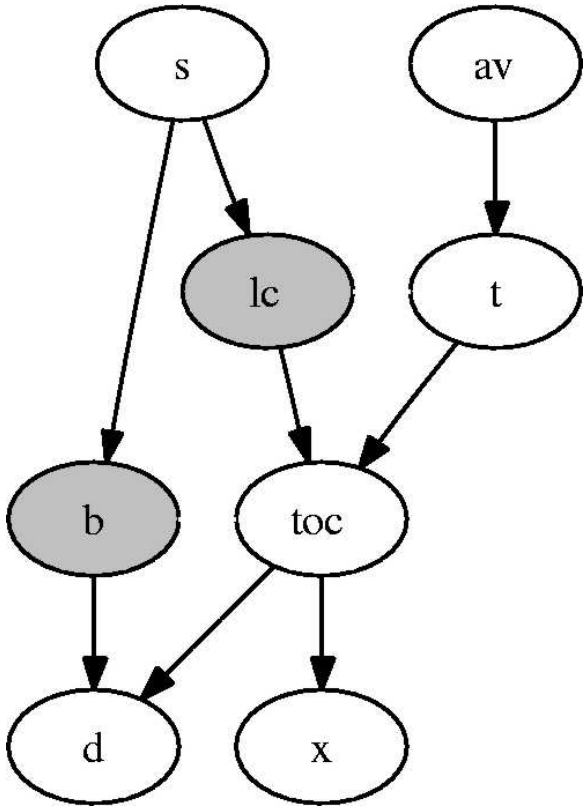
- Approximations.
- Gibbs sampling.
- **Importance and Rejection sampling:** general methods for sampling.

Rejection Sampling, example

Problem. Given values for lc, b, called *evidence*, sample assignments to the remaining variables.

Solution.

1. Sample in lexographic order, e.g., $s, av, t, b, lc, toc, d, x$, using the probability tables for the model, $p(x|\text{parents}(x))$.
2. Reject the sample if it disagrees with the evidence, so back to step 1.

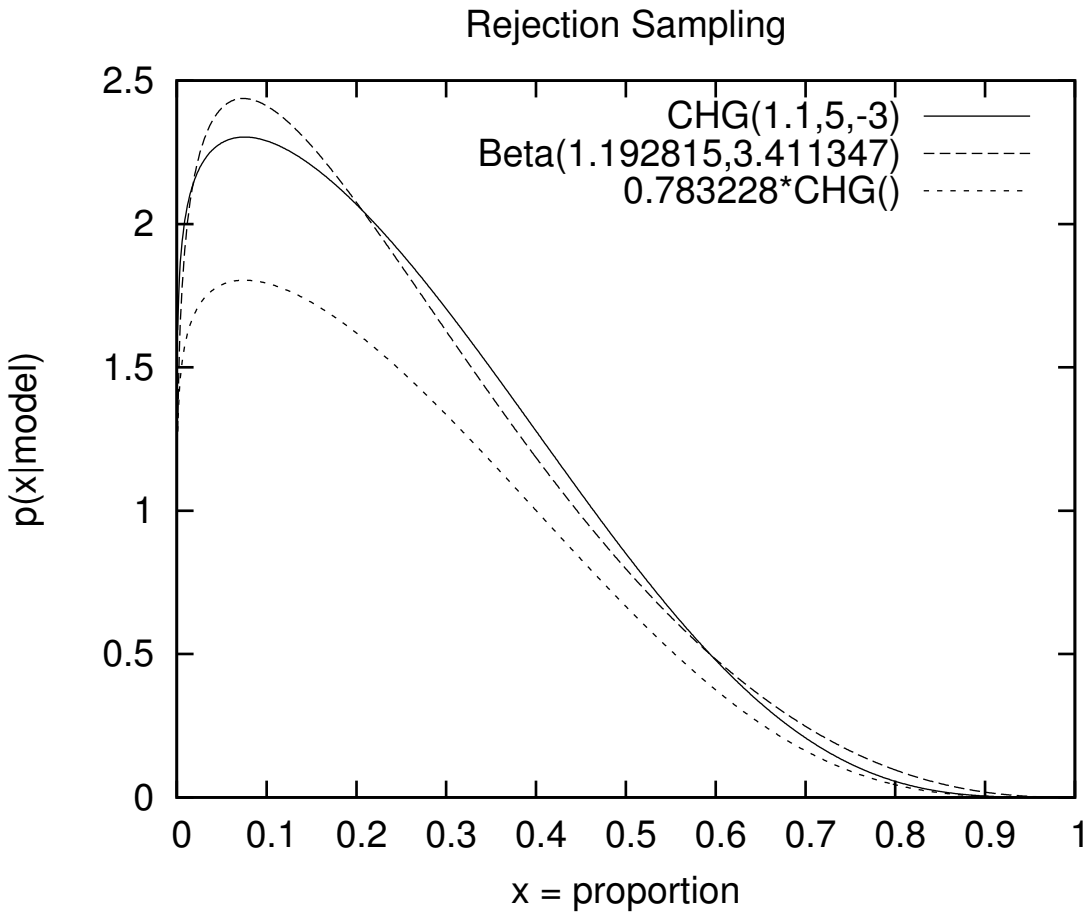


Rejection Sampling in 1-D

Want samples according to a Confluent Hypergeometric, CHG(1.1,5,-3).
So we first compute a similar Beta distribution, Beta(1.192815,3.411347).
Using rejection, shave off the top (1-0.783228) to reshape it back to a CHG(1.1,5,-3), but with only 0.783228 total probability.

1. $x \sim \text{Beta}(1.192815, 3.411347)$
2. Accept with probability

$$\frac{0.783228 * p(x|\text{CHG}(1.1,5,\text{-}3))}{p(x|\text{Beta}(1.192815,3.411347))}$$



Rejection Sampling of $p(X)$

Pick a *proposal distribution* $q(X)$ such that $q(X) > 0$ if $p(X) > 0$. Precompute $A = \min_{X:p(X)>0} \frac{q(X)}{p(X)}$.

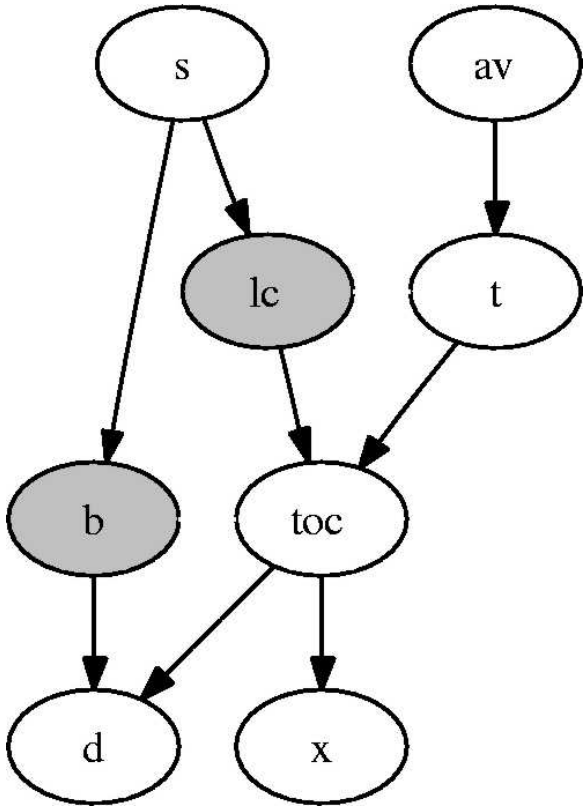
Input: $p(X)$, $q(X)$ and a sampler for it, A.
Repeat sampling.
1. Compute sample $X \sim q(X)$.
2. Compute sample $U \sim \text{Uniform}(0,1)$.

Until $Uq(X) < Ap(X)$.
Return: X

- Probability of acceptance in each loop is $\mathsf{E}_{X\sim q(X)}(\frac{Ap(X)}{q(X)})$ which is given by A.
- Thus expected number of cycles to first acceptance is $\frac{1}{A}$.
- Usually inefficient in large dimensions because then $A \ll 1$.

Importance Sampling, example

Problem. Given values for lc, b, called *evidence*, sample assignments to the remaining variables.



Solution.

1. Sample all but the evidence in lexographic order, e.g., s, av, t, toc, d, x, using the probability tables for the model, $p(x|\text{parents}(x))$.
2. Add a weight to the sample according to likelihood of the evidence:

$$w = p(lc|s)p(b|s)$$

NB. weights not normalised properly so means $\overline{f(X)}$ now computed via $\frac{\sum_i w_i f(X_i)}{\sum_i w_i}$ to normalise.

Importance Sampling of $p(X)$

Pick a proposal distribution $q(X)$ such that $q(X) > 0$ whenever $p(X) > 0$.

Input: unnormalised $p(X)$, $q(X)$ and a sampler for it.
Compute sample $X \sim q(X)$, and weight $w = \frac{p(X)}{q(X)}$.
Return: X, w.

- Every sample is used, but they are weighted differently.
- Usually inefficient in large dimensions as well, somewhat like rejection sampling.
- If estimating $\overline{f(X)}$, and normalisation for $p(X)$ unknown, then use $\widehat{f(X)} = \frac{\sum_i w_i f(X_i)}{\sum_i w_i}$. If $p(X)$ known, leave out the denominator.

Rejection versus Importance Sampling

Rejection sampling can generate a lot less samples per time, but each one is good.

Importance Sampling generates lots of weighted samples but they can be low quality, since a new sample can come in with far higher weight and completely swamp all previous samples. Practically, this is seen as having "large variance".

Sampling on a Directed Graph

Given a directed graph G on discrete variables X. Suppose some subset $E \subset X$ has been supplied and one wishes to sample the remaining variables conditioned on this *evidence*.

Rejection sampling Use proposal distribution $q(X) = p(X)$. Since required distribution is $p(X|E)$, we get that all samples of X agreeing with the evidence E should be accepted. The overall acceptance rate is $p(E)$, but for samples agreeing with the evidence, the acceptance rate is 1. For DAGS, originally called Probabilistic Logic Sampling, by Henrion 1988.

Importance sampling Use proposal distribution $q(X/E) = \prod_{x \in X/E} p(x|\text{par}(x))$. Since required distribution is $p(X|E)$ has a normalisation term $p(E)$ we do not want to or cannot compute, use weight

$$w = \frac{p(X)}{q(X/E)} = \prod_{e \in E} p(e|\text{par}(e))$$

For DAGS, originally called likelihood weighting, by Shachter and Peot 1989.

Adaptive Rejection Sampling

One can use the samples generated so far to get a proposal distribution closer to the required distribution. This is generally quiet complex but worth the effort.

For the directed graph situation previously (Cheng and Drudzel, *JAIR*, 2000):

Adaptive Importance sampling Keep data tables for each variable $x \in X/E$ to estimate $p(x|\text{par}(x) \cup E)$ from the weighted samples formed so far. This value gives the optimal importance sampler, if only we could estimate it precisely! Sampling quality improves as the estimate improves. This is the best algorithm todate for this general family, and quiet sophisticated.

Next Week

- Review Prof. Tirri's notes on Proportions.
- Play with the BetaCoinExperiment applet.