C++ implementation of the Metropolis-Hasting Algorithm

Nicholas Malaya

Institute for Computational Engineering and Sciences
The University of Texas at Austin

December 2nd, 2014

Motivation

Extending MC

- We discussed Monte-Carlo (MC) methods in class
- But we can do better– Markov Chain Monte-Carlo (MCMC) is an extension of MC

MCMC is ubiquitous

- statistics, physics, chemistry, biology, engineering, business, linguistics, etc.
- Especially research: Machine Learning, Uncertainty Quantification, Astrophysics, Particle Physics, Risk assessment
- Primarily used for calculating numerical approximations of multi-dimensional integrals

MCMC

What is it?

- MC = Monte-Carlo
-
-
- Monte-Carlo = Stochastic (i.e. random numbers)

Essentially, a method to generate random

MCMC

What is it?

- CMC = Chain Monte-Carlo
-
- Chain = A series of steps
- Monte-Carlo = Stochastic (i.e. random numbers)

Essentially, a method to generate random steps

MCMC

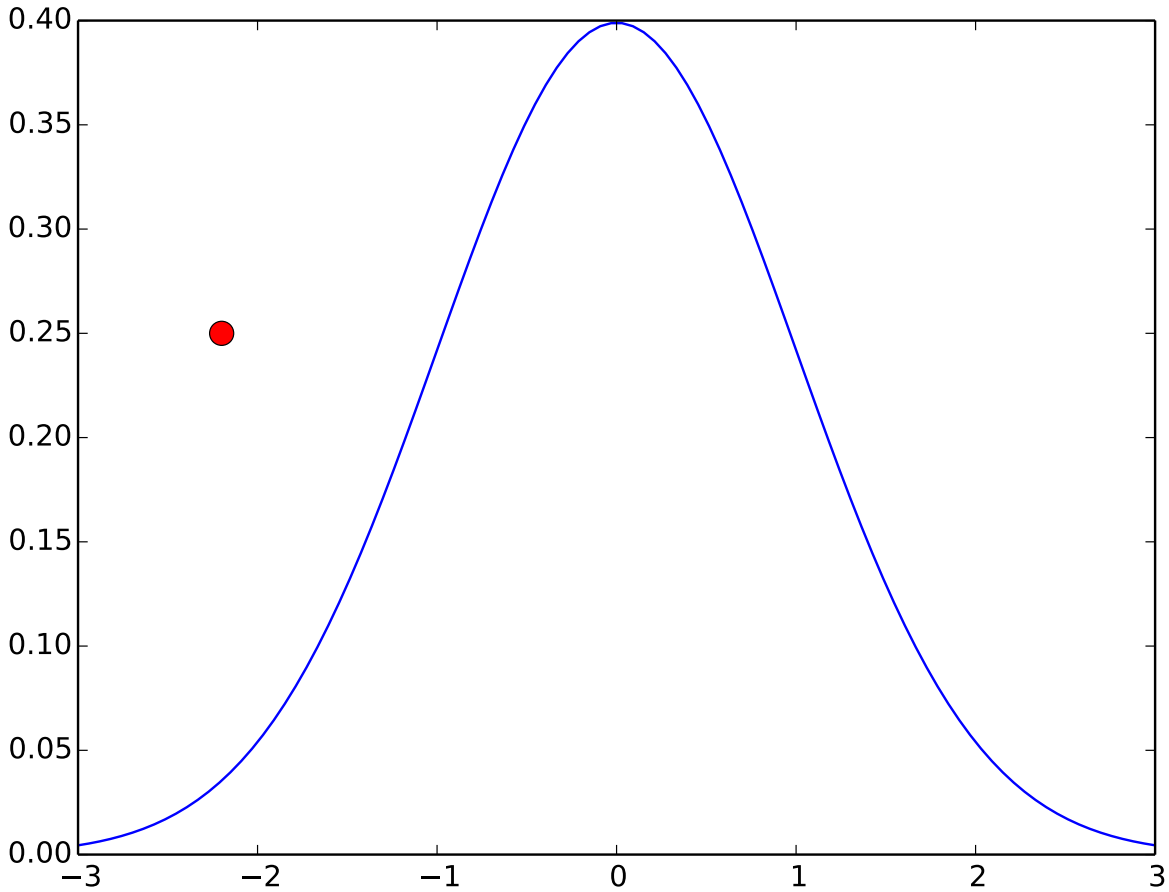
What is it?

- MCMC = Markov Chain Monte-Carlo
- Markov = Each step only depends on previous
- Chain = A series of steps
- Monte-Carlo = Stochastic (i.e. random numbers)

Essentially, a method to generate random steps correlated with the previous step

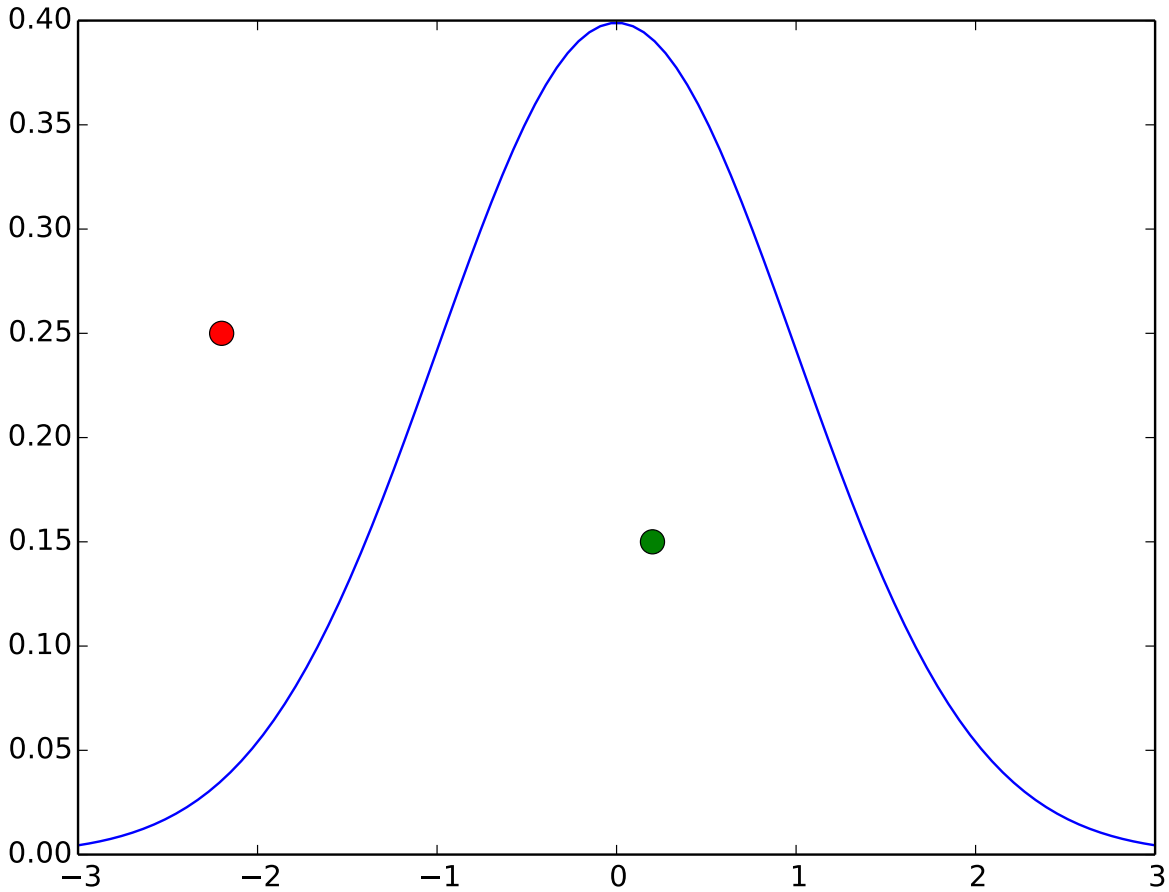
Enter Metropolis-Hastings

Intuitive Picture: Sampling a Gaussian Distribution



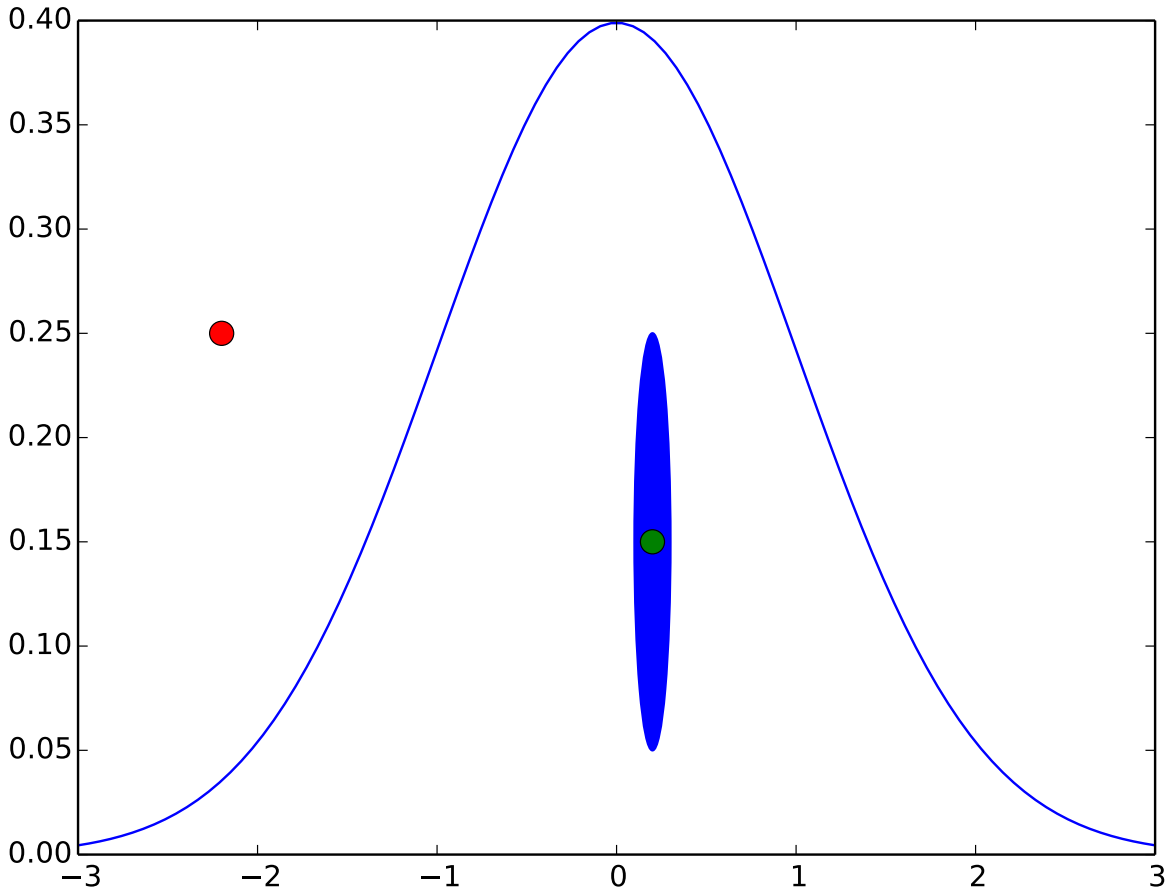
Enter Metropolis-Hastings

Intuitive Picture: Sampling a Gaussian Distribution



Enter Metropolis-Hastings

Intuitive Picture: Sampling a Gaussian Distribution



Enter Metropolis-Hastings

What is it?

- Metropolis-Hastings is an MCMC method for generating these samples.

Algorithm

- Start at current step (ϕ_1)
- Draw proposal: $q(\tilde{\phi}_2|\phi_1) = N(\tilde{\phi}_2|\phi_1, \nu^2)$
- Draw from uniform: $u \sim U(0,1)$
- if: $u < \frac{\pi_n(\tilde{\phi}_2)}{\pi_n(\phi_1)}$ (π_n is bayesian posterior, etc.)
  - $\phi_2 = \tilde{\phi}_2$
- else: $\phi_2 = \phi_1$

Enter Metropolis-Hastings

What is it?

- Metropolis-Hastings is an MCMC method for generating these samples.

Algorithm

- Start at current step (ϕ_1)
- Draw proposal: $q(\tilde{\phi}_2|\phi_1) = N(\tilde{\phi}_2|\phi_1, \nu^2)$
- Draw from uniform: $u \sim U(0,1)$
- if: $u < \frac{\pi_n(\tilde{\phi}_2)}{\pi_n(\phi_1)}$ (π_n is bayesian posterior, etc.)
  - $\phi_2 = \tilde{\phi}_2$
- else: $\phi_2 = \phi_1$

Question: How do we choose ν?

Project Status

Library Snapshot

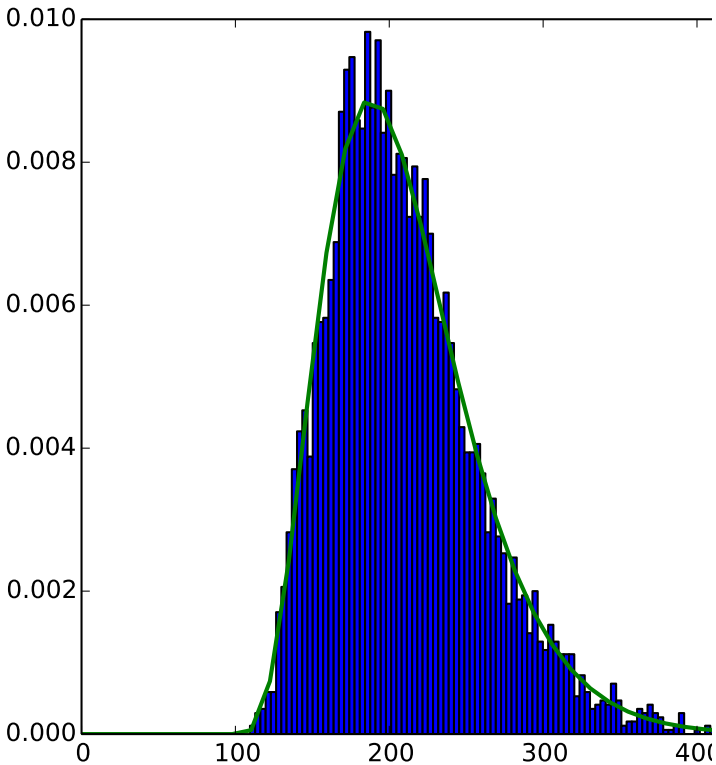
- C++, threaded with OpenMP
- Makefile build system support for:
  - Asus laptop
  - TACC's Lonestar supercomputer
- Git revision control
- CLOC: 254 SLOC C++

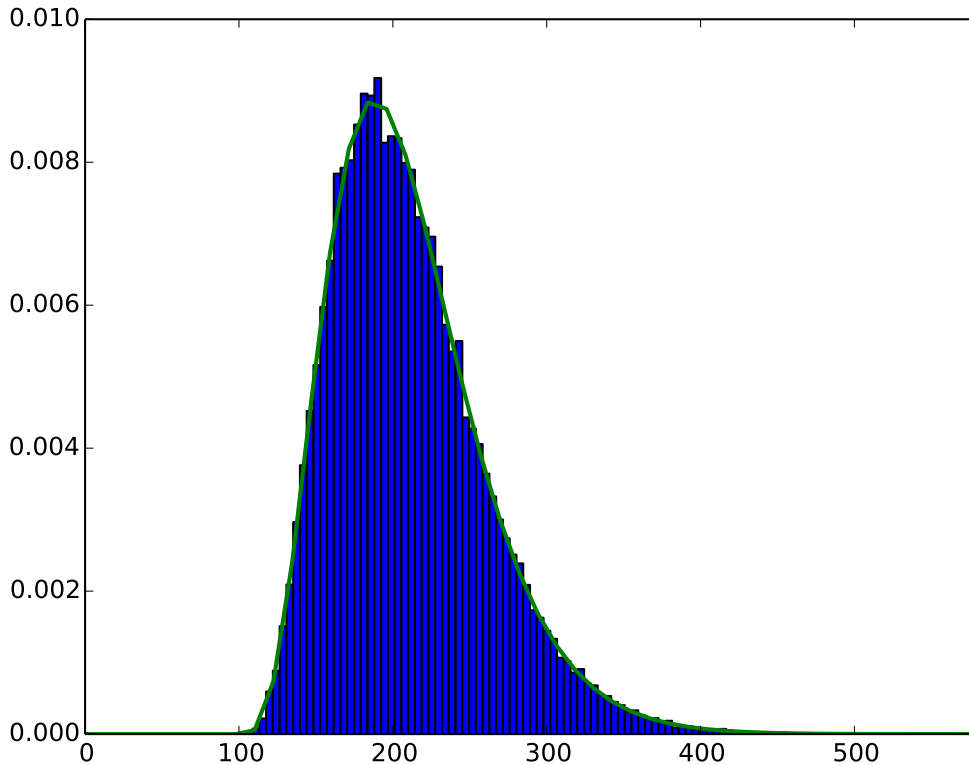
Capabilities

- Gamma, Beta, Normal Statistical Distributions

Metropolis-Hastings Sampling in C++

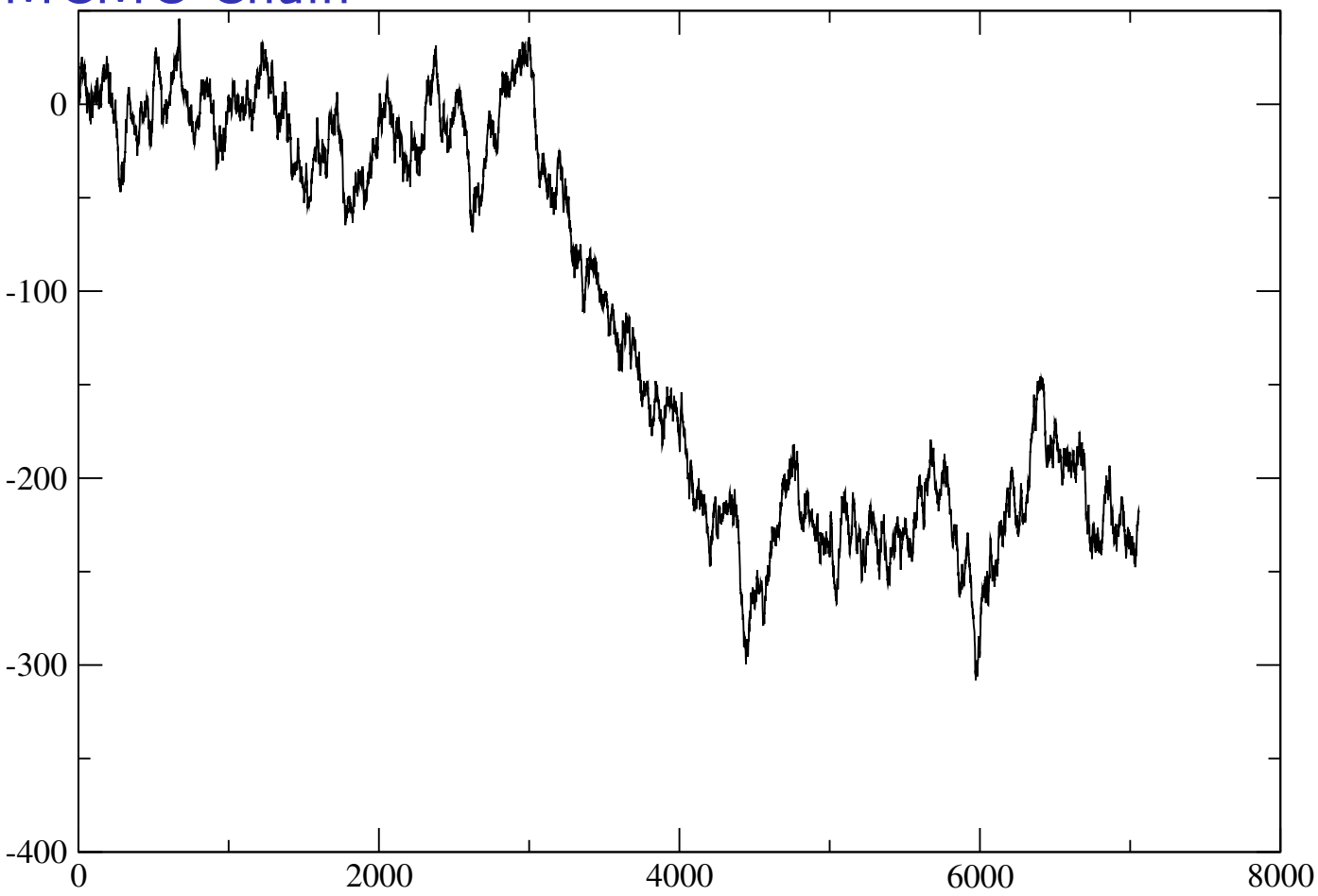
Sampling a Gamma Distribution





Burn-in

MCMC Chain



OpenMP Threading

Embarassingly Parallel

- Each chain is completely independent
- Thus, each thread can run completely independent, aside from ALL_REDUCE at end
- As a result, we anticipate nearly-perfect scalability.

Threading the application

- compile with: g++ test.cpp -fopenmp -lpthread
- OMP_GET_NUM_PROCS != OMP_GET_MAX_THREADS
  - Why might that be?

Conclusions

Summary

- MCMC is a critical algorithm across a wide variety of industries and STEM fields
- We have demonstrated a simple C++ implementation
- This implementation should scale well on a single lonestar node

Thank you!

- nick@ices.utexas.edu for code
- Have a Markovian Day!