Markov Chain Monte Carlo on the GPU

Alex Kaiser

Courant Institute of Mathematical Sciences
New York University

Dec 18, 2012

Sampling

- Problem: given a multidimensional random variable X, want to generate samples of this random variable
- We know how to evaluate the probability density function of f_X up to a constant
- Ex: $X \sim N(\mu, \Sigma)$, an N dimensional Gaussian

$$f_X(\vec{x}) = \alpha \exp\left(-\frac{1}{2}(\vec{x}-\mu)^t \Sigma^{-1}(\vec{x}-\mu)\right)$$

- Here, $\alpha = (2\pi)^{-N/2}\det(\Sigma)^{-1/2}$. In general this normalization factor is a nuisance to compute.
- Need an algorithm to produce these samples

What is this for?

- Estimating expected values
  - Let $x_1 \dots x_K$ be samples from the distribution. Then an estimator for the expected value is given

$$E[\phi(X)] \approx \frac{1}{K} \sum_{i=1}^{K} \phi(x_i)$$

- Integrate something, usually in high dimensions
  - This can be phrased as an expected value

$$\int \phi(x) f_X(x)\, dx = E[\phi(X)] \approx \frac{1}{K} \sum_{i=1}^{K} \phi(x_i)$$

- Parameter estimation
- Simulations in statistical mechanics

Metropolis Algorithm

- The standard algorithm is the *Metropolis Algorithm*
- Design a Markov chain with relatively simple dynamics (explained shortly)
- Stationary distribution of the Markov chain is the distribution we want to sample
- Create a random walk with the Markov chain, the values of the walk are samples of the distribution
- Don't need to know the normalization factor
- This type of sampler is called a *Markov chain Monte Carlo* sampler (MCMC)

Metropolis Algorithm

- Let X_t be the sample (position of the Markov chain) at time t
- Propose a move Y by sampling from a *proposal distribution* T
  - The distribution T is something easy to sample (i.e. uniform)
  - For complicated distributions, lots of work to define a good T
- Evaluate a likelihood function for the move

$$q(Y|X_t) = \frac{T(X_t|Y)f_X(Y)}{T(Y|X_t)f_X(X_t)}$$

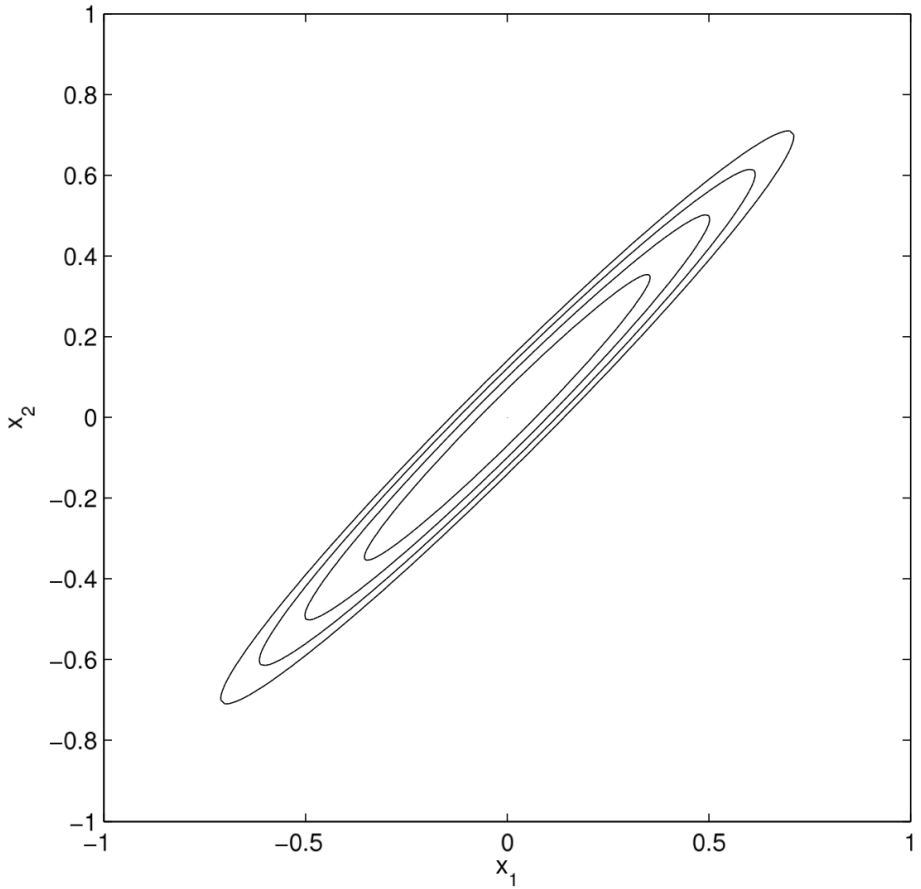
- In case of uniform T, this is $f_X(Y)/f_X(X_t)$
- If $q(Y|X_t) > 1$, accept. Otherwise, accept with probability q.
- If we accept $X_{t+1} = Y$, else $X_{t+1} = X_t$.

Problems with metropolis

- Correlation. Following samples are correlated with the previous sample.
  - Ideally samples would be independent, but the next sample is moved directly from the last
  - To quantify this, compute the *autocorrelation time* τ, a measure of the correlation of the sequence with itself.
  - If M total samples are drawn, then the effective number of independent samples is M/τ
- Serial. There is very little parallelism in this algorithm.
  - Astrophysics researchers using these methods for inference report running for days.

Skewed distributions

$$p(\vec{x}) \propto \exp\left(-\frac{(x_1 - x_2)^2}{2\epsilon} - \frac{(x_1 + x_2)^2}{2}\right)$$



Stretch Move

- Alternative MCMC method developed at Courant by Goodman and Weare in 2010
- Similar structure to Metropolis
  - Still has an accept/reject step with a likelihood calculation
- Use an *ensemble* of walkers
  - Generate a multiple samples on each iteration, each walker is a valid sample
  - Develop an update rule that allows the walkers to be updated in parallel
- Source for some figures and formulas: J. Goodman and J. Weare, *Ensemble Samplers with Affine Invariance*, Communications in Applied Mathematics and Computational Science, v. 5, 2010.

Stretch Move - Setup

- Split the walkers into two groups X_{red}, X_{black}
  - To make proofs work all cannot be updated at once
- To update a walker, randomly select a walker from the other group, then sample on a line between the two walkers.
  - This gives *affine invariance*, algorithm is not sensitive to skewed distributions
- Each group can be updated in parallel. K walkers gives precisely $K/2$ chunks of work that can be computed in parallel.
- Algorithm will require an auxiliary random variable $z \sim g(Z)$
  - Distribution must satisfy $g(1/z) = (1/z)g(z)$
  - Usually $g(z) = 1/\sqrt{2z}$ on $(1/2, 2)$
  - Picked this way to make proofs work
  - This is easy to sample by inverting CDFs. Becomes a quadratic function of a $U(0, 1)$ random variable.

Stretch Move - Algorithm

- Want to compute $X_{red}(k, t+1)$
- Randomly select a walker from the complementary ensemble $X_{black}(j)$
- Sample $z \sim g(Z)$
- Compute a proposed move Y

$$Y = X_{black}(j) + z(X_{red}(k, t) - X_{black}(j))$$

- Compute a likelihood

$$q = z^{N-1} f_X(Y) / f_X(X_{red}(k, t))$$

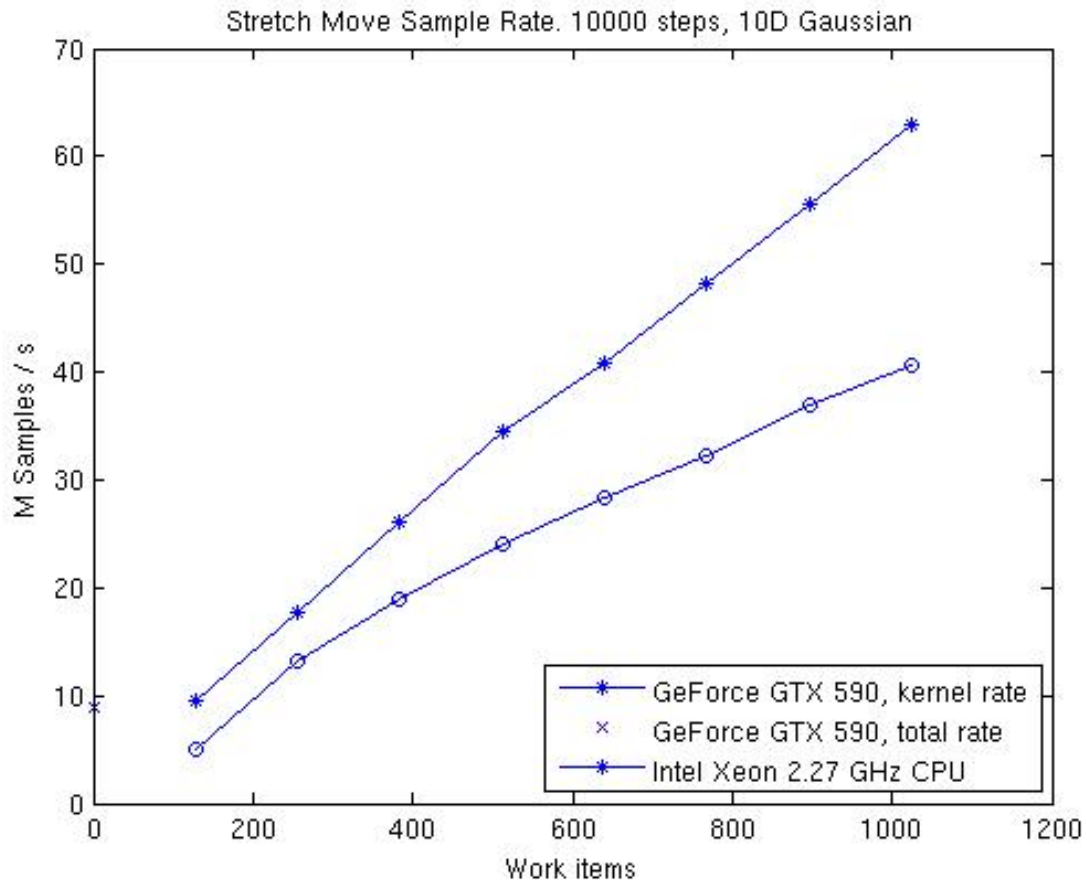
- If $q > 1$, accept. Otherwise, accept with probability q.
- If accept $X_{red}(k, t+1) = Y$, else $X_{red}(k, t+1) = X_{red}(k, t)$.

Stretch Move on the GPU

- Each work-item owns two walkers, one from each group, whole group can be updated in parallel
- Use as many walkers as can be stored in memory, up to the twice the maximum number of work-items the hardware can handle
- Barrier required before updating complementary ensemble, updated walkers need to be read
- No race conditions - clear ownership, each work-item writes to two and only two walkers
- If the PDF takes a consistent amount of time to evaluate, this is almost perfectly load balanced
- Use as many walkers as can be stored in memory, up to the twice the maximum number of work-items the hardware can handle

Stretch Move - performance

- Almost perfect linear scaling with the number of work-items.



Demo

- Looking at pre-made histograms is only so much fun. Let's make some now.

Markov Chain Monte Carlo on the GPU

Alex Kaiser

Courant Institute of Mathematical Sciences
New York University

Dec 18, 2012