Parallel Markov Chain Monte Carlo Simulation by Pre-Fetching

Harsh Sheth
140400012
Mechanical Engineering
IIT Bombay

Manan Doshi
140100015
Mechanical Engineering
IIT Bombay

Tejas Srinivasan
140100025
Mechanical Engineering
IIT Bombay

Deep Tavker
150010001
Mechanical Engineering
IIT Bombay

***Abstract*—We attempt to recreate the method by Brockwell [1] to speed up MCMC simulations to compute the Maximum likelihood of parameters of an ARFIMA model given the data. We do this using pre-fetching.**

I. Introduction

Markov Chain Monte Carlo methods are a class of methods that use a Markov chain to generate samples from a target probability distribution. The generation and traversal of the Markov chain is done using the Hastings Algorithm. The generated samples are then used in the Monte Carlo simulation. We speed up this simulation using pre-fetching.

Pre-fetching allows us to perform the MCMC simulation faster by creating a tree of possible states, and then computing the acceptance probabilities of all these states simultaneously rather than one at a time (since this is the bottleneck step in the process).

II. ARFIMA models

The Autoregressive integrated moving average is a generalization of the Autoregressive Moving Average model. It can be used to generate time series data, y_t given a set of parameters. The model is given by

$$A(L)(1-L)^d y_t = B(l) z_t \tag{1}$$

where,

$$Ly_t = y_{t-1}$$
$$A(L) = 1 - \sum_{i=1}^{p} \rho_i L^i$$
$$B(L) = 1 + \sum_{i=1}^{p} \theta_i L^i$$
$$z_t = \mathcal{N}(\mu, \sigma^2)$$

Here, z_t represents the noise in the model. We consider a simplified model with $A(L) = 1 - \rho L$ and $B(L) = 1$

We wish to solve the inverse problem. Given timeseries data y_t, we wish to find the set of parameters that was most likely responsible for generating this data.

III. Max Likelihood Estimate

Let the set of parameters in the ARFIMA model be θ. This consists of μ and σ^2, the parameters of the error term, ρ and d.

The problem we are attempting to solve is to essentially find

$$\theta_{max} = \text{argmax}_\theta \mathbb{P}(\theta|y) \tag{2}$$

Using Bayes' theorem, we can write the RHS as

$$\mathbb{P}(\theta|y) = \mathbb{P}(y|\theta) \times \frac{\mathbb{P}(\theta)}{\sum_\theta \mathbb{P}(y|\theta)} \tag{3}$$

The denominator is constant for different values of θ for a given y. Thus, when maximizing over θ, we can ignore this term. The term $\mathbb{P}(\theta)$ is the prior distribution of θ. We can set this to a known distribution of θ, if it is known beforehand. This is Maximum A Posteriori Estimation. If we set it to a uniform distribution, Equation 2 reduces to

$$\theta_{max} = \text{argmax}_\theta \mathbb{P}(y|\theta) \tag{4}$$

This is the maximum likelihood estimate of θ

IV. Monte Carlo method to estimate the MLE

We first assume that we can generate samples from the distribution of $\mathbb{P}(\theta|y)$. This is non trivial and will be done using MCMC methods. Given that we can sample from that distribution, finding the MLE is trivial and just involves taking the mean of the obtained parameters

V. Hastings algorithm

We use the Metropolis-Hastings algorithm[2] to generate samples from the distribution $\mathbb{P}(\theta|y)$.

The algorithm is as follows

```
Require: Jumping distribution $J(\theta_i|\theta_j)$, $\theta_0$
  Unscaled probabilities $\Pi(\theta)$
  $t = 0$
  loop
    Generate proposal $\hat{\theta}_t$ from distribution $J(\theta|\theta_t)$
    Compute acceptance probability
    $\rho(\theta_t, \hat{\theta}_t) = \min\left(1, \frac{\Pi(\hat{\theta}_t)}{\Pi(\theta_t)} \frac{J(\theta_t|\hat{\theta}_t)}{J(\hat{\theta}_t|\theta_t)}\right)$
    Accept the proposal with probability $\rho$
    if Proposal accepted then
      $\theta_{t+1} = \hat{\theta}_t$
```

```
        else
            $\theta_{t+1} = \theta_t$
        end if
    end loop
```

The biggest advantage of this method is that we can set $\Pi(\theta) = \mathbb{P}(y|\theta)$, which can be computed from Equation 1. This is because

$$\frac{\Pi(\theta_1)}{\Pi(\theta_1)} = \frac{\mathbb{P}(\theta_1|y)}{\mathbb{P}(\theta_2|y)} = \frac{\mathbb{P}(y|\theta_1)}{\mathbb{P}(y|\theta_2)}$$

The Markov chain generated by the algorithm as an equilibrium distribution of $P(\theta|y)$. That is, when run long enough, the states generated by the algorithm will correspond to the target distribution. This initial time required for the algorithm to converge to the target distribution is called the burnout period.

VI. Speeding it up

A. Markov chains in parallel

One standard way of speeding up MCMC simulations is to run multiple Markov chains in parallel. While this will speed up the simulations by generating more samples, every chain has to initially go through the burnout phase. If the burnout phase is long, no amount of parallelization will reduce the burnout time.

B. Pre-Fetching

We try to reduce the burnout time by speeding up a single Markov Chain using parallelization. Instead of running multiple chains in parallel, we run the same chain using multiple processors. This is non trivial since the Hastings algorithm is a sequential algorithm where θ_t depends on θ_{t-1}.

The computational bottleneck in the Hastings algorithm for out application is the computation of the potential $\Pi(\theta)$. We achieve the speedup by preemptively computing states upto the depth of $\log_2(\text{num\_processors})$ without knowing which proposal will be accepted. We compute the potentials of these states in a parallel fashion where every processor computes the potential of its assigned state. Processor 0 can then rapidly traverse the depth by accepting and rejecting proposals.

This algorithm involves a lot of wasteful computations since states whose potential does not need to be computed ends up being computed. However, it provides an effective way to speedup a process that is inherently sequential.

VII. Results

We ran this simulation for 1000 states in our Markov Chain, with the data being generated using the parameters $d = 0.5, \phi = 0.5, \sigma^2 = 0.5$. In pre-fetching, we would search upto a depth of H=3 states ahead. In 1000 states, we were not able to get the parameter values to converge to the true values used to generate the data.

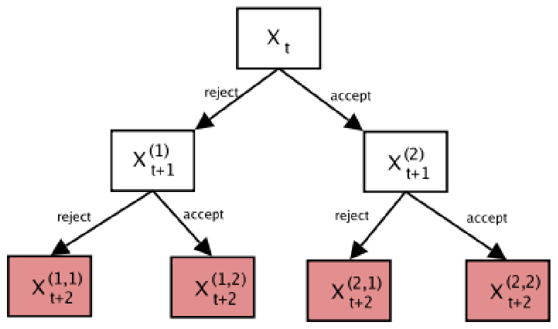
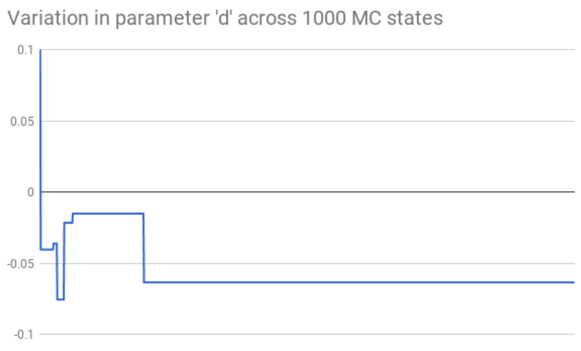
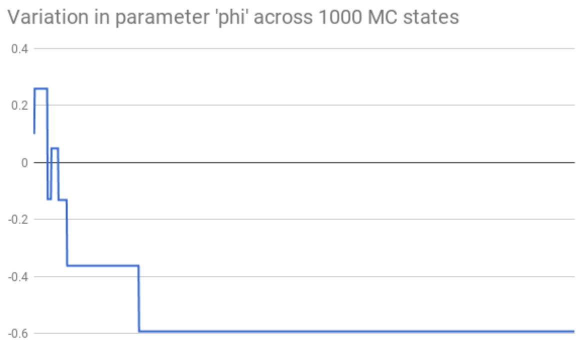
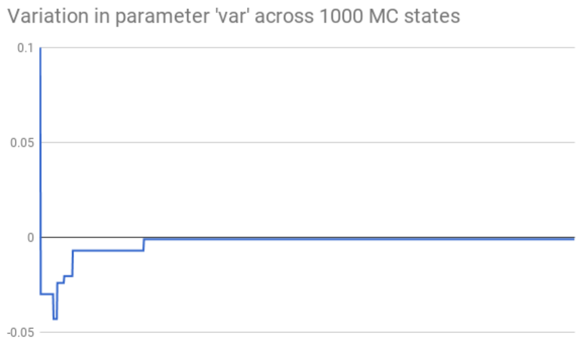


Fig. 1. The possible outcomes in two iterations of a Metropolis-Hastings sampler. In two steps, there are only four unique values, those in the shaded leaf nodes, for which likelihoods must be computed. Parents have the same value as their left(reject)-children







References

[1] Anthony E Brockwell. Parallel markov chain monte carlo simulation by pre-fetching. *Journal of Computational and Graphical Statistics*, 15(1):246–261, 2006.

[2] Christian P Robert and George Casella. The metropolishastings algorithm. In *Monte Carlo Statistical Methods*, pages 231–283. Springer, 1999.