

# Firefly Monte Carlo: Exact MC with data subsets

Feynman Liang, Max Chamberlin, Kai Xu

April 29, 2016

## Contents

<b>1</b>	<b>Introduction:</b>	<b>1</b>
<b>2</b>	<b>Theory</b>	<b>2</b>
2.1	Performance comparison . . . . .	3
2.2	Discussion on the lower bound $B_n(\theta)$ . . . . .	4
2.2.1	MAP-tuning the lower bound . . . . .	4
<b>3</b>	<b>Implementation</b>	<b>4</b>
3.1	Provided reference software . . . . .	5
3.2	Our contributions to the project . . . . .	5
	TODO: Proofread the paper for flow	

## 1 Introduction:

TODO: The report should describe the rationale for selecting the paper and the particular experiments for replication

### The idea:

The key motivation for FlyMC is to achieve speed-ups over typical MCMC, where we would like to sample some parameter  $\theta$  of interest given datapoints  $x_{1:N}$ . FlyMC does this by estimating a large data-set's likelihood using smaller subsets of the data.

To work, FlyMC requires a non-negative function  $B_n(\theta)$  that acts as a lower bound on the likelihood,  $B_n(\theta) \leq L_n(\theta) = P(x_n|\theta)$ . It also introduces auxiliary variables, which will act as indicators for the points for which the further computation (evaluations of  $L_n(\theta)$ ) are required. This gives the motivation for the name, FlyMC, since these auxiliary variables can be thought of as lighting up when further likelihood evaluations are desired. Finally, with these terms in place, an intuitive explanation of FlyMC can be given. We have a lower bound for the likelihood,  $B_n(\theta)$ , which accounts for a large part of the probability mass on the distribution from  $\theta$ . However, there is still a difference  $L_n(\theta) - B_n(\theta)$ , which we are uncertain of and which we would like to compute adjustments for through sampling. FlyMCn constructs a way of sampling from this residual probability mass, which is hopefully much smaller than  $L_n(\theta)$ , and so may result in large reductions in sampling time. The idea is neatly summarised in the graph below:

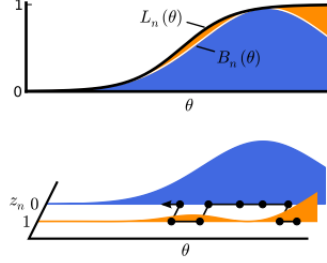


Figure 1: Graphical representation of the decomposition of  $L_n$  into a  $L_n - B_n$  part associated with bright variables  $z_n = 0$  and a  $B_n$  part associated with dim variables  $z_n = 0$  [MA14]

### Rationale for selecting the paper:

We chose FlyMC because of the ubiquity of MCMC sampling methods which see use not just in the fields of machine learning and Bayesian statistics, but also in the fields of computational physics, biology and computational linguistics. If large speed-ups could be obtained using the methods expounded in FlyMC, this would be of great benefit to many different academic communities.

Furthermore, in Bayesian statistics, the recent development of MCMC methods has been a key step in making it possible to compute large hierarchical models that require integrations over hundreds or even thousands of unknown parameters, which is an exciting area of further research. However, to make MCMC work quickly for data of the order of that seen in the 'Big Data' era, any speed-up that is possible would be well received.

In addition to this, FlyMC is based on a mathematically elegant idea. The idea is that when performing inference, all that is necessary is to

### The Experiments to be replicated:

In this practical, we conducted the following three experiments:

- Logistic regression: For this particular task, we tested we tested FlyMC's preformance for the task of classifying 7s and 9s on MNIST, with prior  $P(\theta) \sim \mathcal{N}(0, I)$ .
- Softmax classification: For this task, we tested FlyMC's ability to fit a softmax classification model on three CIFAR-10 classes (airplane, automobile and bird) using Langevin dynamics
- Robust linear regression: For logistic regression task, we generated synthetic data and evaluated the performance of FlyMC in comparison to regular MCMC.

The first two of these three experiments were data-sets used in the paper. However, for the third data-set we decided it would be intersting to see if the same results claimed by a paper could be found in a data-set that was not hand-picked by the authors.

## Main Body

### 2 Theory

Given independent and identically distributed observations  $X = \{x_n\}_{n=1}^N$ , one is often interested in the *likelihood*

$$L(\theta) = P(X|\theta) = \prod_{n=1}^N P(x_n|\theta) = \prod_{n=1}^N L_n(\theta)$$

For example, when performing Metropolis-Hastings MCMC to sample a posterior distribution  $P(\theta|X) \propto P(X|\theta)P(\theta)$  one needs to compute the acceptance probability:

$$A(\theta'|\theta) = \min \left\{ 1, \frac{q(\theta|\theta')L(\theta')P(\theta')}{q(\theta'\theta)L(\theta)P(\theta)} \right\}$$

This means that each iteration requires at least one  $L(\theta)$  (can be cached from the prior iteration) evaluation of the likelihood over all  $N$  points. Unfortunately, in the era of “Big Data”  $N$  may be on the order of petabytes and computing  $L(\theta)$  at each iteration may be prohibitively expensive.

Firefly Monte Carlo (FlyMC) is a method which addresses the computational problem of evaluating  $L(\theta)$  for a large number of data points  $N$  through exact Monte Carlo estimates of the full dataset likelihood using smaller, tractable subsets. As an auxiliary variable method, it works by augmenting the joint distribution with hidden *brightness* variables  $Z = \{z_n \in \{0, 1\}\}_{n=1}^N$  to form a *complete data likelihood*

$$P(X, Z|\theta) = \prod_{n=1}^N P(x_n|\theta)P(z_n|x_n, \theta)$$

Notice that we can recover the desired likelihoods simply through marginalization

$$\sum_Z P(X, Z|\theta) = \sum_{z_1} \cdots \sum_{z_N} \prod_{n=1}^N P(|x_n|\theta)P(z_n|x_n, \theta) = \prod_{n=1}^N P(|x_n|\theta) \sum_{z_n} P(z_n|x_n, \theta) = L(\theta)$$

In FlyMC, the brightness variables are chosen to have Bernoulli distribution

$$P(z_n|x_n, \theta) = \left( \frac{L_n(\theta) - B_n(\theta)}{L_n(\theta)} \right)^{z_n} \left( \frac{B_n(\theta)}{L_n(\theta)} \right)^{1-z_n} \quad (1)$$

where  $B_n(\theta)$  is a lower bound on the likelihood satisfying

$$0 \leq B_n(\theta) \leq L_n(\theta) \quad (2)$$

$$\prod_{n \in A} B_n(\theta) \text{ can be efficiently computed for any data subset } A \subset X \quad (3)$$

1.  $0 \leq B_n(\theta) \leq L_n(\theta)$
2.  $\prod_{n=1}^N B_n(\theta)$  can be efficiently computed

2 is required for 1 to be a valid probability distribution. To see why 3 is necessary, consider computing the complete data likelihood

$$\begin{aligned} P(X, Z|\theta) &= \prod_{n=1}^N P(x_n|\theta)P(z_n|x_n, \theta) \\ &= \prod_{n=1}^N L_n(\theta) \left( \frac{L_n(\theta) - B_n(\theta)}{L_n(\theta)} \right)^{z_n} \left( \frac{B_n(\theta)}{L_n(\theta)} \right)^{1-z_n} \\ &= \prod_{x_n \in X_{bright}} (L_n(\theta) - B_n(\theta)) \prod_{x_n \in X_{dim}} (B_n(\theta)) \end{aligned} \quad (4)$$

$$= \prod_{n=1}^N B_n(\theta) \prod_{x_n \in X_{bright}} \frac{L_n(\theta) - B_n(\theta)}{B_n(\theta)} \quad (5)$$

where the bright points  $X_{bright} = \{x_n : z_n = 1\}$  and dim points  $X_{dim} = \{x_n : z_n = 0\}$  give rise to the “Firefly” in the paper title. Notice that as a consequence of our choice for 1, 4 decomposes into two products where  $\frac{L_n(\theta) - B_n(\theta)}{B_n(\theta)}$  only needs to be evaluated for  $x_n \in X_{bright}$ . By assuming 3, the other term  $\prod_{n=1}^N B_n(\theta)$  can be efficiently computed so the computational complexity is determined by  $X_{bright}$ .

## 2.1 Performance comparison

Instead of evaluating  $L_n(\theta)$  over all  $N$  points, FlyMC only evaluates the likelihood over a subset of points  $X_{bright}$ . This provides a speedup of  $\frac{N}{\#X_{bright}}$  per iteration. However, since the brightness variables  $z_n$  are themselves, to understand the performance improvements we consider the expected number of bright points  $E[\#X_{bright}]$ .

Note that for any point  $x_n$ , 1 implies the probability  $x_n$  is bright is given by  $\frac{L_n(\theta) - B_n(\theta)}{L_n(\theta)}$ . Hence, the number of bright points is given by

$$E[\#X_{bright}] = \sum_{n=1}^N E[z_n] = \sum_{n=1}^N \int P(\theta|X) \frac{L_n(\theta) - B_n(\theta)}{L_n(\theta)} d\theta \quad (6)$$

This says that the number of bright points (and hence the speedup achieved by FlyMC) is related to the gap between the true likelihood  $L_n(\theta)$  and the lower bound  $B_n(\theta)$ , weighted by the posterior  $P(\theta|X)$ .

This result makes intuitive sense. Consider the extreme case where for all  $n$ :  $B_n(\theta) = 0$ . Then  $P(z_n = 1|x_n, \theta) = 1$  so every data point is bright (i.e.  $X = X_{bright}$ ) and 4 reduces to evaluating the full dataset likelihood  $\prod_{n=1}^N L_n(\theta)$  as usual.

In the other extreme, if  $B_n(\theta) = L_n(\theta)$  so all data points are dim. At first it appears that FlyMC counterintuitively offers a  $\frac{N}{\#X_{bright}} = \frac{N}{0} = \infty$  speedup. However, considering 4 reveals that what is computed is  $\prod_{n=1}^N B_n(\theta) = \prod_{n=1}^N L_n(\theta)$ , which by assumption 3 can be computed efficiently. But then computing  $L(\theta) = \prod_{n=1}^N L_n(\theta)$  is efficient and hence is no longer  $O(N)$ , contradicting our earlier assumptions.

## 2.2 Discussion on the lower bound $B_n(\theta)$

The integral role played by the lower bound  $B_n(\theta)$  makes it an important topic of investigation. In this section, we investigate how reasonable and justified are the assumptions required for developing FlyMC's theory.

One of our assumptions (3) was that computation of  $\prod_{n=1}^N B_n(\theta)$  be efficient new  $\theta$ . For example, choosing  $B_n(\theta)$  to be exponential family will make computing  $\prod_{n=1}^N B_n(\theta)$  for each new  $\theta$  be  $O(1)$  because

$$B_n(\theta) \propto \exp(\langle \eta_n, T(\theta) \rangle) \\ \prod_{n=1}^N B_n(\theta) \propto \exp(\langle \sum_{n=1}^N \eta_n, T(\theta) \rangle)$$

and  $\sum_{n=1}^N \eta_n$  can be evaluated once and cached. We believe that this assumption is quite restrictive and have had some difficulty coming up with functional forms which are not exponential family that satisfy this assumption.

The other assumption (2) required  $B_n(\theta)$  be a non-negative lower bound for  $L_n(\theta)$ . This is trivially satisfied by setting  $B_n(\theta) = 0$ , but to be practical we should try to find a lower bound with the smallest gap possible. Several forms of bounds exist for commonly used likelihoods; the original authors consider the Jaakola and Jordan bound on the log logistic function. However, this requirement is quite restrictive and unrealistic for more complex use cases. For example, if  $L_n(\theta)$  is given by some complicated Bayes net or neural network, then a valid  $B_n(\theta)$  for use with FlyMC is not immediately obvious.

### 2.2.1 MAP-tuning the lower bound

In certain cases (e.g. Jaakola and Jordan bound on log logistic),  $B_n$  is part of a parametric family of lower bounds  $\{B_n(\cdot, \xi)\}_\xi$  where  $\xi$  is a free parameter. Since the number of bright points determines the speedup offered by FlyMC, 6 suggests that we should choose  $\xi$  to minimize  $E \left[ \frac{L_n(\theta) - B_n(\theta)}{L_n(\theta)} \right]$  for each datapoint  $n$  where the expectation is taken over the posterior distribution  $P(\theta|X)$ .

The authors of FlyMC propose a *MAP-tuned FlyMC* variant where the posterior expectation  $E \left[ \frac{L_n(\theta) - B_n(\theta)}{L_n(\theta)} \right]$  is replaced with the MAP point estimate  $\frac{L_n(\theta_{MAP}) - B_n(\theta_{MAP})}{L_n(\theta_{MAP})}$  and  $\xi = \arg\min_\xi L_n(\theta_{MAP}) - B_n(\theta_{MAP})$ .  $\theta_{MAP} = \arg\max_\theta P(\theta|X)$  can be found using stochastic gradient descent to handle large data-set concerns.

Our opinions about this method are mixed. While MAP tuning should provide a tighter lower bound than no tuning, approximating the posterior with a delta distribution centered at  $\theta_{MAP}$  seems suspicious. Other methods which could be considered here include first generating some posterior samples using an untuned chain or MCMC on a subset of the data, then tuning  $\xi$  over the posterior samples.

## 3 Implementation

FlyMC augments traditional MCMC by introducing brightness variables  $Z$  which are Gibbs sampled along with the original chain's state  $\theta$ . At a high level, the algorithm consists of the following steps:

1. Initialize  $\theta^{(0)}$
2. For  $t = 1, \dots, N_{iter}$

- (a) For  $n = 1, \dots, N$ : sample  $z_n^{(t)} \sim \text{Bernoulli}\left(\frac{L_n(\theta^{(t)}) - B_n(\theta^{(t)})}{L_n(\theta^{(t)})}\right)$  with  $\theta = \theta^{(t)}$  fixed
3. Propose  $\theta^{(t+1)} \sim q(\theta^{(t+1)}|\theta^{(t)})$
4. Compute acceptance probability  $A(\theta^{(t+1)}|\theta^{(t)}) = \min\left\{1, \frac{q(\theta^{(t)}|\theta^{(t+1)})P(X, Z|\theta^{(t+1)})P(\theta^{(t+1)})}{q(\theta^{(t+1)}|\theta^{(t)})P(X, Z|\theta^{(t)})P(\theta^{(t)})}\right\}$
5. Repeat from step 2 until chain has mixed, yielding a single sample  $z_n^{(\infty)}$

In step 4, the complete data likelihood is computed using 4 and hence only requires evaluating the likelihood over  $X_{\text{bright}}$ .

One possible criticism of the method described above is that step 2(a) requires sampling  $z_n$  from  $N$  Bernoulli distributions each with probability  $\frac{L_n(\theta) - B_n(\theta)}{L_n(\theta)}$  and hence direct sampling will again require  $N$  evaluations of  $L_n(\theta)$ . Fortunately, this can be avoided through *implicit sampling*, where a second MH-MCMC is used to sample the  $z_i$ . The motivation is that ideally many  $z_i = 0$  so it makes sense to only consider proposals where each dim point has a small  $q_{d \rightarrow b} \in [0, 1]$  probability of becoming bright. Then, evaluating the acceptance probability only requires computing  $L_n(\theta)$  at the  $z_n$  which have been proposed to change state.

### 3.1 Provided reference software

Reference code is provided by the authors<sup>1</sup>, which provides implementations FlyMC for logistic regression using MH-MCMC, softmax using Langevin dynamics, and sparse linear regression using slice sampling. The reference implementation includes an implicit sampling scheme for sampling brightness variables  $z_i$ .

### 3.2 Our contributions to the project

However, the only full example provided is a logistic regression example on toy data and is insufficient for reproducing the results reported in the paper. We forked the authors' code<sup>2</sup> and added our own modifications, most importantly:

1. Support for MAP-tuning of the lower bounds, using `scipy.optimize` for MAP estimation
2. Implementation of regular full-dataset MCMC, which is achieved by fixing  $\forall i : z_i = 1$
3. Improved numerical overflow handling in `LogisticModel`
4. Instrumentation for counting log-likelihood evaluations and outputting traces of chain state and log likelihood per iteration
5. Integration, parameter tuning, and reproduction of MNIST Logistic Regression, CIFAR-10 Softmax Classification, and Robust Linear Regression experimental results

We also fixed a significant bug in some caching code provided with the reference implementation and submitted our bugfix back upstream<sup>3</sup>. The problem was that computations of  $\left\{\frac{L_n(\theta) - B_n(\theta)}{B_n(\theta)} : n \in idx\right\}$  were being cached for vectors  $\theta$  and  $idx$ , but that caching was performed over the entire  $idx$  vector rather than elementwise. For example, if  $idx2 \subsetneq idx$ , then even though  $\left\{\frac{L_n(\theta) - B_n(\theta)}{B_n(\theta)} : n \in idx2\right\} \subset \left\{\frac{L_n(\theta) - B_n(\theta)}{B_n(\theta)} : n \in idx\right\}$  the reference implementation would still result in a cache miss because  $idx2 \neq idx$ . This bug causes the number of likelihood evaluations to be twice the number of proposed bright points  $\#X_{\text{bright}}$  at each iteration, which means that FlyMC will only offer speedups if  $\#X_{\text{bright}} < \frac{N}{2}$ .

## Results:

TODO: Provide analysis

TODO: Make up some tables about effective sample size and how the reduction in likelihood evaluations is bigger than the drop in ESS

<sup>1</sup><https://github.com/HIPS/firefly-monte-carlo>

<sup>2</sup><https://github.com/feynmanliang/firefly-monte-carlo>

<sup>3</sup><https://github.com/HIPS/firefly-monte-carlo/pull/1>

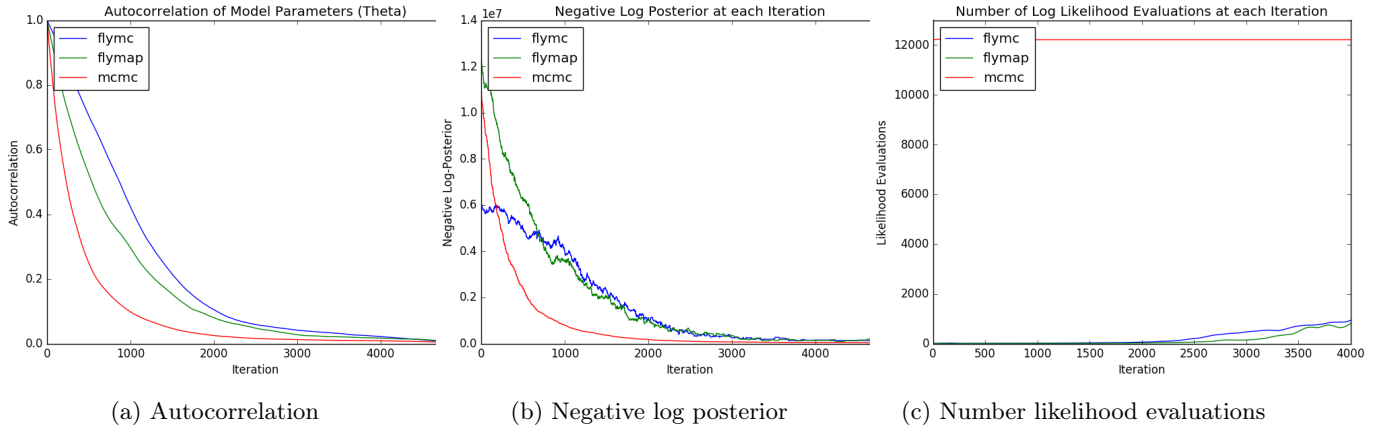


Figure 2: Logistic regression binary classification on MNIST with MH-MCMC

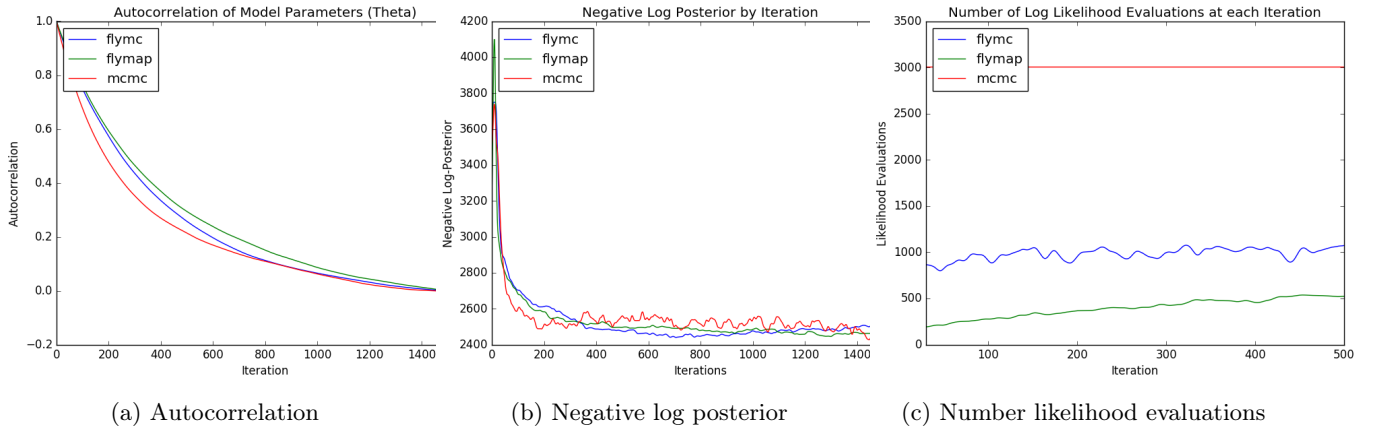


Figure 3: Softmax 3-way classification on CIFAR-10 with Langevin dynamics

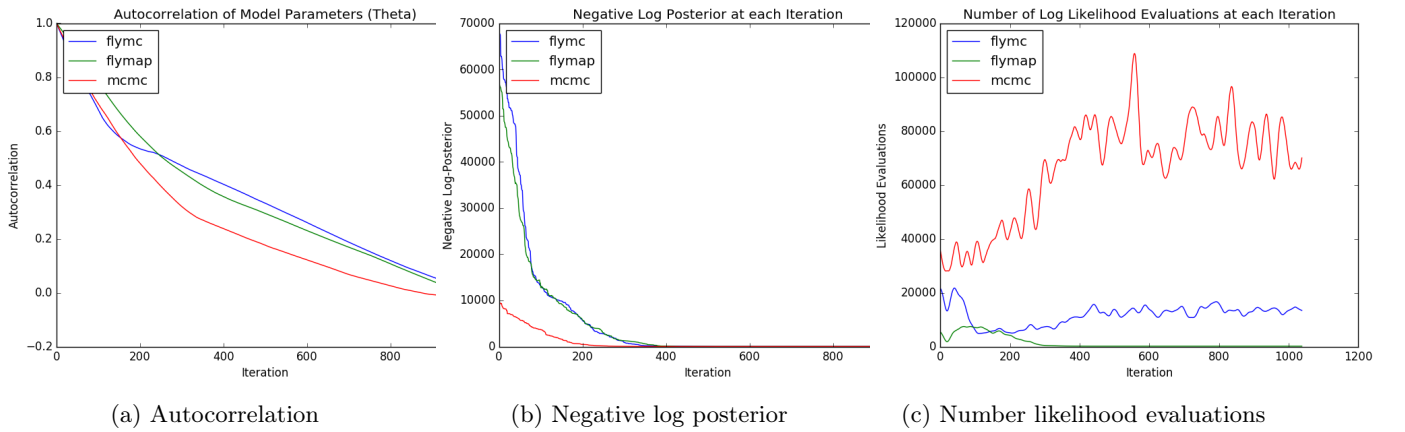


Figure 4: Robust regression on synthesized data with slice sampler

## Conclusion

TODO: Write this section, make sure the mention extension work including: minimizing bound over posterior samples instead of MAP tuning (see “MAP-tuning the lower bound”)

**You should end by critically evaluating your work and suggest what you would have done given more time.**

FlyMC achieves speed-ups over typical MCMC algorithms by estimating a large data-set’s likelihood using smaller subsets. To work, FlyMC requires a non-negative function  $B_n(\theta)$  that acts as a lower bound on the likelihood,  $B_n(\theta) \leq L_n(\theta) = P(\mathbf{x}_n | \theta)$ . In addition, FlyMC:

- Motivations for why we should be interested in FlyMC

We

Main Body: -Mathematical motivations for FLYMC as well as implementations

Results: Graphs Generated.

Conclusion:

## References

[MA14] Dougal Maclaurin and Ryan P Adams. Firefly monte carlo: Exact mcmc with subsets of data. *arXiv preprint arXiv:1403.5693*, 2014.