Sequential Monte Carlo for Approximate Variable Selection in Generalized Linear Models

Maxim Fedotov & Antoine Carnec

Supervisors:
David Rossell and Jack Jewson

Abstract

In the context of high-dimensional generalized linear regression, model selection methods must use heuristics or approximations to explore an exponentially large search space in an efficient manner. The classical Laplace Approximation (LA) is used to calculate an integrated likelihood which is used in Bayes factors to compare models. It has been shown to be model-selection consistent with a fast convergence rate with respect to the number of observations (Rossell and Rubio, 2019; Kass, 1990). Nevertheless, it quickly becomes computationally infeasible, since an optimization problem has to be solved for every possible model that is being explored. The Approximate Laplace Approximation (ALA) is a computationally cheap alternative, which is obtained via a quadratic log-likelihood expansion at an appropriate initial guess on the model parameters. However, it does not recover the same model as the LA asymptotically. We outline a Sequential Monte Carlo (SMC) algorithm that aims to quickly sample models from the LA posterior, using the ALA to get a starting distribution and subsequently applying it to sampled models. Our SMC algorithm exploits the fact that the ALA makes it cheaper to compute Bayes factors, and that the samples start to concentrate on a set of most likely models relatively quickly. We provide an implementation of the algorithm in a Python package `alasmc`.

1 Introduction

A data analyst must fit a model to their data to gain insights from it or make predictions. However, even within a restricted model class, searching through a model space can be exponentially difficult. In this paper we focus on regression models, where for p covariates the number of possible models is 2^p. A principled data-driven procedure that incorporates uncertainty is available using the general framework of Bayesian Inference. We focus on Bayesian Variable Selection in the context of the Generalized Linear Models using so-called Spike and Slap priors. This method has several advantages over non-probabilistic methods. It lets the data analyst ask flexible questions about the significance of variables through considering posterior model probabilities. This framework also allows the data analyst to encode prior information they have about the possible models they are willing to consider.

The cost of this Bayesian approach is in the computing the posterior itself, which boils down to computing a high dimensional integral. The problem of solving these integrals or more generally approximating posterior distributions is tackled by the field of Bayesian Computation. This is the area that this paper is attempting to contribute towards.

We focus on developing a Sequential Monte Carlo computation algorithm to sample binary vectors that indicate variable inclusion in Generalized Linear Regression by utilizing a recently developed computational strategy—the Approximate Laplace Approximation.

The Laplace Approximation (LA) and ALA Rossell et al. (2021) are two methods which both allow to approximate the model posterior. They involve taking a quadratic approximation of an integral around a point, any arbitrary point for the ALA, and the MLE for the LA. The ALA is much cheaper to compute than the LA, but is a worse (although still effective in a precise sense) model selection tool. The ALA can often identify active covariates by itself, but crucially it does not consistently recover the LA-optimal model.

The main idea of this paper is to somehow sample from the LA posterior (which is expensive to sample from) by constructing a sequence of ALA posteriors that converge to the LA posterior. The ideal scenario here is that we can take advantage of the cheap cost of the ALA, while also using the fact that the ALA distributed models are not too distinct from the LA distributed models (owing to the fact that ALA is still good at selecting active variables). The reward we get is a sample from the LA posterior, which has very good model selection properties.

The Sequential Monte Carlo algorithm allows us to perform sampling along a sequence of distributions $\pi_0, \ldots, \pi_T$ by using importance sampling, resampling and MCMC. The algorithm samples a first set of so-called *particles* from π_0 and uses importance sampling to move the set of particles along the sequence of distributions using resampling and MCMC to preserve the accuracy of the approximation.

We perform experiments on simulated data, focusing on the following elements: First, we want to get an understanding of how well the ALA SMC estimates of the posterior model and covariate inclusion probabilities approximate the exact LA posterior. Second, would like to see how does the ALASMC scale with the number of observations n, and the number of covariates p. Third, what are the computational-time gains of ALA SMC with respect to the full enumeration LA.

We find that the difference between the posterior probabilities of covariates inclusion are centered and concentrated at zero—although with some outliers. We observe that our ALASMC marginal inclusion probabilities successfully concentrate around the true LA values. The algorithm is fairly scalable, and the elapsed time depends on the number of models we sample at each iteration of SMC, the way we propose and accept new models (Markov kernel) and the initial point of the SMC algorithm. The algorithm balances between precision of approximation and computational gains with respect to the LA depending on the specified parameters.

1.1 A description of the coming sections

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3,4,5 and 6 summarize the necessary technical concepts that are the building blocks of the algorithm we propose in this paper. In order they are: Bayesian Variable Selection; Generalized Linear Models; the ALA and Sequential Monte Carlo Samplers. Section 7 outlines the definition of our algorithm, as well as implementation details and a discussion on the scalability of the algorithm. Section 8 outlines the results of our simulated experiments. Section 9 discusses ideas we had but did not have time to flesh out and ideas for future work based on current literature.

2 Related Work

SMC Samplers were introduced by Del Moral et al. (2006). They have been applied in Approximate Bayesian Computation rare-event simulation (Del Moral et al., 2012); as well as serving as a general inference tool in probabilistic programming engines Rainforth (2017).

There are a few advantages to SMC sampler over standard MCMC procedues. As we discuss in section 9, properly tuned SMC sampler can avoid the curse of dimensionality that afflicts other Monte Carlo methods. As another boon, SMC tends to be more robust to multimodality than their MCMC counterparts Paulin et al. (2019).

Since they have have access to a particle approximation at every iteration, there are many tricks that can be played to make SMC samplers adaptive. For instance Buchholz et al. (2021) show how to set Hamiltonian Monte Carlo hyperparameters adaptively during a run of SMC. Another relative advantage of SMC algorithms is that they are quite easily parallelisable. In fact, this parallelism can scale up quite easily; for example dividing up particle-specific computations across cores. See (Vergé et al., 2015; Guldas et al., 2015) for related work.

The closest paper to our work is Schäfer and Chopin (2013). They also tackle the problem of variable selection and they introduced the logistic kernel strategy that we use (and introduce in section 7.2). Specifically, they focus on the problem of exact inference on model posteriors in the context of a normally distributed likelihood and coefficient prior. Their approach is to sample directly from the model posterior, using the uniform distribution over p-dimensional binary vectors as the starting distribution and defining $\pi_t = \pi_0^{1-\lambda_t}\pi_T^{\lambda_t}$, where $\lambda_t \in [0, 1]$. In contrast, we do not focus on exact inference (rather we sample from the LA model posterior) but our approach allows us to consider the GLM setting, and we have no constraints on the coefficient or model prior.

The Approximate Laplace Approximation (ALA) is a key component of our SMC algorithm and was recently introduced by Rossell et al. (2021). We introduce it in section 5. The ALA does not consistently estimate the integrated likelihood unlike the LA. Nonetheless, the ALA is shown to have strong model selection properties—despite the fact that it does not approximate the integrated likelihood well. The ALA-optimal model that the ALA consistently estimates is generally different from the truly optimal model

(the most likely model under $p(\boldsymbol{\gamma} \mid \mathbf{y})$). However, they can coincide under some technical conditions and the ALA is reported to perform well in numerical experiments.

3 Bayesian Variable Selection

Let $\gamma_j = \mathbb{1}(\beta_j \neq 0)$, then $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p) \in \{0,1\}^p$ defines a model, i.e. it identifies which covariates are active. Let $p_\gamma := \sum_{j=1}^{p} \gamma_j$ be a number of non-zero parameters in model γ, and β_γ be a subset of coefficients that correspond to model $\boldsymbol{\gamma}$, i.e. $\boldsymbol{\beta}_{\boldsymbol{\gamma}} \in \mathbb{R}^{p_\gamma}$.

The idea of Bayesian Variable Selection is to compute posterior model probabilities $p(\boldsymbol{\gamma} \mid \mathbf{y})$, which gives the analyst information about the most probable model, as well as variable specific inclusion probabilities.

$$p(\boldsymbol{\gamma} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \boldsymbol{\gamma}) p(\boldsymbol{\gamma})}{\sum\limits_{\boldsymbol{\gamma}} p(\mathbf{y} \mid \boldsymbol{\gamma}) p(\boldsymbol{\gamma})} \propto p(\mathbf{y} \mid \boldsymbol{\gamma}) p(\boldsymbol{\gamma}) \tag{1}$$

The computational bottleneck in (1) is the integrated likelihood:

$$p(\mathbf{y} \mid \boldsymbol{\gamma}) = \int\limits_{\boldsymbol{\beta}_{\boldsymbol{\gamma}}} p(\mathbf{y} \mid \boldsymbol{\beta}_{\boldsymbol{\gamma}}, \boldsymbol{\gamma}) p(\boldsymbol{\beta}_{\boldsymbol{\gamma}} \mid \boldsymbol{\gamma}) d\boldsymbol{\beta}_{\boldsymbol{\gamma}} \tag{2}$$

If computed in a straightforward way, it can be the most time-consuming part of the Bayesian model selection approach together with the need to compute it for a lot of models. There are different approximations for these integrals. In this paper, we use the Approximate Laplace Approximation (ALA) introduced in Rossell et al. (2021).

To compute posterior model probabilities, one must introduce a prior distribution for a model, $p(\boldsymbol{\gamma})$, and coefficients given a model, $p(\boldsymbol{\beta}_{\boldsymbol{\gamma}} \mid \boldsymbol{\gamma})$.

Every gamma indicates a possible regression model since

$$P(\gamma_j = 1) = P(\beta_j \neq 0).$$

3.1 Prior Distribution

As an example, in this paper we use Spike-and-Slab type of a prior with a Beta-Binomial prior on models. The prior distribution on coefficients is:

$$p(\boldsymbol{\beta}_{\boldsymbol{\gamma}} \mid \boldsymbol{\gamma}) = \prod_{\{j:\ \gamma_j = 1\}} \mathcal{N}(\beta_j; 0, g\sigma^2)$$

and

$$p(\boldsymbol{\gamma}) = \text{Beta-Binomial}(1,1) = p\left(|\gamma|_0\right) p\left(\gamma \mid\ |\gamma|_0\right) = \frac{1}{d+1} \frac{1}{\begin{pmatrix} d \\ |\gamma|_0 \end{pmatrix}}$$

The Beta-Binomial prior above gives the same probabilities to models of a same size. It is preferred in is believed to work slightly better

We can then define a model-specific likelihood by

$$p(\mathbf{y} \mid \boldsymbol{\beta}_{\boldsymbol{\gamma}}, \boldsymbol{\gamma}) = \prod_{i=1}^{n} p(y_i \mid \boldsymbol{\beta}_{\boldsymbol{\gamma}}, \boldsymbol{\gamma})$$

4 Generalized Linear Models

In this paper we focus on *generalized linear models* (GLMs). Let $Y_1, \ldots, Y_n$ be independent random variables with observed values $\mathbf{y} = (y_1, \ldots, y_n)$. In addition, let $\mathbf{x}_i \in \mathbb{R}^p$ be the observed set of covariates of i-th sample. Denote the corresponding non-random design matrix as $X \in \mathbb{R}^{n \times p}$, and a parameter vector as $\boldsymbol{\beta} \in \mathbb{R}^p$.

The generalized linear regression model is then given by the following expression:

$$h(\mathbb{E}\left[Y_i \mid \mathbf{x}_i\right]) = \mathbf{x}_i^T \boldsymbol{\beta}$$

with canonical link function h.

The corresponding likelihood for model $\boldsymbol{\gamma}$ is

$$\begin{aligned} p(\mathbf{y} \mid \boldsymbol{\beta}_{\boldsymbol{\gamma}}, \boldsymbol{\gamma}) &= \prod_{i=1}^{n} \exp\left\{ y_i \boldsymbol{\eta}_{i,\boldsymbol{\gamma}} - b\left(\boldsymbol{\eta}_{i,\boldsymbol{\gamma}}\right) + c\left(y_i\right) \right\} \\ &= \exp\left\{ \mathbf{y}^T X_{\boldsymbol{\gamma}} \boldsymbol{\beta}_{\boldsymbol{\gamma}} - \sum_{i=1}^{n} b\left(\boldsymbol{\eta}_{i,\boldsymbol{\gamma}}\right) + \sum_{i=1}^{n} c\left(y_i\right) \right\} \end{aligned} \tag{3}$$

where $\boldsymbol{\eta}_{i,\boldsymbol{\gamma}} = \mathbf{x}_{i,\boldsymbol{\gamma}}^T \boldsymbol{\beta}_{\boldsymbol{\gamma}}$ is the corresponding linear predictor, with $b(\boldsymbol{\eta})$ being infinitely differentiable. Note that we omit conditioning on the covariates for clarity, and we exclude a dispersion parameter or multiplier which is typically presented in GLM from consideration for simplicity.

In this paper we will use various optimization methods, including Newton-Raphson, to explore posterior model probabilities. We need to retrieve the form of gradient and Hessian of the GLM negative log-likelihood. Denote $l(\mathbf{y} \mid \boldsymbol{\beta}_{\gamma}, \gamma) := -\log p(\mathbf{y} \mid \boldsymbol{\beta}_{\gamma}, \gamma)$. From (3) we obtain directly that the gradient and the Hessian have the following form:

$$g_{\gamma}(\boldsymbol{\beta}_{\gamma}) = \nabla_{\boldsymbol{\beta}_{\gamma}} l(\mathbf{y} \mid \boldsymbol{\beta}_{\gamma}, \gamma) = -X_{\gamma}^T \mathbf{y} + \sum_{i=1}^{n} b'(\mathbf{x}_{i,\gamma}^T \boldsymbol{\beta}_{\gamma}) \mathbf{x}_{i,\gamma} \tag{4}$$

$$H_{\gamma}(\boldsymbol{\beta}_{\gamma}) = \nabla_{\boldsymbol{\beta}_{\gamma}}^2 l(\mathbf{y} \mid \boldsymbol{\beta}_{\gamma}, \gamma) = \sum_{i=1}^{n} \mathbf{x}_{i,\gamma}^T b''(\mathbf{x}_{i,\gamma}^T \boldsymbol{\beta}_{\gamma}) \mathbf{x}_{i,\gamma} = X_{\boldsymbol{\gamma}}^T D(X_{\boldsymbol{\gamma}} \boldsymbol{\beta}_{\boldsymbol{\gamma}}) X_{\boldsymbol{\gamma}} \tag{5}$$

where $D(X_{\boldsymbol{\gamma}} \boldsymbol{\beta}_{\boldsymbol{\gamma}}) = \operatorname{diag}\{b''(\mathbf{x}_{1,\gamma}^T \boldsymbol{\beta}_{\gamma}), \ldots, b''(\mathbf{x}_{n,\gamma}^T \boldsymbol{\beta}_{\gamma})\}$. Note that sometimes we will write simply g_{γ} and H_{γ} for clarity.

Binomial Logistic Regression

As one of the examples, we consider binomial logistic regression. Let $Y_i \in \{0, 1\}$ be independent random variables for $i = 1, \ldots, n$, then binomial logit models $Y_i \sim Bern(\theta_i)$, where the link function is

$$\ln\left(\frac{\theta_i}{1-\theta_i}\right) = \mathbf{x}_i^T \boldsymbol{\beta} \tag{6}$$

Therefore, the model specific likelihood function can be written as follows:

$$p(\mathbf{y} \mid \boldsymbol{\beta}_{\boldsymbol{\gamma}}, \boldsymbol{\gamma}) = \exp\left\{ \mathbf{y}^T X_{\boldsymbol{\gamma}} \boldsymbol{\beta}_{\boldsymbol{\gamma}} - \sum_{i=1}^{n} \ln\left(1 + e^{\mathbf{x}_{i,\boldsymbol{\gamma}}^T \boldsymbol{\beta}_{\boldsymbol{\gamma}}}\right) \right\} \tag{7}$$

So, we have

$$\begin{aligned} b'(\mathbf{x}_{i,\gamma}^T \boldsymbol{\beta}_{\gamma}) &= \frac{1}{1 + e^{-\mathbf{x}_{i,\boldsymbol{\gamma}}^T \boldsymbol{\beta}_{\boldsymbol{\gamma}}}} \\ b''(\mathbf{x}_{i,\gamma}^T \boldsymbol{\beta}_{\gamma}) &= b'(\mathbf{x}_{i,\gamma}^T \boldsymbol{\beta}_{\gamma})(1 - b'(\mathbf{x}_{i,\gamma}^T \boldsymbol{\beta}_{\gamma})) \end{aligned}$$

Poisson Regression

Let $Y_i \in \mathbb{N}_0$, the Poisson regression assumes that $Y_i \sim \text{Poisson}(\mathbf{x}_i^T \boldsymbol{\beta})$. The model specific likelihood in this case is

$$p(\mathbf{y} \mid \boldsymbol{\beta}_{\boldsymbol{\gamma}}, \boldsymbol{\gamma}) = \exp\left\{ \mathbf{y}^T X \boldsymbol{\beta} - \sum_{i=1}^{n} e^{\mathbf{x}_i^T \boldsymbol{\beta}} - \log(y_i!) \right\}$$

where

$$b'(\mathbf{x}_{i,\gamma}^T \boldsymbol{\beta}_{\boldsymbol{\gamma}}) = b''(\mathbf{x}_{i,\boldsymbol{\gamma}} \boldsymbol{\beta}_{\boldsymbol{\gamma}}) = e^{\mathbf{x}_{i,\boldsymbol{\gamma}}^T \boldsymbol{\beta}_{\boldsymbol{\gamma}}}$$

The link function of the Poisson regression is:

$$\ln(\mathbb{E}(y_i \mid \mathbf{x}_i, \boldsymbol{\beta})) = \mathbf{x}_i^T \boldsymbol{\beta}$$

Note that b' and b'' are exponential for the Poisson regression. This means that a bad initial guess on the coefficients may lead to explosive gradient and hessian of the log likelihood when doing Newton-Raphson iterations. Not only that, such bad points can also occur during the run of the optimization procedure.

That is, Poisson regression suffers from a phenomenon of over-dispersion, which means that there is a significant mismatch between the observed variance and the true variance of the model – the former is higher that the latter one. Therefore, to cope with this problem one should consider a curvature adjustment that we discuss in subsection 5.3.

5 The ALA

5.1 The Laplace approximation

The Laplace Approximation (LA) is a analytical trick that is used to approximate difficult to compute integrals arising in Bayesian computations. We will be considering the ALA to the relevant integrated likelihood in our context—equation (2). Briefly, the LA consists in taking a second order Taylor expansion of $\log p(\mathbf{y}, \boldsymbol{\beta}_{\boldsymbol{\gamma}} \mid \boldsymbol{\gamma})$ around the MAP estimate $\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}} = \text{argmax}_{\boldsymbol{\beta}_{\boldsymbol{\gamma}}} p(\mathbf{y}, \boldsymbol{\beta}_{\boldsymbol{\gamma}} \mid \boldsymbol{\gamma})$.

Denote $\hat{p}(\boldsymbol{\gamma} \mid \mathbf{y}) \propto \hat{p}(\mathbf{y} \mid \boldsymbol{\gamma}) p(\boldsymbol{\gamma})$ as the LA model posterior, The resulting expression is

$$\hat{p}(\mathbf{y} \mid \boldsymbol{\gamma}) = p(\mathbf{y} \mid \hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}}, \boldsymbol{\gamma}) p(\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}} \mid \boldsymbol{\gamma}) 2\pi^{p_\gamma/2} |H_\gamma(\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}})|^{-\frac{1}{2}}.$$

where $H_\gamma(\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}})$ is a hessian of the negative log-integrand, $-\log p(\mathbf{y}, \boldsymbol{\beta}_{\boldsymbol{\gamma}} \mid \boldsymbol{\gamma})$, estimated at the MLE for model γ. Laplace approximations have very nice model selection properties in our context. They have been proved to consistently estimate the integrated likelihood and the resulting Bayes Factors converge at a rate of $O(n^{-2})$ under some technical conditions (Kass, 1990; Rossell et al., 2021). These strong model selection properties motivate the goal that our proposed algorithm aims for, which is to sample from the LA approximation of $p(\boldsymbol{\gamma} \mid \mathbf{y})$.

The key computational bottleneck however is that if we want to evaluate $\hat{p}(\mathbf{y} \mid \boldsymbol{\gamma})$, we must solve an optimization problem to find $\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}}$. So we must solve an optimization problem every time we want to evaluate the likelihood of a model.

5.2 The Approximate Laplace Approximation

The Approximate Laplace Approximation (henceforth ALA), first introduced by Rossell et al. (2021), is a cheap alternative to the Laplace approximation, where instead of taking an approximation around the MAP, the approximation is done around some other point $\boldsymbol{\beta}_{\boldsymbol{\gamma}0}$.

In this context of generalized linear models, Rossell et al. (2021) show that the case where $\boldsymbol{\beta}_{\boldsymbol{\gamma}0} = \mathbf{0}$, the ALA approximation to the integral in (2) can be computed in 'one go'—in the sense that there are no heavy model specific computations that need to be done. This makes ALA a computationally cheap option for inference. After precomputing some statistics (the gradient and hessian) on the full model, any subsequent model-specific evaluations of the posterior are quick to compute. In our SMC algorithm (which we introduce in Section 7), we can take advantage of this when $\boldsymbol{\beta}_{\boldsymbol{\gamma}} = \mathbf{0}$, which allows us to sample our initial set of particles relatively quickly using a simple Gibbs Sampler (see section 6.4.1).

In our setting, we are interested in approximating $p(\boldsymbol{\gamma} \mid \mathbf{y})$ with the ALA. More specifically, we approximate the integrated likelihood 2 with the ALA model evidence $\tilde{p}(\mathbf{y} \mid \boldsymbol{\gamma})$, which means we can define

$$\tilde{p}(\boldsymbol{\gamma} \mid \mathbf{y}) = \frac{1}{Z} \tilde{p}(\mathbf{y} \mid \boldsymbol{\gamma}) p(\boldsymbol{\gamma})$$

where Z is a normalising constant.

Let $\boldsymbol{\beta}_{\boldsymbol{\gamma 0}}$ be an initial point at which the approximation is done for model $\boldsymbol{\gamma}$. The one-step estimator based on Newton-Raphson method is then $\tilde{\boldsymbol{\beta}}_{\boldsymbol{\gamma}} = \boldsymbol{\beta}_{\boldsymbol{\gamma 0}} - H_{\gamma 0}^{-1} g_{\gamma 0}$. Then

$$\tilde{p}(\mathbf{y} \mid \boldsymbol{\gamma}) = p(\mathbf{y} \mid \boldsymbol{\beta}_{\boldsymbol{\gamma 0}}, \boldsymbol{\gamma}) p(\tilde{\boldsymbol{\beta}}_{\boldsymbol{\gamma}} \mid \boldsymbol{\gamma}) (2\pi)^{\frac{p_{\boldsymbol{\gamma}}}{2}} |H_{\gamma 0}|^{-\frac{1}{2}} \exp\left\{ \frac{1}{2} g_{\gamma 0}^T H_{\gamma 0}^{-1} g_{\gamma 0} \right\}. \tag{8}$$

where $g_{\gamma 0}$ and $H_{\gamma 0}$ are the gradient and the hessian of a negative log-likelihood, $-\log p(\mathbf{y} \mid \boldsymbol{\beta}_{\boldsymbol{\gamma}}, \boldsymbol{\gamma})$.

Note, that it is possible to use a common non-zero starting point $\boldsymbol{\beta}_{\mathbf{0}}$, and set $\boldsymbol{\beta}_{\boldsymbol{\gamma 0}} = \left[\boldsymbol{\beta}_{\mathbf{0},\boldsymbol{j}}\right]_{\gamma_j = 1}$ to be an initial point for model $\boldsymbol{\gamma}$, where $\boldsymbol{\beta}_{\mathbf{0},\boldsymbol{j}}$ is j-th element in $\boldsymbol{\beta}_{\mathbf{0}}$.

5.3 Curvature Adjustment

The curvature adjustment is crucial for improving the performance of the ALA for finite samples (Rossell et al., 2021). It sometimes happens that an observed variance of y and the one that predicted by a model γ do not match. This problem is more common for some models (like Poisson regression) and can also occur at specific points β_γ. In this case, we can end up with an intractable hessian at β_γ.

The hessian computed in ALA is:

$$H_{\boldsymbol{\gamma 0}} = X_{\boldsymbol{\gamma}}^T \cdot \operatorname{diag}\{b''(\mathbf{x}_{1,\gamma}^T \boldsymbol{\beta}_\gamma), \ldots, b''(\mathbf{x}_{n,\gamma}^T \boldsymbol{\beta}_\gamma)\} \cdot X_{\boldsymbol{\gamma}} = X_{\boldsymbol{\gamma}}^T \operatorname{Cov}\left(y \mid X_{\boldsymbol{\gamma}}, \boldsymbol{\beta} = \boldsymbol{\beta}_{\boldsymbol{\gamma 0}}\right) X_{\boldsymbol{\gamma}}$$

Suppose that the data is truly generated by some distribution F^*, which can (or can not) correspond to a point β^* in the model space we are searching through. The cause of the problem is that the estimated mean, $\mu_{\gamma 0} = \mathbb{E}[y \mid Z_\gamma, \beta = \beta_{\gamma 0}]$, does not match the true mean, $\mathbb{E}[y \mid Z_\gamma, \beta = \beta_\gamma^*]$. So, it can impose a significant difference between the model predicted covariance $\operatorname{Cov}\left(y \mid Z_\gamma, \beta = \beta_{\gamma 0}, \phi\right)$ and $\mathbb{E}_{F^*}[(y - \mu_{\gamma 0})^T (y - \mu_{\gamma 0}) \mid Z_\gamma]$.

One of the ways to cope with it is to multiply the model estimated covariance by the over-dispersion estimator:

$$\hat{\rho} = \frac{\sum_{i=1}^n (y_i - h^{-1}(\mathbf{x}_i^T \boldsymbol{\beta}_{\boldsymbol{\gamma 0}}))^2 / [b''(\mathbf{x}_i^T \boldsymbol{\beta}_{\boldsymbol{\gamma 0}})]}{n - \left|\boldsymbol{\beta}_{\boldsymbol{\gamma 0}}\right|_0}$$

where $|\cdot|_0$ is L-0 norm of a vector, and h^{-1} is the link function. So, the model specific covariance is replaced by $\hat{\rho} \operatorname{Cov}\left(y \mid X_\gamma, \beta = \beta_{\gamma 0}\right)$.

6 Sequential Monte-Carlo Samplers

Sequential Monte Carlo samplers are a class of algorithms that aim to sample from a desired distribution using a specific sequential strategy. The goal of the strategy is to define a sequence of distributions $\pi_0,\ \pi_1,\ \ldots,\ \pi_T$ such that π_T is the distribution you wish to sample from or the one that converges to it.

The first step is choosing the initial distribution, π_0, that is relatively easy to sample from but ideally is not really far from the target distribution, and move that sample

along the sequence of distributions using a technique which allows you to sample from π_t given π_{t-1}. More specifically, The algorithm boils down to doing importance sampling on your sample at every step, then resampling and applying a Markov Kernel to diversify the set of particles—we will introduce these notions in the next subsections.

In this section, we outline the building blocks of Sequential Monte Carlo algorithms before we introduce the main algorithm we develop in this thesis. The reader already familiar with SMC can safely skip this section. The main concepts to know are *importance sampling* (how we sample from π_{t-1} to π_t), *effective sample size* (how we understand how good our current sample is), *resampling* (a technique to increase diversity in our sample) and *Metropolis-Hastings kernels* (given an unnormalized density, how do we sample from the corresponding distribution).

6.1 Normalized Importance Sampling

So we have in mind that given a sample from π_{t-1}, we want to sample from a distribution π_t. The important concept to understand that makes SMC methods work is *importance sampling*. In general, Importance Sampling allows one to sample from some distribution p, under the following conditions:

1. You have access the probability density function of p up to a normalizing constant.
2. You can access to the probability density function of some other distribution q up to a normalizing constant.
3. You can sample from q.
4. The support of q is a superset of the support of p.

The idea of importance sampling is that if you sample from q and appropriately weight the resulting sample, you can recover a genuine sample from p. Define

$$w(x) = \frac{\tilde{p}(x)}{\tilde{q}(x)}$$

where $\tilde{p}$ and $\tilde{q}$ are the unnormalized densities of the p and q distributions[1].

Then take any function $\varphi : \mathbb{B} \to \mathbb{R}$ from the sample space (for example $\varphi(x) = \mathbb{1}(x \geq 0.8)$), then importance sampling allows one to recover a consistent (but biased) estimate of $\mathbb{E}_p\left[\varphi(\cdot)\right]$.

$$\frac{\sum_{i=1}^N w(x_i)\varphi(x_i)}{\sum_{i=1}^N w(x_i)} \xrightarrow{p} \mathbb{E}_p\left[\varphi(\cdot)\right], \quad x_i \sim q$$

Defining $W(x_j) = \frac{w(x_j)}{\sum_{n=1}^N w(x_n)}$, we equivalently have

$$\sum_{n=1}^N W(x_n)\varphi(x_n) \xrightarrow{p} \mathbb{E}_p\left[\varphi(\cdot)\right], \quad x_i \sim q \tag{9}$$

[1]We now see why it is required that the support of q is a superset of p.

One can motivate the above by utilizing the so-called *plug in principle* on the following identity (Chopin et al., 2020)

$$\begin{aligned}\int \varphi(x)p(x) &= \frac{\int \varphi(x)p(x)}{\int p(x)} \\ &= \frac{\int \varphi(x)\frac{\tilde{p}(x)}{\tilde{q}(x)}q(x)}{\int \frac{\tilde{p}(x)}{\tilde{q}(x)}q(x)} \\ &= \frac{\int \varphi(x)w(x)q(x)}{\int w(x)q(x)}. \end{aligned} \tag{10}$$

Chopin et al. (2020) discuss asymptotic and non-asymptotic properties of (10). In particular, they show that the MSE converges to 0 at rate $O(\frac{1}{N})$ when the φ function is bounded.

The take away from (9)is that this works for any φ. So one way we could think about this is that the weighted sample is an *approximate* sample of q, insofar as taking any function on the approximate sample will give a consistent estimate of the expectation of that function on the target distribution.

Obviously, the performance of this procedure is dependent on the proposal density q. To get an intuition why, consider the following diagram:

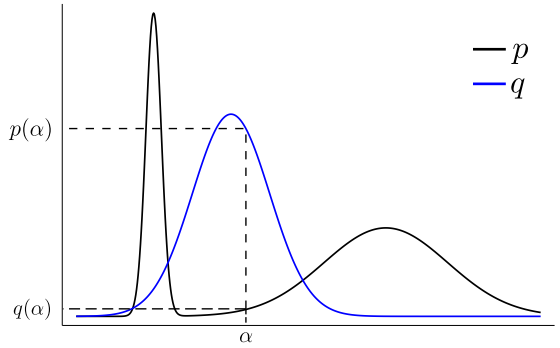


Figure 1: An illustration of a poorly performing proposal distribution q

We see that at point α, $W(\alpha)$ would be quite small. However, if another point β gets sampled near the first mode of the distribution, the weight $W(\beta)$ will be quite large since $q(\beta)$ is small. This increases the variance of the estimate of $\mathbb{E}_p\left[\varphi(\cdot)\right]$. Furthermore, we can see that the second mode of p will not be sufficiently explored since the proposal distribution q does not propose many points near the right-hand mode of the target distribution. In this case we can see that the q distribution is not a very good proposal distribution. The intuition we are trying to communicate here is that in general, the importance sampling procedure works well the closer the q distribution is to the p distribution.

6.2 Effective Sample Size

So we saw that if there is a big disparity between the proposal and target distribution, the variance of the weights (and the variance of the resulting estimator) will substantially increase. It is useful in SMC algorithms to monitor the variance in the weights we calculate while doing importance sampling. If we observe that one weight is very large, and the others are relatively low, then this indicates that our importance sampling procedure is performing poorly.

In light of this, a useful metric to quantify the weight degeneracy in the current sample is the Effective Sample Size (henceforth ESS):

$$\mathrm{ESS}(\mathbf{X}) = \frac{(\sum_{n=1}^{N} w(x_n))^2}{\sum_{n=1}^{N} w(x_n)^2} = \frac{1}{\sum_{n=1}^{N} W(x_n)^2}$$

The following properties of the ESS that are important to keep in mind:

1. The effective sample size lies in the $[1, N]$ interval. If $ESS = 1$, then we have our sample is concentrated on 1 point, whereas $\mathrm{ESS} = N$ indicates that all of our sample points are equally probable.

2. As $N \to \infty$, $\frac{N}{\mathrm{ESS}}$ converges to the chi-squared pseudo-distance between the proposal and target distribution (between π_t and π_{t+1} in the SMC algorithm). This cements the intuition that the ESS gives you a sense of how well the importance sampling step is performing.

Another useful (but distinct) measure of sample diversity—described in (Schäfer and Chopin, 2013)—is a sample measure that is unique to the discrete setting that we are operating under in this paper. It is simply defined as

$$\zeta(\mathbf{X}) = \frac{\#\{\mathbf{x}_n \mid n = 1, \ldots, N\}}{N}.$$

We use this particle diversity measure to adaptively choose the number of times we apply the MCMC transition kernel to each particle.

6.3 Resampling

The notions we have introduced thus far are sufficient to implement the idea of sampling sequentially along a sequence of distributions. However, this would not work well in practice, since the number of particles with non-negligible weight would fall very quickly—the approximation will center around one point. This means that the ESS would go to 1, and the variance of the resulting estimate on $\mathbb{E}_p[\varphi]$ would be high.

A solution to the following is to *resample*. Meaning that we draw an entirely new sample based on the normalized weights we have calculated at the present iteration. There are many strategies to do this efficiently, but we employ the most simple one in this paper. We simply resample by drawing from a Multinomial$(W_1^t, W_2^t, \ldots, W_N^t)$[2].

[2]It is worth noting that there are other ways of doing this, which we did not implement for the sake of time / simplicity. Consult (Douc and Cappé, 2005) for a comparison of the different resampling schemes available

However, this does not quite solve the problem, because we will end up with not many unique models in the end. Ideally, we would like a way to *diversify* our set of particles. We do this by applying an MCMC kernel that is invariant to the current target distribution π_t.

6.4 Markov Kernels & MCMC

Markov Chain Monte Carlo is a class of algorithms that allows one to sample from a distribution by constructing a chain of sampled points that eventually resemble iid draws from the target distribution. Specifically, associated to an MCMC procedure is a *transition kernel* M which allows you to sample the next point in the chain given the previous point. For the sake of brevity we do not outline the theory of Markov Chain Monte Carlo in this report. It is sufficient to state that under some technical conditions, MCMC procedures allow one to sample from a distribution where one only has access to the density function of this distribution.

With respect to the SMC algorithm, MCMC procedures allow us to *move* our particles at time t with some a transition kernel M_t such that after moving our system of particles, our system of particles still constitutes a sample of π_t. The intuition is that the diversify the current sample of particles, so that not all of the weight is centered on just a few points.

For our purposes, it is sufficient to define a *kernel* as a function that takes a point in $\{0,1\}^p$ and allows us to sample another point. Provide that the chain satisfies some conditions, this new point should still constitute a sample from the stationary distribution associated to the kernel.

6.4.1 Gibbs Sampler

We use a simple random scan gibbs sampler as a baseline to move the particles in our SMC algorithm. Notice that

$$p(\gamma_j \mid \boldsymbol{\gamma}_{-j}, \mathbf{y}) = \frac{p(\boldsymbol{\gamma}_{(j=1)} \mid \mathbf{y})}{p(\boldsymbol{\gamma}_{(j=1)} \mid \mathbf{y}) + p(\boldsymbol{\gamma}_{(j=0)} \mid \mathbf{y})}. \tag{11}$$

So one move of the resulting MCMC procedure takes a model $\boldsymbol{\gamma}$, uniformly samples a number $i \in \{1, \ldots, p\}$, and return $\boldsymbol{\gamma}$ with the i-th entry sampled from a Bernoulli with probability (11).

6.5 Bringing it all together: A generic SMC algorithm

Algorithm 1 gathers all of the above elements to give a conceptual overview of a generic SMC sampler.
Notation: N is the number of particles we have ; w_n^t indicating the unnormalized weight associated to particle n at time t ; W_n^t is the unnormalized weight ; $\boldsymbol{\gamma}_n^t$ is the nth particle at time t ;Γ_t is the set of particles at time t ; M_t is the markov kernel at time t.
Keep, there are a few implementation details that are hidden in the general description

of algorithm 1. We elaborate more on those details when we discuss our algorithm in section 7.

Algorithm 1: Basic SMC Sampler

Input: Distributions $\pi_0, \ldots, \pi_T$ where $\pi_t(x) = \frac{1}{L_t}\tilde{\pi}_t(x)$; Markov Kernel M_t

(n indicates that action is done for $n = 1, \ldots, N$)

$\boldsymbol{\gamma}_n^0 \sim \tilde{\pi}_0(\cdot)$

$w_n^0 \leftarrow \tilde{\pi}_0(\boldsymbol{\gamma}_n^0)$

$W_n^0 \leftarrow \frac{w_n^0}{\sum_{j=1}^n w_j^0}$

for $t = 1, \ldots, T$ **do**

  $\hat{\boldsymbol{\gamma}}_n^t \leftarrow$ resample $\boldsymbol{\gamma}_n^{t-1}$ with Multinomial$(W_1^{t-1}, W_2^{t-1}, \ldots, W_N^{t-1})$

  $\boldsymbol{\gamma}_n^t \sim M_t(\hat{\boldsymbol{\gamma}}_n^t)$

  $w_n^t \leftarrow \frac{\tilde{\pi}_t(\hat{\boldsymbol{\gamma}}_n^t)}{\tilde{\pi}_{t-1}(\hat{\boldsymbol{\gamma}}_n^t)}$

  $W_n^t \leftarrow \frac{w_n^t}{\sum_{j=1}^n w_j^t}$

end

When the algorithm terminates we should have a weighted sample $\{W_n^T, \boldsymbol{\gamma}_n^T\}_{n=1}^N$. We can then consistently estimate the expectation of an arbitrary function φ with

$$\mathbb{E}_{\pi_T}[\varphi] \approx \sum_{n=1}^{N} W_n^T \varphi(\boldsymbol{\gamma}_n^T).$$

In our context, an important case is $\varphi(x) = \mathbb{1}(\boldsymbol{\gamma}_i = 1) = \boldsymbol{\gamma}_i$, in which case we can recover the posterior inclusion probability for variable i

$$P(\gamma_i \mid \mathbf{y}) = \sum_{n=1}^{N} W_n^T \boldsymbol{\gamma}_{n,i}^T.$$

Convergence and CLT type results can be derived for this estimator. We refer the interested reader to (Chopin et al., 2020, §11).

7 ALA Meets SMC

The following observation inspires our Sequential Monte Carlo approach. The ALA and the LA only differ conceptually insofar as they are approximations taken around two different points (The ALA at an arbitrary point, and the LA at the MAP or MLE). So if we define a sequence of points $\beta_0, \ldots, \beta_T$ such that β_T is the MLE, then this implicitly define a sequence of distributions from any ALA approximation — at an initial guess β_0 — to the LA approximation at β_T.

More concretely, let $\tilde{p}_{t+1}(\boldsymbol{\gamma} \mid \mathbf{y}) \propto \tilde{p}_{t+1}(\mathbf{y} \mid \boldsymbol{\gamma})p(\boldsymbol{\gamma})$, where $\tilde{p}_{t+1}(\mathbf{y} \mid \boldsymbol{\gamma})$ is the ALA approximation at $\beta_{t\gamma}$, where $\beta_{t\gamma}$ is the t-th iteration of the Newton-Raphson method to solve the problem

$$\max_{\beta_\gamma} \; p(\mathbf{y} \mid \boldsymbol{\beta}_{\boldsymbol{\gamma}}, \boldsymbol{\gamma}). \tag{12}$$

That is, the sequence $\left\{\tilde{p}_t(\mathbf{y} \mid \boldsymbol{\gamma})\right\}_{t=1}^{T}$ is defined by ALA using $\beta_{t\gamma}$, $g_\gamma(\beta_{t\gamma})$, $H_\gamma(\beta_{t\gamma})$ (see equation 8). The reader will recall that GLM likelihood are concave in general, and strictly concave if the X matrix has full column rank. This ensures that β_T is uniquely defined.

So our sequence of distributions $\pi_0, \ldots, \pi_T$ will be learnt *adaptively*, and will simply be $\tilde{p}_1(\boldsymbol{\gamma} \mid \mathbf{y}), \ldots, \tilde{p}_T(\boldsymbol{\gamma} \mid \mathbf{y})$. Specifically we will have that π_t is the ALA model evidence after t steps of an optimization procedure on (12).

Note that this strategy implies that we are solving—at least partially—an optimization problem for every model that we encounter during our SMC algorithm. We are already doing better than if we were sampling from the LA, since this would involve *fully* solving an optimization problem for every model we encounter.

We outline the pseudo-code for the algorithm in Algorithm 2.

Algorithm 2: ALASMC Sampler

Input: Markov Kernel M_t, convergence threshold ε

(n indicates that action is done for $n = 1, \ldots, N$)

$\boldsymbol{\gamma}_n^0 \sim \tilde{p}_0(\cdot \mid \mathbf{y})$

$w_n^0 \leftarrow 1$

$W_n^0 \leftarrow \frac{1}{N}$

$t \leftarrow 1$

while $\left| \frac{\tilde{p}_t(\mathbf{y} \mid \boldsymbol{\gamma}_n^t) p(\boldsymbol{\gamma}_n^t)}{\tilde{p}_{t-1}(\mathbf{y} \mid \boldsymbol{\gamma}_n^{t-1}) p(\boldsymbol{\gamma}_n^{t-1})} - 1 \right| \geq \varepsilon$ **do**

  if $ESS(\Gamma_{t-1}) < ESS_{min}$ **then**

    $\hat{\boldsymbol{\gamma}}_n^t \leftarrow$ resample $\boldsymbol{\gamma}_n^{t-1}$ with Multinomial$(W_1^{t-1}, W_2^{t-1}, \ldots, W_N^{t-1})$

    $\hat{w}_n^{t-1} \leftarrow 1$

  else

    $\hat{\boldsymbol{\gamma}}_n^t \leftarrow \boldsymbol{\gamma}_n^{t-1}$

    $\hat{w}_n^{t-1} \leftarrow w_n^{t-1}$

  end

  $\boldsymbol{\gamma}_n^t \sim M_t(\hat{\boldsymbol{\gamma}}_n^t)$

  $w_n^t \leftarrow \hat{w}_n^{t-1} \frac{\tilde{p}_t(\mathbf{y} \mid \boldsymbol{\gamma}_n^t)}{\tilde{p}_{t-1}(\mathbf{y} \mid \boldsymbol{\gamma}_n^t)}$

  $W_n^t \leftarrow \frac{w_n^t}{\sum_{j=1}^n w_j^t}$

  $t \leftarrow t + 1$

end

We only resample if the ESS falls below a certain pre-specified threshold. A common choice in the literature for this value is $\frac{N}{2}$. We now explain how we implemented the various parts of this algorithm. To sample from $\tilde{\pi}_0$ we used the random-scan Gibbs Kernel we describe in 6.4.1. We set a burn-in parameter B to allow the Markov Chain to mix. For the Markov Kernels, we used again random-scan Gibbs and we also try the logistic kernel introduced in (Schäfer and Chopin, 2013) (which we briefly describe in section 7.2). We must specify how to move the particles when applying the Markov kernel M_t. We could a priori set some integer k of MCMC steps to apply to every particle. However, the quality of the resulting sample approximation of the posterior suffers if k is too small. We follow the technique outlined in Schäfer and Chopin (2013) Procedure 4 to adaptively set k, where they calculate the particle diversity (explained in section 6.2) after every particle has been moved once, and terminate if the particle diversity did not change too much.

Conceptually the algorithm terminates where the optimization problem (12) terminates for every model $\boldsymbol{\gamma}$ in our particle system. We use the following strategy to assess

the convergence of the algorithm. If

$$\left| \frac{\tilde{p}_{t+1}(\mathbf{y} \mid \boldsymbol{\gamma}_n^{t+1}) p(\boldsymbol{\gamma}_n^{t+1})}{\tilde{p}_t(\mathbf{y} \mid \boldsymbol{\gamma}_n^t) p(\boldsymbol{\gamma}_n^t)} - 1 \right| < \varepsilon$$

for $n = 1, \ldots, N$ for a suitably small epsilon, then the algorithm terminates. In our implementation we use a similar, but computationally more tractable stopping criterion, that is – the algorithm stops when

$$\left| \ln \tilde{p}_{t+1}(\mathbf{y} \mid \boldsymbol{\gamma}_n^{t+1}) + \ln p(\boldsymbol{\gamma}_n^{t+1}) - \ln \tilde{p}_t(\mathbf{y} \mid \boldsymbol{\gamma}_n^t) - \ln p(\boldsymbol{\gamma}_n^t) \right| < \varepsilon_{loglike}$$

When this condition is satisfied, it means that model-specific likelihood did not change much from t to $t+1$. Unless the likelihood is extremely flat, this will mean that the model-specific optimization problems are practically solved.

Let us say a few words on why this algorithm might seem like a good (or bad) idea a priori. Recall that our algorithm aims to sample from the LA posterior $\hat{p}(\boldsymbol{\gamma} \mid \mathbf{y})$. A straightforward way to do so would be to run an MCMC procedure to sample directly from $\hat{p}(\boldsymbol{\gamma} \mid \mathbf{y})$. However, this would involve solving a full optimization problem for every model we encounter in our Markov Chain. Our algorithm on the other hand, avoids having to fully solve an optimization problem for every single model we encounter. We simply run t iterations of an optimization procedure if we encounter a model at iteration t.

The main hope for the algorithm is that $\{\boldsymbol{\gamma}_n^0\}_{n=1}^N$ already contains many models that are supersets of the true model $\boldsymbol{\gamma}^*$. Rossell et al. (2021) show that in general the ALA does not recover the same optimal model as LA. Therefore, if the initial ALA sample misses some active variables, then the algorithm relies on the MCMC step within the SMC procedure to find models with the missing active variables.

While this algorithm may still seem prohibitively expensive to be scalable, we make the following two points: Since the idea is that ALA concentrates on certain variables immediately, we hope to be seeing the same models many times, meaning we can save CPU time by saving computations and having a smaller number of unique models to evaluate. Second, in the case of a truly sparse model most particles will have p_γ be relatively low too—hence making the Hessian computations cheaper.

7.1 The Starting Coefficients

The performance of algorithm 2 is dependent on its initialisation. There is a trade-off here. Starting at a closer point to the model-specific MLE will mean that the algorithm will converge faster—but may mean that the initial distribution $\tilde{p}_0$ is too narrow. Too narrow in the sense that some active variables could be omitted from the sample. Starting at a farther away point means the algorithm will take longer to converge. However, in the case of starting at $\mathbf{0}$ we can benefit from ALA precomputation (as discussed in section 5).

For example, a 'close' initial guess could be the MLE under the full model when it exists and is tractable. That is, we can set

$$\boldsymbol{\beta}_0 = \arg\max_{\boldsymbol{\beta}_\gamma} p(\boldsymbol{y} \mid \boldsymbol{\gamma} = \mathbb{1}_p, \boldsymbol{\beta}_{\boldsymbol{\gamma}})$$

when the MLE exists and is unique; the ratio n/p is not too small and other technical conditions hold, e.g. at the local maximum points the hessian is not singular. This initial point is going to be common in a sense that for each model γ we can use a projection of it to the subset of the model space where γ lies, e.g.

$$\boldsymbol{\beta}_{0\boldsymbol{\gamma}} = [\boldsymbol{\beta}_0]_\gamma \in \mathbb{R}^{p_\gamma}.$$

It is worth mentioning that when the MLE is intractable and does not exist, one can use the penalized likelihood estimates. For example, one can set

$$\boldsymbol{\beta}_0 = \arg\max_{\boldsymbol{\beta}_{\boldsymbol{\gamma}}} \left\{ \ln p(\boldsymbol{y} \mid \boldsymbol{\gamma} = \mathbb{1}_p, \boldsymbol{\beta}_{\boldsymbol{\gamma}}) + \frac{\alpha}{2} ||\boldsymbol{\beta}_{\boldsymbol{\gamma}}||_2^2 \right\},$$

where $||\cdot||_2$ denotes the Euclidean norm. In this case the performance is going to depend on the parameters of said penalization, e.g α for the example above.

7.2 Designing an efficient Markov Kernel

For the Markov Kernel M_t, we can defer to the work of Schäfer and Chopin (2013), who motivate and develop a strategy to construct a fast-mixing kernel. They note that 'local' Metropolis Hastings kernels (i.e proposals that are conditional on the current state) do not mix fast enough. They propose an *independent* metropolis-hastings kernel, which must be fast to evaluate and must be relatively economical in it's number of parameters.

They show that an independent Metropolis-Hastings sampler q, would be faster mixing given that it approximates the target distribution well enough. At the stage of applying the MCMC steps inside the SMC loop, we have the benefit of having an empirical distribution for the current distribution π_t—the authors use this to design a parametric proposal distribution q_θ that aims to capture the cross moments of the empirical distribution of the particles. Briefly, their strategy is as follows: Suppose we have a parametric distribution $q_\theta(\boldsymbol{\gamma})$, we can decompose this as

$$q_\theta(\boldsymbol{\gamma}) = q_\theta(\gamma_1) \prod_{i=2}^{p} q_\theta(\gamma_i \mid \gamma_{1:i-1}).$$

The idea is to estimate the conditional q_θ distributions in reference to $\pi(\gamma_i \mid \gamma_{1:i-1})$—which we have access to through our system of particles. So set

$$q_\theta(\gamma_i \mid \gamma_{1:i-1}) = p_i^{\gamma_i}(1-p_i)^{1-\gamma_i};$$
$$p_i = \text{logistic}(b_{i,i} + \sum_{j=1}^{i-1} b_{i,j}\gamma_j)$$

.

The procedure would then be to simply estimate $q_\theta(\boldsymbol{\gamma})$ through a maximum likelihood optimization procedure when the markov kernel M_t must be produced. Schäfer and Chopin (2013) outline some tricks to make this work in practice, such as only estimating regression for those variables which do not have very high or very low marginal probabilities, and screening out regressor variables based on a sample correlation measure.

7.3 Python Package

We provide an implemention of Algorithm 2 written in Python. We implement all of the procedures described in this paper including the Logistic Kernel described in the previous section. The package has been designed to be modular and easily extendible. The code can be found at `https://github.com/antotocar34/masters_project`.

8 Synthetic Data

The purpose of the experiments is to: compare ALASMC and Laplace Approximation estimates of the posterior model probabilities, $\hat{p}(\gamma \mid \mathbf{y})$, and posterior probabilities of covariates inclusion, $\hat{p}(\gamma_j \mid \mathbf{y})$ for $j = 1, \ldots, p$; compare the elapsed time of producing these estimates via ALASMC and full enumeration of the model space with the Laplace Approximation.

8.1 Data Generation and Parameters

We discuss Logistic and Poisson regressions in section 4. To sample the design matrix, we first fix the number of observations n, and covariates p. Then, we sample $X^{2:p} \in \mathbb{R}^{n \times p-1}$,

$$X^{2:p} \sim \mathcal{N}(\vec{0}, \Sigma), \text{ where } \Sigma^{p \times p} : \ \Sigma_{ii} = 1, \ \Sigma_{ij} = \rho, \ i \neq j.$$

The resulting design matrix is $X = (\mathbb{1}_n, X^{2:p})$, that is, the first column corresponds to an intercept of the model. The parameter ρ corresponds to the strength of covariance among the features. The higher the covariance between an active feature and a spurious one is, the harder it gets to distinguish from what feature the real effect comes.

After sampling X, we are sampling $\mathbf{y}$ based on a specified true vector of coefficients[3], $\boldsymbol{\beta}^*$, according to the conditional distribution of a model, that is:

$$\textit{Logistic regression}$$
$$\theta_i = \frac{1}{1 + e^{-\mathbf{x}_i^T \boldsymbol{\beta}^*}}, \ y_i \mid \mathbf{x}_i, \ \boldsymbol{\beta}^* \sim \text{Bern}(\theta_i)$$
$$\textit{Poisson regression}$$
$$\lambda_i = e^{\mathbf{x}_i^T \boldsymbol{\beta}^*}, \ y_i \mid \mathbf{x}_i, \ \boldsymbol{\beta}^* \sim \text{Pois}(\lambda_i)$$

For some experiments we simulate multiple datasets and perform multiple runs of ALASMC. We denote the number of particles at one iteration of SMC as N, the number of particles to cut while obtaining the initial sample (burn-in) – B. Recall that we denote the log-likelihood tolerance which is used in ALASMC as $\varepsilon_{loglike}$, and the gradient tolerance corresponding to the Newton-Raphson method – ε_{grad}.

[3]Note that the first entry of the vector of coefficient corresponds to an intercept.

8.2 Results

One of the experiments that we run is based on the simulations from Rossell et al. (2021) (see Section 5.1). That is, the Binomial Logit has $\boldsymbol{\beta}^* = (\mathbf{0}_8, 0.5, 1)$ and the Poisson model has $\boldsymbol{\beta}^* = (\mathbf{0}_4, 0.5, 1, \mathbf{0}_5)$. For Poisson, only $X^{2:6} \sim \mathcal{N}(0, \Sigma)$, and $X^{7:11}$ contains the squared versions of the covariates in $X^{2:6}$. In addition, for the Poisson regression the intercept is forced into a model. We use the Normal prior on the coefficients and the Beta-Binomial prior on the models discussed in subsection 3.1. We sample 30 dataset for each model and run the ALASMC 50 times for each dataset. We initialize SMC at the MLE of a full model.

The experiment considers the difference between the posterior inclusion probabilities of the covariates obtained with ALASMC and LA. The LA estimates are obtained via full enumeration of the model space. For the sake of comparison, we use the Newton-Raphson method in the LA to compute the likelihood mode for each model (see section 5). We also set common gradient tolerance for the ALASMC and the mode computation in the LA full model space enumeration.

The results are presented in Figure 2 and Table 1. The differences between LA and ALASMC marginal posterior probabilities are centered at zero, which means that the algorithm fulfills its goal to target the exact LA estimates. Notice, that the variance is higher for a smaller coefficient. Nevertheless, we have also encountered some SMC runs that stand-out in a way that the estimates do not converge, but it is only 5% of all rounds. The results with the pathological runs are represented in Figure 4 and Figure 5. In this experiments we do not use the curvature adjustment for the Logistic kernel which could also affect the probability to observe these SMC runs that produce intractable results, especially because the covariates are highly correlated.

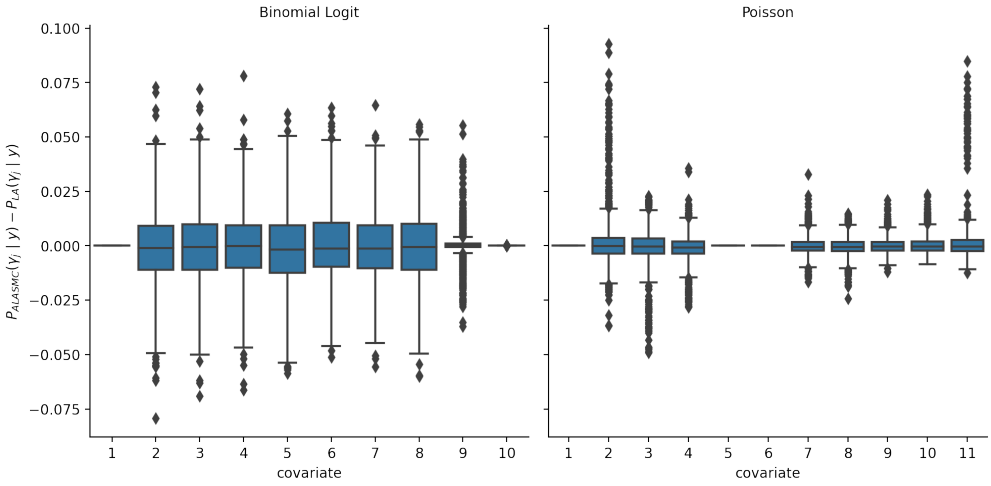
In general, the approximation precision can be increased by enlarging the number of particles at each SMC iteration (for this experiment we have only $N = 5000$) and the way the particles are moved (e.g. in this experiment we use Gibbs kernel discussed in subsubsection 6.4.1 and apply it only once to each particle.)

Table 1 summarizes the proportion of times when the ALA posterior mode matches the LA posterior mode. These results is one more piece of evidence that the ALA SMC gets close enough to LA posterior, but we can see that for Poisson the χ^2 distance between the posterior distribution of ALA SMC and LA is higher, which means that the overall LA posterior distribution for Poisson is harder to approximate. Regarding the recovery rates of ALA SMC, it also reflects the issue with the SMC not really converging all the times.

Now, it is the most likely that it is due to using non-adjusted hessian estimates for Binomial Logit, in Rossell et al. (2021) it is argued that, even for Binomial Logit, using unadjusted ALA can lead to a greater posterior probability mismatch with respect to LA estimates, i.e.: for some samples and SMC runs we can get a bad initial set of particles which affects the performance of the algorithm.

To illustrate the importance of setting a proper initial estimate, we run an experiment that summarizes the number of iterations made until convergence with respect to different numbers of observations, $n \in \{500, 1000, 2500, 5000\}$ given $p = 10$, and dimensions of a

Figure 2: Difference between ALASMC and LA inclusion probabilities.



Logistic: $\boldsymbol{\beta}^* = (\mathbf{0}_8, 0.5, 1)$, $X^{2:10} \sim \mathcal{N}(0, \Sigma)$, unadjusted curvature. *Poisson:* $\boldsymbol{\beta}^* = (\mathbf{0}_4, 0.5, 1, \mathbf{0}_5)$, $X^{2:6} \sim \mathcal{N}(0, \Sigma)$ and $X^{7:11}$ contains the squares of $X^{2:6}$, adjusted curvature, intercept is forced into the model. *Note:* $n = 1000$, $\beta_0 = \arg\max_{\beta_\gamma} p(\boldsymbol{y} \mid \gamma = \mathbb{1}_p, \beta_\gamma)$, $\rho = 0.5$, $N = 5000$, $B = 5000$, $\varepsilon_{grad} = 1e-13$, $\varepsilon_{loglike} = 1e-10$, 30 datasets, 50 runs of SMC.
Intervals: contain 95% of the experimental runs.

Table 1: ALA SMC recovery rates and chi-squared distance to LA posterior.

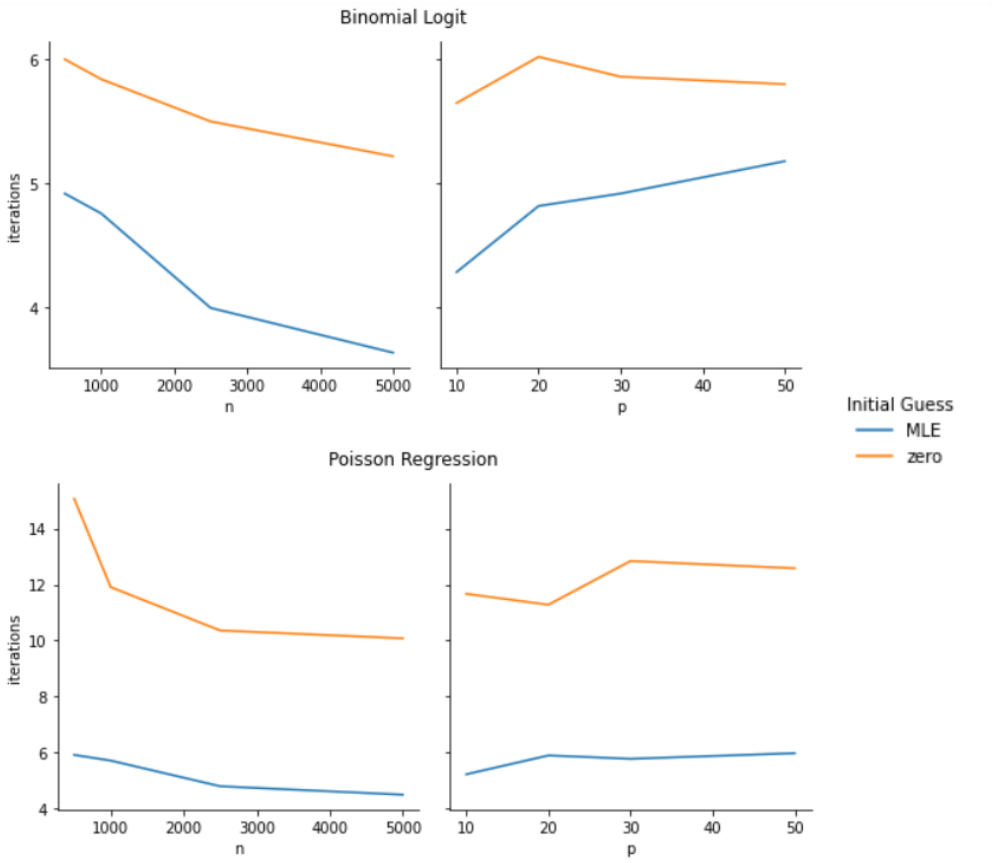
	Recovers		χ^2 distance
Regression	LA	True	to LA posterior
Logistic	0.975	0.853	0.029
Poisson	1.000	1.000	0.921

Logistic: $\boldsymbol{\beta}^* = (\mathbf{0}_8, 0.5, 1)$, $X^{2:10} \sim \mathcal{N}(0, \Sigma)$, non-adjusted curvature.
Poisson: $\boldsymbol{\beta}^* = (\mathbf{0}_4, 0.5, 1, \mathbf{0}_5)$, $X^{2:6} \sim \mathcal{N}(0, \Sigma)$ and $X^{7:11}$ contains the squares of $X^{2:6}$, adjusted curvature.
Note: $n = 1000$, $\beta_0 = \arg\max_{\beta_\gamma} p(\boldsymbol{y} \mid \gamma = \mathbb{1}_p, \beta_\gamma)$, $\rho = 0.5$, $N = 5000, B = 5000, \varepsilon_{grad} = 1e-13$, $\varepsilon_{loglike} = 1e-10$, 30 datasets, 50 runs of SMC.

model space, $p \in \{10, 20, 30, 50\}$ given $n = 1000$. The results are displayed in Figure 3. Notice that for Binomial Login the number of iterations jumps at $p = 50$, it reflects the fact that the MLE estimate

We find that the way one sets the initial guess on the coefficients can affect both the computation time and the quality of approximation, especially – posterior probability matching the true model if it lies in the interior of the considered model space. The results presented in Table 2.

Figure 3: Number of iterations of ALASMC until convergence.



Note: $\boldsymbol{\beta}^* = (\mathbf{0}_{p-3}, 0.5, 0.5, 0.5)$, $\rho = 0.5$, adjusted curvature, $N = 5000$, $B = 5000$, $\varepsilon_{grad} = 1e-5$, $\varepsilon_{loglike} = 1e-5$, 50 datasets, 1 run of SMC.

The increase in the computation time is explained by the fact that the zero vector can lie pretty far from the MLE, so it takes longer to get to it. The dependence on the initial coefficients is more notable for the Poisson regression model, which still might be due to the over-dispersion phenomenon which can be present in bad points even with the curvature adjustment. ALASMC for Poisson initialized at zero vector struggles to converge in a way that its posterior mode matches the true model, but the problem disappears when we initialize at MLE under the full model. Nevertheless, we also see the decrease in the quality of initialization at MLE of the full model when p increases and n stays the same on the example of the Binomial Logit.

We report that the implementation of the logistic kernel mentioned in subsection 7.2 does not work well for our case. It has a low acceptance rate throughout runtime of the algorithm. The main reason we postulate for that is the fact that the ALA immediately focuses on a narrow set of covariates, meaning that there is not enough variation in the

Table 2: ALASMC selection rate of a true model and elapsed time by the initial coefficients.

Model	Initial guess	p	Recovers truth	Included active	Discarded spurious	Elapsed time (s).
Binomial Logit	MLE	10	0.90	2.98	6.92	4.11
		20	0.94	2.94	16.96	9.28
		30	0.76	3.00	26.70	13.83
		50	0.86	2.96	46.84	20.87
	zero	10	0.86	2.96	6.90	5.03
		20	0.96	3.00	16.96	10.85
		30	0.86	2.92	26.90	14.07
		50	0.92	2.92	46.96	20.34
Poisson	MLE	10	0.96	3.00	6.96	4.76
		20	0.96	3.00	16.96	7.93
		30	1.00	3.00	27.00	10.43
		50	1.00	3.00	47.00	15.40
	zero	10	0.24	3.00	6.24	9.55
		20	0.00	3.00	16.00	16.31
		30	0.02	3.00	26.02	46.11
		50	0.00	3.00	46.00	53.53

Note: $n = 1000$, $\boldsymbol{\beta}^* = (\mathbf{0}_{p-3}, 0.5, 0.5, 0.5)$, $\rho = 0.5$, adjusted curvature, $N = 5000$, $B = 5000$, $\varepsilon_{grad} = 1e-5$, $\varepsilon_{loglike} = 1e-5$, 50 datasets, 1 run of SMC.
Initial guess represents the way of choosing the initial coefficients: "zero" – just initialize the ALA SMC at $\beta_0 = \vec{0}$, "MLE" – initialize at the MLE of the full model. "Recovers true" shows the percentage of times the ALASMC posterior mode matches the true model.

data to estimate $q_\theta(\boldsymbol{\gamma})$ well. The possible improvement of it might be using a mix of the Logistic and Gibbs kernels, so Gibbs is run to increase particle diversity and when it reaches a decent level, the Logistic kernel is applied.

We also consider the computational gains of ALA SMC with respect to the LA obtained via a full enumeration of the model space. The results are displayed in Table 3. Note that the number of particles and the burn-in parameters differ and are chosen in a sensible way to balance the computation time and precision (see notes below the table). The results show that the ALA SMC is able to provide a significant computational gain with respect to LA without loosing too much in precision.

9 Future Work and Extensions

- Our implementation could be improved. We did not focus on speed. For instance we did not implement the ALA precomputation when $\boldsymbol{\beta}_{\boldsymbol{\gamma}0} = \mathbf{0}$ could drastically speed up the sampling of the initial particles.

Table 3: Computational gains of ALASMC over LA and precision metrics.

model	p	method	Running time, s.	Recovers LA	Eucl. distance between marginals	χ^2 distance to LA posterior
Binomial Logit	10	ALASMC	1.392	1.000	0.101	0.050
		LA	1.400	—	—	—
	12	ALASMC	4.565	1.000	0.054	0.036
		LA	7.867	—	—	—
	14	ALASMC	7.677	1.000	0.048	0.041
		LA	32.764	—	—	—
Poisson	10	ALASMC	1.273	1.000	0.024	0.015
		LA	3.777	—	—	—
	12	ALASMC	4.871	0.961	0.046	0.763
		LA	11.505	—	—	—
	14	ALASMC	6.386	0.961	0.041	0.777
		LA	37.992	—	—	—

Note: $n = 1000$, $\boldsymbol{\beta}^* = (\mathbf{0}_{p-3}, 0.5, 0.5, 0.5)$, $\rho = 0.5$, adjusted curvature, $\varepsilon_{grad} = 1e-5$, $\varepsilon_{loglike} = 1e-5$, 50 datasets, 1 run of SMC.
For $p = 10$: $N = 1000$, $B = 2500$; $p = 12$: $N = 3000$, $B = 5000$; $p = 14$: $N = 5000$, $B = 5000$
"Euclidean distance between marginal posterior probabilities" shows the average (over simulated datasets) of $\sqrt{\sum_{j=1}^{p}(p_{ALASMC}(\gamma_j \mid \boldsymbol{y}) - p_{LA}(\gamma_j \mid \boldsymbol{y}))^2}$.
"Recovers LA" shows the proportion of runs such that the ALA SMC posterior mode matches the LA posterior mode.

- A key condition for the algorithm to be yield an accurate approximation is to ensure the particle approximation doesn't lost it's accuracy as the iterations progress. A common strategy called *tempering*[4] addresses this issue. The idea is to control the sequence by solving for $\text{ESS}(\mathbf{X}) = \text{ESS}^*$, where ESS^* is a prespecified constant. This is done by parameterizing the sequence of distributions by some parameter λ_t. One such a parameterization could look like

$$\pi_t(x) = \pi_0(x)^{1-\lambda_t}\pi_T(x)^{\lambda_t}$$

  One can then control the accuracy of the particle approximation by choosing a λ_t which does not decrease the ESS too much—by solving by solving for $\text{ESS}(\mathbf{X}) = \text{ESS}^*$ for λ_t, where ESS^* is a prespecified constant. It is described in (Chopin et al., 2020, §17.2.3) how tempering can avoid the curse of dimensionality that Monte Carlo methods usually suffer from. They also show with numerical experiments that this adaptive tempering procedure can provide important improvement to the final accuracy of the algorithm.

  In the context of the algorithm presented in this paper, this λ_t parameterization is

[4]One which we do not employ in this thesis, however see section 9 for a discussion related to this.

not feasible—since the sequence of distributions cannot be expressed deterministically.

However, it may still be worth it to control the degeneration of the particles. Recall that to calculate $\tilde{p}_t(\boldsymbol{\gamma} \mid \mathbf{y})$, we must first run t iterations of an optimization algorithm on the model-specific likelihood. Suppose that the optimization procedure is a iterative descent type algorithm. A possible improvement to the algorithm would introduce setting a step size parameter ρ_t for every optimization procedure associated to γ. For instance assume we are in iteration t of ALASMC procedure. Then would have

$$\beta_{\boldsymbol{\gamma}t} = \beta_{\boldsymbol{\gamma}0} - H^{-1}_{\boldsymbol{\gamma}0} g_{\boldsymbol{\gamma}0} - H^{-1}_{\boldsymbol{\gamma}1} g_{\boldsymbol{\gamma}1} - \ldots - \rho_t H^{-1}_{\boldsymbol{\gamma}t-1} g_{\boldsymbol{\gamma}t-1}$$

where $H_{\boldsymbol{\gamma}k} = H(\beta_{\boldsymbol{\gamma}k})$. With this in mind, imagine that at time t the ESS is very low. This implies that the particle approximation has gone wrong and the resulting estimators will have high variance. A possible strategy would be to decrease ρ_t and re-estimate the ESS. The costly part of this procedure would be the estimation of the ESS, as N new weights would have to be computed. Perhaps there is some way to devise a low-variance estimator of ESS so that we do not to compute the weights of all the particles. Suppose you wanted to know the marginal probability of a certain variable, the authors design an ESS measure that improve the importance sampling estimate given a specific function we are trying to calculate the expectation of. Perhaps an ESS measure geared towards the inference of marginal variable posteriors would yield a better-performing algorithm.

- In section 7 we chose to use the Newton-Raphson optimization procedure to calculate $\tilde{p}(\mathbf{y} \mid \boldsymbol{\gamma})$. However, we could have chosen any other optimization algorithm in principle.

  There are a few reasons why this might not be a great choice. Well first of all is computation time. Calculating a Hessian is an $O(p^3_\gamma)$ operation, and quickly becomes memory intensive for high n. Additionally, the fast convergence of Newton-Raphson is not necessarily an advantage, since a large distance between $\boldsymbol{\beta}_{\boldsymbol{\gamma}j}$ and $\boldsymbol{\beta}_{\boldsymbol{\gamma}(j+1)}$ could diminish the accuracy of the particle approximation in the context of this paper. Other algorithms to try out could include Nestorov's Accelerated gradient, quasi-Newton methods or stochastic gradient descent. These are advantageous as we would avoid the costly computation of a Hessian for every optimization step.

- A fundamental part of any SMC algorithm are the Markov Kernel move steps that ensure the accuracy of the particle approximation at every iteration. In this paper, we do not use particularly sophisticated kernels. Intuitively, having a high mutation rate—especially in later iterations of the algorithm—may be important to capture all the models that the LA captures but the ALA doesn't. Related to this, Dau and Chopin (2020) outline a new kind of SMC algorithm that does the MCMC regeneration in a smarter way. They show how the resulting algorithm is not difficult to tune in terms of choosing the number of kernel steps (we referred to

it as k in section 7) to apply to each particle. Additionally, their method provides an estimator of the variance of the SMC estimate.

10 Conclusion

In this paper we combine two methodologies – the Approximate Laplace Approximation and Sequential Monte Carlo samplers – to obtain an algorithm for Bayesian variable selection which scales moderately well with the dimensionality of the model space. The Laplace Approximation provides an estimate for the integrated-likelihood, $p(\mathbf{y} \mid \boldsymbol{\gamma}) = \int_{\boldsymbol{\beta}_\gamma} p(\mathbf{y} \mid \boldsymbol{\beta}_\gamma, \boldsymbol{\gamma}) p(\boldsymbol{\beta}_\gamma) d\boldsymbol{\beta}_\gamma$, which is a bottleneck in Bayesian Inference. But it requires one to solve a costly optimization problem for each model. The ALA is a computationally cheap way to compute an approximation for the said integrated likelihood. The sequence of distributions $\{\tilde{p}_t(\boldsymbol{\gamma}|\mathbf{y})\}_t$ which is obtained by iteratively applying the ALA converges to the LA posterior. The goal is to utilize this property via SMC to sample efficiently from the LA posterior model probability distribution.

We outline the ALA SMC algorithm and conduct experiments. The results show that the ALA SMC is able to successfully target the LA posterior model probability distribution, it also provides a substantial gain in the computation time and does not lose too much in precision when the parameters of the algorithm are carefully chosen. We also observe that the way of choosing the initial coefficients has a significant impact for both the precision and the elapsed time of the algorithm. That is, choosing MLE under the full model instead of zero-vector decreases the number of iteration until the convergence of the algorithm, and sufficiently increases the proportion runs where ALA SMC posterior mode matches the true model for the Poisson regression.

References

Buchholz, A., Chopin, N., and Jacob, P. E. (2021). Adaptive tuning of hamiltonian monte carlo within sequential monte carlo. *Bayesian Analysis*, 16(3):745–771.

Chopin, N., Papaspiliopoulos, O., et al. (2020). *An introduction to sequential Monte Carlo*. Springer.

Dau, H.-D. and Chopin, N. (2020). Waste-free sequential monte carlo. *arXiv preprint arXiv:2011.02328*.

Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436.

Del Moral, P., Doucet, A., and Jasra, A. (2012). An adaptive sequential monte carlo method for approximate bayesian computation. *Statistics and computing*, 22(5):1009–1020.

Douc, R. and Cappé, O. (2005). Comparison of resampling schemes for particle filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pages 64–69. IEEE.

Guldas, H., Cemgil, A. T., Whiteley, N., and Heine, K. (2015). A practical introduction to butterfly and adaptive resampling in sequential monte carlo. *IFAC-PapersOnLine*, 48(28):787–792.

Kass, R. E. (1990). The validity of posterior expansions based on laplace's method. *Bayesian and likelihood methods in statistics and econometrics*, pages 473–487.

Paulin, D., Jasra, A., and Thiery, A. (2019). Error bounds for sequential monte carlo samplers for multimodal distributions. *Bernoulli*, 25(1):310–340.

Rainforth, T. W. G. (2017). *Automating inference, learning, and design using probabilistic programming*. PhD thesis, University of Oxford.

Rossell, D., Abril, O., and Bhattacharya, A. (2021). Approximate laplace approximations for scalable model selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 83(4):853–879.

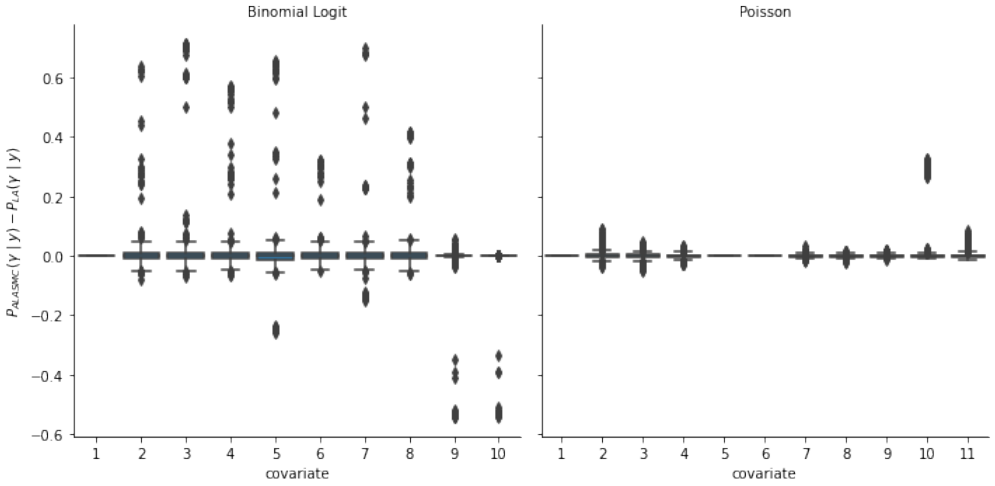
Rossell, D. and Rubio, F. J. (2019). Additive bayesian variable selection under censoring and misspecification. *arXiv preprint arXiv:1907.13563*.

Schäfer, C. and Chopin, N. (2013). Sequential monte carlo on large binary sampling spaces. *Statistics and Computing*, 23(2):163–184.

Vergé, C., Dubarry, C., Del Moral, P., and Moulines, E. (2015). On parallel implementation of sequential monte carlo methods: the island particle model. *Statistics and Computing*, 25(2):243–260.

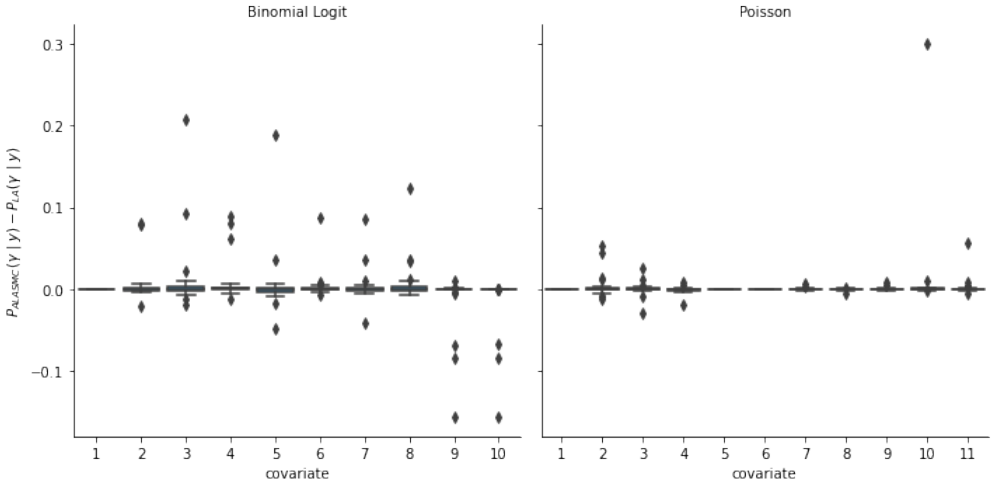
A Experimental Appendix

Figure 4: Difference between ALASMC and LA posterior probabilities (with not converged algorithms).



Logistic: $\boldsymbol{\beta}^* = (\mathbf{0}_8, 0.5, 1)$, $X^{2:10} \sim \mathcal{N}(0, \Sigma)$, unadjusted curvature. *Poisson:* $\boldsymbol{\beta}^* = (\mathbf{0}_4, 0.5, 1, \mathbf{0}_5)$, $X^{2:6} \sim \mathcal{N}(0, \Sigma)$ and $X^{7:11}$ contains the squares of $X^{2:6}$, adjusted curvature, intercept is forced to the model. *Note:* $\beta_0 = \arg\max_{\beta_\gamma} p(\boldsymbol{y} \mid \gamma = \mathbb{1}_p, \beta_\gamma)$, $\rho = 0.5$, $N = 5000$, $B = 5000$, $\varepsilon_{grad} = 1e-13$, $\varepsilon_{loglike} = 1e-10$, 30 datasets, 50 runs of SMC.
Intervals: contain 95% of the experimental runs.

Figure 5: Difference between ALASMC and LA posterior probabilities (averaged runs of SMC for a dataset).



Logistic: $\boldsymbol{\beta}^* = (\mathbf{0}_8, 0.5, 1)$, $X^{2:10} \sim \mathcal{N}(0, \Sigma)$, unadjusted curvature. *Poisson:* $\boldsymbol{\beta}^* = (\mathbf{0}_4, 0.5, 1, \mathbf{0}_5)$, $X^{2:6} \sim \mathcal{N}(0, \Sigma)$ and $X^{7:11}$ contains the squares of $X^{2:6}$, adjusted curvature, intercept is forced to the model. *Note:* $\beta_0 = \arg\max_{\beta_\gamma} p(\boldsymbol{y} \mid \gamma = \mathbb{1}_p, \beta_\gamma)$, $\rho = 0.5$, $N = 5000$, $B = 5000$, $\varepsilon_{grad} = 1e-13$, $\varepsilon_{loglike} = 1e-10$, 30 datasets, 50 runs of SMC.
Intervals: contain 95% of the experimental runs.