# 1. Advanced Inference Techniques

In this module we briefly review some more advanced techniques for fitting models. The main purpose is to highlight these approaches and their benefits, rather than provide a detailed description.

## 1.1. Variational inference

MCMC and MAP estimation represent two extremes of the possibilities for approximating a posterior distribution. MCMC is accurate but computationally expensive, whereas MAP is very inaccurate (in terms of approximating the whole distribution) but relatively fast. In between these extremes there are many other strategies for approximating posteriors. Variational inference (VI) has a emerged as one promising approach and has developed a fairly active research community [2].

VI frames the problem of approximating the posterior distribution as an optimisation problem. The basic idea is to identify a family of approximating distributions $q(\theta|\lambda)$ which are in some sense simpler than the posterior $p(\theta|X)$. The goal is then to find the settings of the variational parameters $\lambda$ that minimise the distance between $q$ and the posterior. Typically the Kullback-Leibler divergence is used, though other alternatives are being explored.

VI is typically much faster than MCMC methods and gives a more accurate posterior approximation than MAP estimation. Historically mean field VI (MFVI) has been the most popular approach [1]. MFVI assumes $q(\theta|\lambda)$ decomposes as product of distribution for each parameter in the model i.e. $q(\theta|\lambda) = \prod_{i=1}^{D} q(\theta_i|\lambda_i)$ where $\theta = (\theta_1, ..., \theta_D)$. The main issue with this approach is that it cannot capture correlation between variables. It was also hard to apply when the model was not composed of distributions in the conjugate exponential family. The primary reason for this constraint was the need to compute expectation under the approximating distribution.

Recent work has significantly relaxed the constraints of MFVI. The key insight has been that difficult to compute expectations can be approximated using Monte Carlo (MC) methods during the optimisation process. Specifically we use MC to estimate the gradient of the distance between $q(\theta|\lambda)$ and $p(\theta|X)$ to perform *stochastic gradient descent*. This greatly expands the scope of models that VI can be applied to, and allows for approximating distributions which can capture correlation between model parameters. The key challenge is controlling the variance of the MC estimates. These advances have motivated an interesting area of research is using neural networks (deep learning) as part of the approximating distribution. Neural networks are powerful tools for function approximation and can in principle lead to quite accurate posterior approximation.

Stochastic gradient descent has also been used to sub-sample data to estimate the gradient of the distance. This can lead to a significant reduction in computational complexity for each step of the gradient descent procedure. One example of the utility of this approach is fitting topic models to large data sets like Wikipedia with millions of data points. Sub-sampling turns out to be a major advantage for VI and is something that cannot be currently done with MCMC.

## 1.2. Advanced MCMC methods

The Metropolis-Hastings (MH) algorithm is a widely used and often effective MCMC technique. However, it can struggle with high dimensional problems. The basic issue is the *curse of dimensionality*. In general if we propose new values far from the current one with MH they will be rejected. As a result we are forced to explore a small local space around our current value. When the dimensionality of the parameter space is high, this means that we cannot move any single component very far. One solution we have discussed is to *block* the data an perform updates on small subsets of variables. This approach begins to fail when there is strong correlation between variables in different blocks.

The other common problem for MH is local optima in the posterior distribution. In general it can be hard for MH to move between modes in the distribution. This can be a problem if we initialise the MH sampler near a local mode which is far from the global optima. In this case we will get stuck sampling around the local mode and fail to explore the rest of the space where most distribution mass exists.

There are a number of more advanced MCMC methods which can be used to overcome these issues. Many of these methods can be used together and with MH to produce better mixing samplers.

### 1.2.1. Simulated annealing

Simulated annealing (SA) is method from the optimisation literature inspired by a physical model of cooling metals. The basic idea of SA is to introduce a sequence of distributions. As we move through the sequence of distributions sampling from them becomes harder, until we reach the distribution we are interested in.

One way to construct this sequence is to anneal the likelihood as follows

$$\begin{aligned} p_\beta(\theta|X) &= p(\theta|X)^\beta p(\theta)^{1-\beta} \\ &= \left(\frac{p(X|\theta)}{p(X)}\right)^\beta p(\theta) \end{aligned}$$

where we take the sequence $\beta_1 = 0 < \beta_2 < \cdots < \beta_T = 1$. The first distribution in the sequence is the prior and the last the posterior. To use this sequence in an MCMC framework we start at $\beta_1$ and run some form of MCMC update like MH. After some number of iterations we then move to $\beta_2$ keeping the last value from our previous updates. We continue this until we reach $\beta_T$ at which point we are sampling from the posterior. We then collect samples from this last iteration as our posterior approximation.

The intuition behind SA is that it will flatten the posterior out early on, making it easy to move between modes. As the the annealing parameter increases, the distribution $p_\beta$ will be become more rugged until we reach the posterior. Hopefully by that point the sampler is near the mode and sampling well from it. In principle SA can escape local optima from bad initialisation.

Tuning the values of $\beta_i$ and selecting the correct number is the main challenge for SA. If the distance between $\beta_i$ and $\beta_{i+1}$ is large, then the associated distributions will be quite different. This in turn can lead the algorithm to get trapped in modes.

### 1.2.2. Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) is an efficient method for sampling high dimensional continuous parameters. Because HMC jointly updates many parameters (potentially all of them) it can work well when there is correlation. When the parameters are highly correlated HMC can significantly outperform MH.

The basic idea is to propose a new value for the MCMC sampler by deterministically moving along a path defined by Hamiltonian dynamics. The connection to posterior inference is that the joint distribution appears as the potential energy term in the Hamiltonian. If we could follow the Hamiltonian path exactly we would then generate a new value which is a sample from our posterior distribution. In practice we cannot follow this path exactly, so we need to use numerical methods to approximate a set of differential equations (DEs). To correct for the numerical errors this introduces, we then need to perform an accept/reject step like MH.

There are two parameters that need to be tuned: the step size in our approximation of the DE and the number of steps we use. Recent work developing automated approaches for these parameters has made HMC much easier to implement. One of the major applications has been in the probabilistic programming language STAN. Auto tuned HMC is the main algorithm used in STAN, and seems to work well for a wide range of models.

The main deficiency of HMC is that it only applies to continuous parameters. There are a few ways around this issue. The first is to marginalise the discrete variables in the model. For example we can sum out the cluster indicator variables in mixture models, or use the forward-backward algorithm to marginalise the hidden states in an HMM. If this is not possible, HMC can still be used to update blocks of continuous variables in the same way as MH.

### 1.2.3. Sequential Monte Carlo

Sequential Monte Carlo (SMC) is a very general algorithm for sampling from high dimensional distributions. The basic idea is to break the sampling problem down into smaller and easier to sample problems. To achieve this SMC maintains a collection of *particles* representing a partial state. At each iteration of the SMC algorithm we update the particles and compute a weight for them. The weight, roughly speaking, is a measure of how good a fit the partial solution represented by the particle is to the data. Resampling is used after each iteration or when some criteria is met to replace low weight particles with high weight ones. At the end of the SMC algorithm we have a set of particles representing the sampled parameters, and a collection of weights that can be used to form an approximation to the posterior.

SMC is often used to sample from distributions with an obvious sequential structure. The classic example is the posterior distribution of the hidden states of an HMM. In this case the $t^{\text{th}}$ iteration of the SMC algorithm proposes a new value for the hidden state $z_t$ of the chain. The weight function depends on the partial likelihood of the data up to $t$. While it is most obvious how to apply SMC in models with sequential structure, it is much more general. Interesting examples include inference in Bayesian mixture models, phylogenetics and probabilistic graphical models.

There are a few considerations when designing an efficient SMC sampler. The first is to identify a sequential decomposition of the problem. In some cases like HMMs this is obvious, while others require more thought. The next consideration is the sequence of target distributions $\{\gamma_t\}_{t=1}^T$. This is related to sequential decomposition, but there is some flexibility. The only real constraint is that the final distribution in the sequence is proportional to the distribution we wish to sample i.e. $\gamma_T(\boldsymbol{\theta}_{1:T}) \propto p(\boldsymbol{\theta}_{1:T}|X)$ for posterior inference. Here we use the notation $\boldsymbol{\theta}_{1:t} = (\theta_1, ..., \theta_t)$ to indicate the sequence of parameters that have been sampled. The final design choice the set of proposal distributions $\{q_t\}_{t=1}^T$ to use at each step. The only real constraint is that we must be able to propose every possible value of $\theta_t$ at each step.

### 1.2.4. Particle Gibbs

SMC can be a powerful method for sampling high dimensional parameters. It is especially useful when the parameters are discrete and HMC cannot be applied. One major deficiency of SMC is that we can only make a single pass. For example, if we want to update the hidden states of an HMM we have to fix the values of the other parameters like the transition matrix.

A simple idea is to use the blocking strategy like we do for MH. We could then use SMC to update the hidden states, and some other MCMC move to update the remaining parameters. We then iterate these steps to draw multiple samples from the posterior. Naively applying this approach does not lead to a valid sampler. The key problem is that SMC does not sample from the correct conditional distribution. The Particle Gibbs (PG) algorithm provides a solution to this problem.

The key modification the PG algorithm makes to the proposed blocked sampler, is to use conditional SMC (cSMC) in place of SMC. This simply amounts to ensuring there is one particle which follows a path that leads to the current state. The modifications required to the standard SMC algorithm to achieve this are fairly minimal. Intuitively the conditional particle acts to keep the other particles in the SMC swarm near the current value. This is achieved through the resampling step, whereby the conditional particle can replace other particles.

**Remark 1.** Recall that in blocked MH sampling we have the parameter vector $\theta = (\theta_1, \theta_2)$ and we would like to sample from the conditional distribution $p(\theta_1|X, \theta_2)$ like a Gibbs sampler. In principle it seems like we need to know $p(X, \theta_2)$ but this cancels away in the acceptance ratio so we only ever need to evaluate the joint distribution $p(X, \theta_1, \theta_2)$. The PG algorithm essentially mimics this cancellation process through the addition of the conditional path.

**Remark 2.** The most challenging aspect of implementing the PG algorithm is typically doing the book keeping for the conditional path.

### 1.2.5. Parallel tempering

Parallel tempering (PT) is a very general approach to address the issues of multi-modal or hard to sample from posteriors. Like SA we have sequence of distributions $\{p_{\beta_i}\}_{i=1}^T$. The difference is we concurrently perform updates for all these distributions using an MCMC algorithm. Periodically we propose to swap the parameters between two chains. The posterior approximation is generated by sampling from the chain with annealing parameter $\beta_T = 1$.

The basic idea of PT is that chains with lower values of $\beta$ are able to explore the space more easily and jump between modes. By swapping values between chains we can slowly percolate new values to the top chain with parameter $\beta_T$ which approximates the posterior. This can lead to big jumps in the space which can cross between modes.

The key issue is designing a good sequence of $\beta_1 < \cdots < \beta_T$ and ensuring we have the right number of chains. If we do not use enough chains the probability of accepting a swap becomes low because the distributions are too different between chains. If we use too many chains, then it will take a very long time for new values to move from the lowest chain to the top. Historically this was a major challenge when implementing PT.

Recent work on *non-reversible* PT methods has made this less of an issue. The key insight from this work is that using a non-reversible scheme for swapping can drastically accelerate the probability of a parameter moving from the bottom chain to the top. In practice this means that we can use as many chains as we can afford computationally. An attractive feature of PT is that each chain can run independently on a separate CPU which allows for massive parallel computation. Only when swaps are proposed do we need to work serially.

### 1.2.6. Approximate Bayesian computation

Approximate Bayesian computation (ABC) is fairly recent approach for sampling from complex models where the likelihood may be intractable or expensive. These type of models come up frequently in population genetics. A simple example is the Wright-Fisher model. While it is reasonably easy to simulate from this model, it is hard to evaluate the probability of the observed data. This poses a problem for classic Bayesian inference where the likelihood is of central importance.

The solution ABC applies for estimating model parameters in these models is surprisingly simple. Assume we have a model with parameters $\theta$. We are able to simulate data given $\theta$, but cannot easily evaluate the likelihood $p(X|\theta)$. In ABC we simply propose values of $\tilde{\theta}$ from the prior distribution, then simulate data $\tilde{X}$ and then compute a measure of distance between the simulated data $\tilde{X}$ and observed data $X$. If the distance is less than a threshold, $\varepsilon$, we add the $\tilde{\theta}$ to our collection of samples used to approximate the posterior.

The accuracy of the posterior approximation depends on the value $\varepsilon$. If $\varepsilon$ is large we will accept many samples, but have a less accurate posterior approximation. In contrast if we use a small value of $\varepsilon$ we will often reject samples and need to run more trials, but the approximation will be better.

One key design choice is the distance metric used to compare the observed data to the simulated data. If we have high dimensional data, then a metric which uses all the data will typically have large distances. Thus, in practice the data is often summarised using lower dimensional summary statistics which are then compared. For example, we could use the mean and variance of the data as our summary statistics and compute the euclidean distance. ABC is a fairly recent development, and there is a great deal of research going on in the field. One of the core problems being considered is how best to choose summary statistics.

ABC is computationally demanding. Typically many millions of simulations will be required before a sufficient number of samples can be collected. Thus, while it could in principle be used to fit any model where other MCMC approaches apply, it will be far too computationally demanding.

# Bibliography

**[1]** Matthew James Beal et al. *Variational algorithms for approximate Bayesian inference*. University of London London, 2003.

**[2]** David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: a review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.