

Statistical analysis of neural data:
Monte Carlo techniques for decoding spike trains

Liam Paninski
Department of Statistics and Center for Theoretical Neuroscience
Columbia University
<http://www.stat.columbia.edu/~liam>

March 31, 2009

Contents

| | | |
|----------|--|-----------|
| 1 | Often we need to integrate over the posterior, not just maximize it | 3 |
| 2 | “Monte Carlo” techniques provide a tractable method for computing integrals | 3 |
| 2.1 | Example: decoding hippocampal place field activity (the static case) | 4 |
| 3 | The “importance sampling” idea extends the range of the basic Monte Carlo technique, but is only effective in low dimensions | 5 |
| 4 | Rejection sampling is a basic building block of more general, powerful Monte Carlo methods | 6 |
| 5 | “Rao-Blackwellization”: it is wise to compute as much as possible analytically | 7 |
| 6 | Markov chain Monte Carlo (MCMC) provides a general-purpose integration technique | 8 |
| 6.1 | The Metropolis-Hastings sampling algorithm is a fundamental tool for constructing Markov chains with the correct equilibrium density | 9 |
| 6.2 | Choosing the proposal density is a key step in optimizing the efficiency of Metropolis-Hastings | 10 |
| 6.2.1 | Non-isotropic random-walk Metropolis is the simplest effective proposal . . . | 11 |
| 6.2.2 | Hybrid (Hamiltonian) Monte Carlo typically mixes much faster than random-walk Metropolis | 13 |
| 6.2.3 | The “Gibbs sampler” is a key special case of the Metropolis-Hastings algorithm | 15 |
| 6.2.4 | The hit-and-run algorithm performs Gibbs sampling in random directions . . | 16 |
| 6.3 | Comparing mixing speed in the log-concave case: for smooth densities, hybrid methods are best; for densities with “corners,” hit-and-run is more effective | 17 |
| 6.4 | Example: comparing MAP and Monte Carlo stimulus decoding given GLM observations | 19 |
| 7 | Gibbs sampling is useful in a wide array of applications | 24 |
| 7.1 | Example: sampling network spike trains given the observed activity of a subset of neurons in the population | 25 |
| 7.2 | Blockwise sampling is often more efficient than Gibbs sampling each variable individually | 25 |
| 7.3 | Example: computing errorbars in low-rank stimulus encoding models | 27 |
| 7.4 | Example: Bayesian adaptive regression splines for time-varying firing rate estimation | 28 |
| 7.5 | Example: estimating hierarchical network spike train models | 29 |
| 7.6 | Example: “slice sampling” provides another easy way to sample from a unimodal density | 29 |
| 7.7 | Example: decoding in the presence of model uncertainty | 29 |
| 8 | Computing marginal likelihood is nontrivial but necessary in a number of applications | 31 |
| 9 | Further reading | 33 |

¹Much of these notes are directly adapted from (Ahmadian et al., 2008). Thanks also to Y. Mishchenko for helpful

1 Often we need to integrate over the posterior, not just maximize it

We have spent most of our time up to now talking about optimization problems — how to maximize the likelihood with respect to some model parameters \vec{k} , or to maximize the *a posteriori* density of a given stimulus \vec{x} . But in many cases we are more interested in computing integrals instead of maximizers. For example, in the change-point setting we were interested in computing marginal probabilities, which required us to integrate out a density over some unobserved variable. Another example: in the decoding context, we might like to choose our estimate \hat{x} to optimize some posterior expected cost,

$$\int C(\vec{x}, \hat{x}) p(\vec{x}|D) d\vec{x}$$

(here, choosing the optimal \hat{x} is a convex optimization problem in \hat{x} whenever $C(\vec{x}, \hat{x})$ is convex in \hat{x} for all \vec{x} — but we must still perform an integral over \vec{x}). As the simplest case, if $C(., .)$ is the squared error between \vec{x} and \hat{x} ,

$$C(\vec{x}, \hat{x}) = \|\vec{x} - \hat{x}\|_2^2,$$

then we may set the gradient with respect to \hat{x} to zero to obtain the classical result that

$$\arg \min_{\hat{x}} \int \|\vec{x} - \hat{x}\|_2^2 p(\vec{x}|D) d\vec{x} = E(\vec{x}|D) = \int \vec{x} p(\vec{x}|D) d\vec{x},$$

where again we have to perform an integral over \vec{x} .

Sadly, however, these integrals are typically not tractable analytically, except in the Gaussian case we have discussed previously. In many cases our standard Gaussian approximation for $p(\vec{x}|D)$ provides an incomplete or inaccurate summary of the posterior, and we need to employ numerical techniques. The straightforward Riemann approach of breaking \vec{x} into a large number of rectangular bins and approximating the integral as a finite sum becomes exponentially inefficient in the dimension of \vec{x} : if we break the unit cube into equally-sized cubes of length L , then $(1/L)^{\dim(\vec{x})}$ blocks are required. This is yet another version of the “curse of dimensionality.”

2 “Monte Carlo” techniques provide a tractable method for computing integrals

To solve this problem we introduce the concept of Monte Carlo integration. We want to compute

$$E_p(g) = \int g(\vec{x}) p(\vec{x}) d\vec{x},$$

for some $g(\vec{x})$. If

$$\int |g(\vec{x})| p(\vec{x}) d\vec{x} < \infty,$$

then we know, by the strong law of large numbers, that if \vec{x}_j are i.i.d. samples from $p(\vec{x})$, then

$$\hat{g}_N^{(p)} \equiv \frac{1}{N} \sum_{j=1}^N g(\vec{x}_j) \rightarrow E_p(g), \quad N \rightarrow \infty.$$

discussions.

Thus we have reduced our hard integration problem to an apparently simpler problem of drawing i.i.d. samples from $p(\vec{x})$. To decide how many samples N are sufficient, we may simply compute

$$\text{Var}(\hat{g}_N) = \frac{1}{N} \text{Var}[g(\vec{x})];$$

when N is large enough that this variance is sufficiently small, we may stop sampling. Of course, in general, computing $\text{Var}[g(\vec{x})]$ is usually also intractable, and so we have to estimate this variance from data, and this can be a difficult problem in itself, e.g. if the random variable $g(\vec{x})$ has large tails.

2.1 Example: decoding hippocampal place field activity (the static case)

As a simple concrete example, imagine that we would like to decode the position x of a rat along a one-dimensional track, given some observed set of neural spike counts $D = \{n_i\}$. Assume that, given x , the spike counts n_i are conditionally independent Poisson random variables with rate

$$\lambda_i(x) = f_i(x),$$

where $f_i(\cdot)$ is neuron i 's tuning curve for position x . Further assume that we know the *a priori* distribution $p(x)$ of observing any position x (i.e., we have watched the rat's behavior long enough to estimate $p(x)$). Then, if we write

$$E(x|D) = \int xp(x|D)dx = \int x \frac{p(D|x)p(x)}{p(D)} dx = \frac{\int xp(D|x)p(x)dx}{p(D)},$$

with

$$p(D) = \int p(x, D)dx = \int p(x)p(D|x)dx,$$

we see that we may apply the Monte Carlo idea twice to compute $E(x|D)$, once for the numerator and once for the denominator: if x_j are i.i.d. samples from the prior density $p(x)$, then

$$\int xp(D|x)p(x)dx = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{j=1}^N x_j p(D|x_j)$$

and

$$p(D) = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{j=1}^N p(D|x_j),$$

with

$$p(D|x) = \prod_i [\lambda_i^{n_i} \exp(-\lambda_i)/n_i!] = \prod_i \frac{f_i(x)^{n_i} \exp(-f_i(x))}{n_i!}.$$

However, these direct methods for computing $p(D)$ and $E(x|D)$ are numerically unstable (i.e., we may need a very large number of samples N to obtain reliable estimates) and are not recommended, for reasons we will discuss below.

3 The “importance sampling” idea extends the range of the basic Monte Carlo technique, but is only effective in low dimensions

Two major problems can arise in a direct application of the Monte Carlo idea. First, it might be difficult to sample directly from $p(\vec{x})$, particularly when \vec{x} is high-dimensional. Second, the likelihood term $p(D|\vec{x})$ might be very sharply peaked; in the preceding position-decoding example, this is often the case when many neurons are observed, since the product over neurons i tends to become exponentially concentrated around its maximum. This means that most of our samples are wasted, since they fall in regions where the $p(D|\vec{x})$ term is relatively very small, leading to a large variance in our estimate of $E(\vec{x}|D)$.

One solution to both of these problems is to use the “importance sampling” trick: instead of drawing i.i.d. samples from $p(\vec{x})$, we draw \vec{x}_j from a “proposal” density $q(\vec{x})$ instead, and then form

$$\hat{g}_N^{(q)} \equiv \frac{1}{N} \sum_{i=1}^N \frac{p(\vec{x}_j)}{q(\vec{x}_j)} g(\vec{x}_j) \xrightarrow{N \rightarrow \infty} \int \frac{p(\vec{x})}{q(\vec{x})} g(\vec{x}) q(\vec{x}) d\vec{x} = \int p(\vec{x}) g(\vec{x}) d\vec{x} = E_p(g).$$

The terms $\frac{p(\vec{x}_j)}{q(\vec{x}_j)}$ — defined so that we obtain the correct expectation, despite the fact that we are not sampling from $p(\vec{x})$ — are known as “importance weights”: we are biasing our sampler so that we can potentially (assuming we choose a good $q(\vec{x})$) spend more of our time sampling from the “important” part of the density.

What constitutes a good choice for $q(\vec{x})$ here? It is reasonable to try to make the variance of our estimate $\hat{g}_N^{(q)}$ as small as possible. We may write out this variance explicitly:

$$Var(\hat{g}_N) = \frac{1}{N} \left(\int \frac{g(\vec{x})^2 p(\vec{x})^2}{q(\vec{x})} d\vec{x} - E_p(g)^2 \right)$$

(note that this reduces to the simpler $Var[g(\vec{x})]$ formula above in the case that $p = q$); a simple calculus of variations argument gives that the optimal q is

$$q^*(\vec{x}) = \frac{1}{Z} |g(\vec{x})| p(\vec{x}),$$

with the normalizer $Z = \int |g(\vec{x})| p(\vec{x}) d\vec{x} < \infty$, by assumption. For example, if we want to compute $p(D) = \int p(\vec{x}) p(D|\vec{x}) d\vec{x}$ for some fixed observation D , the optimal proposal density is

$$q(\vec{x}) = (1/Z) p(\vec{x}) |p(D|\vec{x})| = (1/Z) p(\vec{x}) p(D|\vec{x}) = p(\vec{x}|D),$$

which makes intuitive sense.

It is worth noting that a bad choice of $q(\vec{x})$, on the other hand, can be quite damaging. For example, if $q(\vec{x})$ is such that $\int \frac{g(\vec{x})^2 p(\vec{x})^2}{q(\vec{x})} d\vec{x}$ is infinite (i.e., if the importance weight $\frac{p(\vec{x}_j)}{q(\vec{x}_j)}$ is very large with high $p(\vec{x})$ -probability), then the variance of our estimate \hat{g}_N will be infinite, a situation we would obviously like to avoid. The rule of thumb is that the proposal q should have “fatter tails” than p , in order to avoid very large values of the importance weight p/q .

If we return to the example discussed at length above — decoding the spike trains from a population of GLM-like neurons — then a simple proposal density $q(\vec{x})$ suggests itself: our Gaussian approximation to $p(\vec{x}|D)$. Once we obtain \vec{x}_{MAP} and the Hessian J_x of the log-posterior evaluated at \vec{x}_{MAP} , then to sample from this distribution we need only form $\vec{x}_j = J_x^{-1/2} \vec{z}_j + \vec{x}_{MAP}$, where \vec{z}_j is a standard white Gaussian vector: this proposal $q(\vec{x})$ is therefore very easy to sample from, and is

by construction a fairly good approximation of the true $p(\vec{x}|D)$. However, the Gaussian density has very rapidly-decaying tails, and therefore the importance weight p/q can be exponentially large, leading to an unstable estimator. A better approach is to instead use a multivariate-t proposal with the same mean and covariance. The multivariate-t density with mean μ , covariance matrix C , and degrees of freedom ν is given by

$$t_{\mu,C,\nu}(\vec{x}) = \frac{\Gamma[(\nu+d)/2]}{(\pi\nu)^{d/2}\Gamma(\nu/2)|C|^{1/2}} [1 + \nu^{-1}(\vec{x} - \mu)^T C^{-1}(\vec{x} - \mu)]^{-(\nu+d)/2}, \quad d = \dim(\vec{x})$$

(Nadarajah and Kotz, 2008). This density has power-law (instead of exponential) tails and can be conveniently written as a scale mixture of Gaussians (Portilla et al., 2003):

$$t_{\mu,C,\nu}(\vec{x}) = \int_0^\infty \mathcal{N}_{\mu,C/s}(\vec{x}) h(s) ds,$$

where $h(s)$ is the univariate gamma($\nu/2, \nu/2$) density. Thus, to sample from $t_{\vec{x}_{MAP}, J_x^{-1}, \nu}$, we simply draw

$$\vec{x}_j = s_j^{-1/2} J_x^{-1/2} \vec{z}_j + \vec{x}_{MAP}, \quad s_j \sim \text{gamma}(\nu/2, \nu/2), \quad \vec{z}_j \sim \mathcal{N}_{0,I},$$

where the random scale factor s_j is independent of the Gaussian vector \vec{z}_j . The $t_{\mu,C,\nu}$ -density converges to the corresponding Gaussian density $\mathcal{N}_{\mu,C}$ as $\nu \rightarrow \infty$, as can be seen via well-known properties of the gamma density; $\nu \approx 4$ is a common choice in importance sampling applications (Gelman et al., 2003).

Unfortunately, the utility of importance sampling is often limited to \vec{x} of modest dimensionality. To see why, imagine that we want to compute the expectation of \vec{x} , where each component \vec{x}_i of the vector \vec{x} is generated from some fixed one-dimensional distribution; in fact, for simplicity, we may assume that these components are i.i.d., that is, $p(\vec{x}) = \prod_i p(\vec{x}_i)$. Now imagine we have chosen a proposal density $q(\vec{x}_i)$ which is close to, but not equal to, $p(\vec{x}_i)$. Then if we examine the distribution of the log-importance weight,

$$\log p(\vec{x}) - \log q(\vec{x}) = \sum_i \log p(\vec{x}_i) - \log q(\vec{x}_i) = \sum_i \log \frac{p}{q}(\vec{x}_i),$$

we see that the variance of $\log[p(\vec{x})/q(\vec{x})]$ grows linearly with $\dim(\vec{x})$, since $\log[p(\vec{x})/q(\vec{x})]$ is just a sum of independent random variables in this case. Now when we exponentiate, we see that the importance weight takes on very large values (when $\log[p(\vec{x})/q(\vec{x})]$ is large) and very small values, and this is problematic because it means that our estimate \hat{g}_N is dominated by just a few samples (those samples – or even just the single sample – for which $p(\vec{x})/q(\vec{x})$ is much larger than any of the other importance weights). Thus, as the dimension of \vec{x} increases, the efficiency of the importance sampler becomes progressively smaller. This will motivate the development of more powerful “Markov chain Monte Carlo” (MCMC) tools described below.

4 Rejection sampling is a basic building block of more general, powerful Monte Carlo methods

The importance sampling method provides us with a way to weight samples from the proposal density so that we may compute unbiased averages. But it is often quite useful (for graphical reasons, or as we will see below, as a fundamental building block for more powerful Markov chain-based Monte Carlo methods) to have actual samples from the distribution, not just expectations. It is fairly straightforward (under certain restrictions) to adapt the importance sampling idea to

provide exact samples. We apply the rejection sampling idea: that is, we draw from our proposal q , and then accept with probability cp/q . The constant c is chosen such that $\max_x cp(x)/q(x) \leq 1$, i.e., that the acceptance probability is always less than or equal to one. It is clear that this algorithm provides exact samples, once we write down the density $p^*(x)$ from which the rejection sampler is drawing:

$$p^*(x) = \frac{1}{Z}q(x)p(\text{accept}|x) = \frac{1}{Z}q(x)c\frac{p(x)}{q(x)},$$

where the normalization

$$Z = \int q(x)c\frac{p(x)}{q(x)}dx = c$$

is just the proportion of samples that are accepted; thus

$$p^*(x) = \frac{1}{c}q(x)c\frac{p(x)}{q(x)} = p(x).$$

(Note that the thinning algorithm we discussed in the point process chapter was a special case of this idea.) However, in many cases (particularly in high-dimensional applications) the restriction that $p(x)/q(x)$ be bounded is not met, or the bound $1/c = \max_x p(x)/q(x)$ is so large as to make the method useless. In this case, we must turn to the more general MCMC methods discussed below.

5 “Rao-Blackwellization”: it is wise to compute as much as possible analytically

We saw above that the importance sampling idea (or indeed, the Monte Carlo approach in general) is most effective when we can control the dimensionality of the vector \vec{x} which needs to be sampled. One natural idea along these lines is to perform as much of the integral as possible analytically. More precisely, it is sometimes possible to decompose $p(\vec{x})$ into two pieces,

$$p(\vec{x}) = p(\vec{x}_1)p(\vec{x}_2|\vec{x}_1).$$

In some cases, this decomposition may be chosen such that the integral

$$g(\vec{x}_1) \equiv \int p(\vec{x}_2|\vec{x}_1)g(\vec{x}_1, \vec{x}_2)d\vec{x}_2$$

may be calculated analytically². Now the integral $E_p(g)$ may be computed as

$$E_p(g) = \int d\vec{x}_1p(\vec{x}_1) \int d\vec{x}_2p(\vec{x}_2|\vec{x}_1)g(\vec{x}_1, \vec{x}_2) = \int d\vec{x}_1p(\vec{x}_1)g(\vec{x}_1);$$

in particular, we may apply Monte Carlo methods (now in a the lower-dimensional \vec{x}_1 space) to the integral on the right. The Rao-Blackwell theorem (a version of the usual bias-variance decomposition (Schervish, 1995)) guarantees that this approach, which effectively replaces a randomized estimator for $g(\vec{x}_1)$ with its analytical conditional expectation, will perform at least as well as Monte Carlo integration applied to \vec{x} directly; hence this approach is often referred to as “Rao-Blackwellization.” We will see a number of examples of this approach below.

²The most common example of this is that \vec{x}_2 is conditionally Gaussian given \vec{x}_1 , and $g(\cdot)$ is some simple function such as a polynomial or exponential. For example, it is easy to see that this pertains when we are decoding Gaussian stimuli \vec{x} given observations from a GLM model and the number m of spike counts observed is smaller than the dimensionality of \vec{x} . Thus, in this case, just as in the case that we are optimizing (not integrating) over \vec{x} — where we made use of the representer theorem to reduce the computational load — the computation time is determined by the smaller of $\dim(\vec{x})$ and m .

6 Markov chain Monte Carlo (MCMC) provides a general-purpose integration technique

As we discussed above, if we can sample from a distribution we can (in principle) compute any quantity of interest: means, variances, correlations, and even quantities which are nonlinear functions of the distribution, such as mutual information (as we will see below).

However, it can often be quite challenging to sample directly from a complex distribution $p(\vec{x})$. One (quite clever) idea that has become the default method (Robert and Casella, 2005) is to invent an ergodic Markov chain³ whose conditional distributions $p(\vec{y}|\vec{x})$ may be sampled tractably and whose equilibrium measure is exactly $p(\vec{x})$. Then we may simply iteratively run the chain forward, by recursively sampling

$$p(\vec{x}_{N+1}|\vec{x}_N).$$

We know that the distribution of \vec{x} after N steps of the chain is

$$p_N(\vec{x}) \equiv p(\vec{x}_N) = T^N p_0(\vec{x}),$$

where T abbreviates the transition operator

$$Tg = \int g(\vec{x})p(\vec{y}|\vec{x})d\vec{x}$$

(think of the simplest case, that \vec{x} can take on only a finite number of values d — then g is a d -dimensional vector and T corresponds to a $d \times d$ matrix) and that

$$p_N(\vec{x}) \rightarrow p(\vec{x}), N \rightarrow \infty.$$

Thus if we run the chain long enough, \vec{x}_N will be arbitrarily close to a sample from $p(\vec{x})$.

One drawback is that we don't know how long we need to run the Markov chain: how large does N have to be for $p_N(\vec{x})$ to be “close enough” to $p(\vec{x})$? In the case that the sample space of \vec{x} is finite, we know from basic matrix theory that the largest eigenvalue of T is 1, with corresponding eigenvector $p(\vec{x})$; in an ergodic chain, all other eigenvalues are strictly less than one. Since the size of these smaller eigenvalues determines the rate of convergence of $T^N p_0(\vec{x})$ to $p(\vec{x})$, the goal is to choose T such that the second-largest eigenvalue is as small as possible. More generally, we tend to argue more qualitatively that the chain should “mix” — i.e., “forget” the initial distribution $p_0(\vec{x})$ and evolve towards equilibrium — as quickly as possible; we will discuss this issue more quantitatively in section 6.3 below. We also want to choose the initial density $p_0(\vec{x})$ to be as close to the target density $p(\vec{x})$ as possible (within the constraint that we may sample from $p_0(\vec{x})$ directly); for example, in the decoding context, a reasonable choice for $p_0(\vec{x})$ is the Laplace-approximation proposal density we discussed above. Typically the chain is run for some “burn-in” period before any samples are recorded, so that the effects of the initialization are forgotten and the chain may be considered to be in equilibrium. (If the chain mixes slowly, we'll have to “burn” more samples — yet another reason to construct faster-mixing chains.)

Finding a Markov chain that satisfies our desiderata — (1), ergodicity (or more quantitatively, a rapid mixing speed), and (2),

$$Tp(\vec{x}) = p(\vec{x}),$$

that is, $p(\vec{x})$ as an equilibrium measure — is the hard part, but a few general-purpose approaches have been developed and are quite widely used.

³Ergodicity is a property of a Markov chain that guarantees that the Markov chain evolves towards a unique equilibrium density; roughly speaking, a Markov chain is ergodic if the probability of getting from one state to another is nonzero, and if the chain is *acyclic* in the sense that particles which start at any state \vec{x} do not return back to \vec{x} at any fixed time with probability one.

6.1 The Metropolis-Hastings sampling algorithm is a fundamental tool for constructing Markov chains with the correct equilibrium density

First we give a simple condition for $p(\vec{x})$ to be the equilibrium measure: we say a chain satisfies “detailed balance,” or “reversibility” under $p(\vec{x})$ if

$$p(\vec{x})p(\vec{y}|\vec{x}) = p(\vec{y})p(\vec{x}|\vec{y}) \quad \forall(\vec{x}, \vec{y}).$$

This is a reasonably intuitive condition: it says that, in equilibrium (under $p(\vec{x})$), the rate of flow towards \vec{x} from \vec{y} is exactly equal to the flow of mass towards \vec{y} from \vec{x} . Hence the total rate of change of $p(\cdot)$ is zero; more precisely,

$$(Tp)(\vec{y}) = \int p(\vec{x})p(\vec{y}|\vec{x})d\vec{x} = \int p(\vec{y})p(\vec{x}|\vec{y})d\vec{x} = p(\vec{y}) \int p(\vec{x}|\vec{y})d\vec{x} = p(\vec{y}),$$

as desired.

Now we are in a position to define the Metropolis-Hastings sampling algorithm. This is based on the rejection sampling idea we discussed above: given \vec{x} , we sample from some proposal density $q(\vec{y}|\vec{x})$, and accept this proposal with probability $\alpha(\vec{y}|\vec{x})$; note that that both our proposal density $q(\cdot|\cdot)$ and acceptance probability $\alpha(\cdot|\cdot)$ are allowed to depend on \vec{x} . Thus the total probability of rejecting a step away from \vec{x} is simply

$$R(\vec{x}) = 1 - \int \alpha(\vec{y}|\vec{x})q(\vec{y}|\vec{x})d\vec{y}.$$

Now we must choose $\alpha(\vec{y}|\vec{x})$ carefully so that the corresponding chain

$$p(\vec{y}|\vec{x}) = \alpha(\vec{y}|\vec{x})q(\vec{y}|\vec{x}) + R(\vec{x})\delta(\vec{y} - \vec{x})$$

satisfies detailed balance under $p(\vec{x})$.

This turns out to be easier than it might first appear. First, note that for $\vec{x} \neq \vec{y}$, $p(\vec{y}|\vec{x}) = \alpha(\vec{y}|\vec{x})q(\vec{y}|\vec{x})$. Therefore, to satisfy detailed balance, we want to set

$$p(\vec{x})\alpha(\vec{y}|\vec{x})q(\vec{y}|\vec{x}) = p(\vec{y})\alpha(\vec{x}|\vec{y})q(\vec{x}|\vec{y}) \quad \forall(\vec{x}, \vec{y}). \quad (1)$$

Further, we want to make our acceptance probabilities $\alpha(\vec{y}|\vec{x})$ and $\alpha(\vec{x}|\vec{y})$ as large as possible, to avoid wasting proposals. Clearly, the ideal situation is that detailed balance is satisfied automatically, without any rejections at all, i.e.,

$$p(\vec{x})q(\vec{y}|\vec{x}) = p(\vec{y})q(\vec{x}|\vec{y}),$$

in which case we can set $\alpha(\vec{y}|\vec{x}) = \alpha(\vec{x}|\vec{y}) = 1$. But of course this equation will usually not hold; often we will have either

$$p(\vec{x})q(\vec{y}|\vec{x}) > p(\vec{y})q(\vec{x}|\vec{y})$$

or

$$p(\vec{x})q(\vec{y}|\vec{x}) < p(\vec{y})q(\vec{x}|\vec{y}).$$

In the first case, we want to make the left-hand side smaller to satisfy detailed balance, so we introduce the factor $\alpha(\vec{y}|\vec{x}) < 1$ on the left-hand side. Similarly, in the second case, we introduce the factor $\alpha(\vec{x}|\vec{y})$ to make the right-hand side smaller. To satisfy eq. (1), we just set $\alpha(\vec{y}|\vec{x}) =$

$p(\vec{y})q(\vec{x}|\vec{y})/p(\vec{x})q(\vec{y}|\vec{x})$ on the left-hand side, and flip this ratio to set $\alpha(\vec{x}|\vec{y})$ on the right-hand side. Putting these three cases together, we can express our acceptance probability a bit more elegantly,

$$\alpha(\vec{y}|\vec{x}) \equiv \min \left(1, \frac{p(\vec{y})q(\vec{x}|\vec{y})}{p(\vec{x})q(\vec{y}|\vec{x})} \right),$$

for any pair (\vec{x}, \vec{y}) . This completes the definition of the Metropolis-Hastings (M-H) algorithm.

Thus we have succeeded in our goal of constructing a simple algorithm that allows us to sample from a distribution which becomes arbitrarily close to $p(\vec{x})$ (as N increases). See (Billera and Diaconis, 2001) for a geometric interpretation of the algorithm; it turns out that the choice of the acceptance ratio $\alpha(\cdot|\cdot)$ corresponds to a kind of projection of our proposal Markov chain onto the set of chains which satisfy detailed balance under $p(\vec{x})$. See (Neal, 2004) for further details, including a generalization to the more powerful class of non-reversible chains.

It is worth pointing out a couple special features of this algorithm. First, note that in the special case of symmetric proposals, $q(\vec{y}|\vec{x}) = q(\vec{x}|\vec{y})$, then the acceptance probability simplifies:

$$\alpha_{symm}(\vec{y}|\vec{x}) \equiv \min \left(1, \frac{p(\vec{y})}{p(\vec{x})} \right);$$

this was the original Metropolis algorithm (later generalized to the nonsymmetric case by Hastings). This proposal has the nice feature that proposals which increase the value of $p(\cdot)$ are always accepted (i.e., if $p(\vec{y}) \geq p(\vec{x})$, then $\alpha_{symm}(\vec{y}|\vec{x}) = 1$), while proposals which decrease $p(\cdot)$ are only accepted probabilistically. This makes the analogy with the “simulated annealing” technique from optimization quite transparent; the M-H algorithm is just simulated annealing when the “temperature” is set to one.

Note also, very importantly, that no matter what proposal density we use we only need to know $p(\vec{x})$ up to a constant. That is, it is enough to know that

$$p(\vec{x}) = \frac{1}{Z}r(\vec{x})$$

for some relatively easy-to-compute $r(\vec{x})$ — we do *not* need to calculate the normalization constant Z (since the Z term cancels in the ratio $p(\vec{y})/p(\vec{x})$), which in many cases is intractable. This property has made the Metropolis-Hastings algorithm perhaps the fundamental algorithm in Bayesian statistics.

6.2 Choosing the proposal density is a key step in optimizing the efficiency of Metropolis-Hastings

As discussed above, the key to the success of any MCMC algorithm is to find a chain that reaches equilibrium and mixes as quickly as possible. In the context of the M-H algorithm, we want to choose the proposal density $q(\vec{y}|\vec{x})$ to maximize the mixing speed and to minimize the time needed to compute and sample from $q(\cdot)$. A good rule of thumb is that we would like the proposals $q(\cdot)$ to match the true density $p(\cdot)$ as well as possible (in the limiting case that $p(\cdot)$ and $q(\cdot)$ are exactly equal, it is clear that the acceptance probability α will be one, and the M-H algorithm gives i.i.d. samples from $p(\cdot)$, which is the best we can hope for; of course, in general, we can’t sample directly from $p(\cdot)$, or else the MCMC technique would not be necessary).

The so-called “independence” sampler is the simplest possible proposal: we choose the proposal $q(\cdot)$ to be independent of the current state \vec{x} , i.e. $q(\vec{y}|\vec{x}) = q(\vec{y})$, for some fixed $q(\vec{y})$. For example, the proposal $q(\vec{x})$ could be taken to be our Gaussian or student-t approximation to $p(\vec{x}|D)$. This

is extremely easy to code, and the simplicity of the proposal makes detailed mathematical analysis of the mixing properties of the resulting Markov chain much more tractable than in the general case (Liu, 2002). However, as we discussed above, it can be very difficult to choose a $q(\vec{x})$ which is both easy to sample from directly and which is close enough to the (possibly high-dimensional) $p(\vec{x})$ that the importance weights $p(\vec{x})/q(\vec{x})$ — and therefore the M-H term

$$\frac{p(\vec{y})q(\vec{x}|\vec{y})}{p(\vec{x})q(\vec{y}|\vec{x})} = \frac{p(\vec{y})q(\vec{x})}{p(\vec{x})q(\vec{y})}$$

— are well-behaved. Thus in most cases we need to choose a proposal density which depends on the current point \vec{y} .

We will discuss a few of the most useful example proposals below, but before going into these specifics, it is worth emphasizing that, all of the M-H algorithms we will discuss propose jumps $\vec{y} - \vec{x}$ which are in some sense “local.” Thus, just as in the optimization setting, M-H methods are typically much more effective in sampling from unimodal (e.g., log-concave) target densities $p(\vec{x})$ than more general $p(\vec{x})$, which might have local maxima that act as “traps” for the sampler, decreasing the overall mixing speed of the chain. Therefore our focus in this section will be on constructing efficient M-H methods for log-concave densities; in particular, we will focus on the example of stimulus decoding given GLM observations, where we know that log-concave posteriors arise quite naturally.

6.2.1 Non-isotropic random-walk Metropolis is the simplest effective proposal

Perhaps the most common proposal is of the random walk type: $q(\vec{y}|\vec{x}) = q(\vec{y} - \vec{x})$, for some fixed density $q(\cdot)$. This is a special case of the symmetric proposal discussed above; recall that the acceptance probability simplifies somewhat in this case. Centered isotropic Gaussian distributions are a simple choice for $q(\cdot)$, leading to proposals

$$\vec{y} \sim \vec{x} + \sigma \vec{\epsilon}, \tag{2}$$

where $\vec{\epsilon}$ is Gaussian of zero mean and identity covariance, and σ determines the proposal jump scale. (We will discuss the choice of σ at more length in the next section.) Of course, different choices of the proposal distribution will affect the mixing rate of the chain. To increase this rate, it is generally a good idea to align the axes of $q(\cdot)$ with the target density, if possible, so that the proposal jump scales in different directions are roughly proportional to the width of $p(\vec{x})$ along those directions. Such proposals will reduce the rejection probability and increase the average jump size by biasing the chain to jump in more favorable directions. For Gaussian proposals, we can thus choose the covariance matrix of $q(\cdot)$ to be proportional to the covariance of $p(\vec{x})$. Of course, calculating the latter covariance is often a difficult problem (which the MCMC method is intended to solve!), but in many cases we can exploit our usual Laplace approximation ideas and take the inverse of the Hessian of the log-posterior at MAP as a rough approximation for the covariance. This is equivalent to modifying the proposal rule (2) to

$$\vec{y} \sim \vec{x} + \sigma A \vec{\epsilon}, \tag{3}$$

where A is the Cholesky decomposition of H^{-1}

$$AA^T = H^{-1}, \tag{4}$$

and H is the negative log-Hessian of the log-posterior at the MAP solution. We refer to chains with such jump proposals as non-isotropic Gaussian RWM. Figure 1 compares the isotropic and nonisotropic proposals.

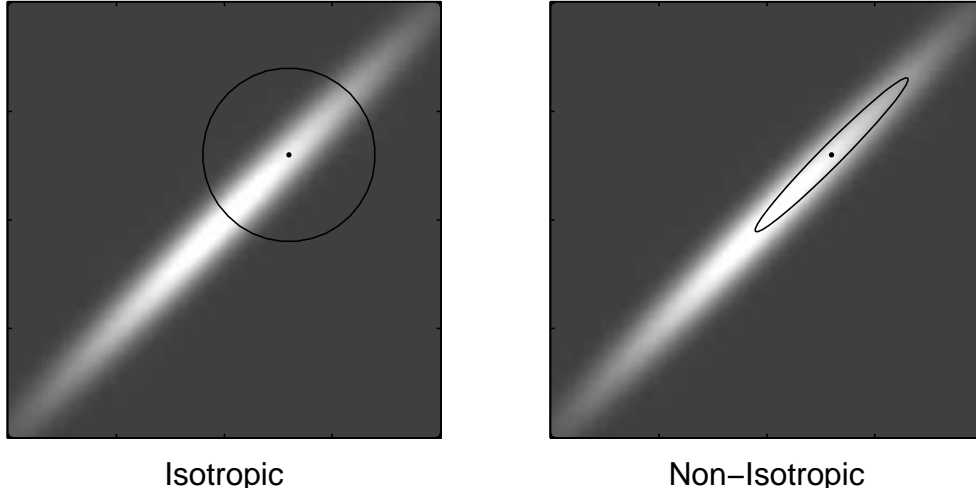


Figure 1: Comparison of isotropic and non-isotropic Markov jumps for the Gaussian RWM and hit-and-run chains. In the RWM case, the circle and the ellipse are level sets of the Gaussian proposal distributions for jumping from the dot at their center. In isotropic (non-isotropic) hit-and-run, the jump direction \mathbf{n} is generated by normalizing a vector sampled from an isotropic (non-isotropic) Gaussian distribution centered at the origin. The non-isotropic distributions were constructed using the Hessian in the Laplace approximation, so that the ellipse is described by $\vec{x}^T H \vec{x} = \text{const}$. When the underlying distribution, $p(\vec{x})$, is highly non-isotropic, it is disadvantageous to jump isotropically, as this reduces the average jump size (since many proposals are rejected, corresponding to a jump size of zero) and slows down the chain’s mixing rate. In RWM, the proposal jump scale can not be much larger than the scale of the narrow “waist” of the underlying distribution, lest the rejection rate gets large (as most proposals will fall in the dark region of small $p(\vec{x})$) and the chain gets stuck. Non-isotropic sampling alleviates this problem.

The modification Eq. (3) is equivalent to running a chain with isotropic proposals Eq. (2), but for the auxiliary distribution $\tilde{p}(\tilde{x}) = |A|p(A\tilde{x})$ (whose corresponding Laplace approximation corresponds to a standard Gaussian with identity covariance), and subsequently transforming the samples, \tilde{x}_t , by the matrix A to obtain samples $\vec{x}_t = A\tilde{x}_t$ from $p(\vec{x})$. Implementing non-isotropic sampling using the transformed distribution $\tilde{p}(\tilde{x})$, instead of modifying the proposals as in Eq. (3), is more readily extended to chains more complicated than RWM (see below).

In general, the multiplication of a vector of dimensionality d by a matrix requires $O(d^2)$ time, and the inversion of a $d \times d$ matrix $O(d^3)$. In the decoding problems we are interested in, the dimension of \vec{x} is often proportional to the temporal duration of the stimulus. Thus, naively, the one-time inversion of H and calculation of A takes $O(T^3)$ computations, where T is the duration of the stimulus, while the multiplication of \vec{x} by A in *each step* of the MCMC algorithm takes $O(T^2)$ computations. This would make the decoding of stimuli with even moderate duration forbidding. Fortunately, the quasi-locality of the GLM model allows us to overcome this limitation. Since the stimulus filters in the GLM will in general have a finite temporal duration, T_k , the Hessian of the GLM log-likelihood is banded in time: the matrix element $J_{t_1 n_1, t_2 n_2} \equiv -\partial^2 \log p(\mathbf{r}|\vec{x}) / \partial x(t_1, n_1) \partial x(t_2, n_2)$ vanishes when $|t_1 - t_2| \geq 2T_k - 1$.

Now, the Hessian of the log-posterior is the sum of the Hessians of the log-prior and the log-likelihood, which in the Gaussian case is

$$H = J + C^{-1},$$

where \mathcal{C} is the prior covariance. Thus, if \mathcal{C}^{-1} is banded, H will be banded in time as well. As an example, Gaussian autoregressive processes of any finite order form a large class of priors which have banded \mathcal{C}^{-1} . In particular, for white-noise stimuli, \mathcal{C}^{-1} is diagonal, and therefore H will have the same bandwidth as J . Efficient algorithms can find the Cholesky decomposition of a banded $d \times d$ matrix, with bandwidth n_b , in a number of computations $\propto n_b^2 d$, instead of $\propto d^3$. Likewise, if B is a banded triangular matrix with bandwidth n_b , the linear equation $B\vec{x} = \vec{y}$ can be solved for \vec{x} in $\propto n_b d$ computations.

Therefore, to calculate $\vec{x} = A\tilde{x}$ from \tilde{x} in each step of the Markov chain, we proceed as follows. Before starting the chain, we first calculate the Cholesky decomposition of H , such that $H = B^T B$ and $\vec{x} = A\tilde{x} = B^{-1}\tilde{x}$. Then, at each step of the MCMC, given \tilde{x}_t , we find \vec{x}_t by solving the equation $B\vec{x}_t = \tilde{x}_t$. Since both of these procedures involve a number of computations that only scale with d (and thus with T), we can perform the whole MCMC decoding in $O(T)$ computational time. This allows us to decode stimuli with essentially unlimited durations. Similar $O(T)$ methods have been applied frequently in the state-space model literature (Shephard and Pitt, 1997; Davis and Rodriguez-Yam, 2005; Jungbacker and Koopman, 2007); we will discuss these state-space methods in more depth in a later chapter.

6.2.2 Hybrid (Hamiltonian) Monte Carlo typically mixes much faster than random-walk Metropolis

The random-walk chains discussed above are convenient, but often mix slowly in practice, basically because random walks spend a lot of time re-covering old ground: in the simplest case, a Brownian motion requires $O(n^2)$ time to achieve a distance of n steps from its starting point, on average. The so-called hybrid (or Hamiltonian) Monte Carlo (HMC) method is a powerful method for constructing chains which avoid the slow mixing behavior of random-walk chains. This method was originally inspired by the equations of Hamiltonian dynamics for the molecules in a gas (Duane et al., 1987), but has since been used extensively in Bayesian settings (for its use in sampling from posteriors based on GLMs see (Ishwaran, 1999); see also (Neal, 1996) for further applications and extensions).

Let's begin with the physical analogy. Let \vec{x} be the vector of positions of all the molecules in a gas at a moment in time, and \vec{p} the vector of their momenta (\vec{x} and \vec{p} have the same dimension). In thermal equilibrium, these variables have the distribution $p(\vec{x}, \vec{p}) \propto e^{-H(\vec{x}, \vec{p})}$, where

$$H(\vec{x}, \vec{p}) = \frac{1}{2}\vec{p}^T\vec{p} + \mathcal{E}(\vec{x})$$

is the Hamiltonian (total energy) of the system, which is the sum of the kinetic energy, $\frac{1}{2}\vec{p}^T\vec{p}$, and the potential energy, $\mathcal{E}(\vec{x})$. Notice that due to the factorization of this distribution, \vec{x} and \vec{p} are independently distributed; the marginal distribution of \vec{x} is proportional to $\exp(-\mathcal{E}(\vec{x}))$ and that of \vec{p} is the Gaussian $\propto \exp(-\frac{1}{2}\vec{p}^T\vec{p})$. Thus if we could obtain samples $\{(\vec{x}_t, \vec{p}_t)\}$ from $p(\vec{x}, \vec{p})$ with the potential energy $\mathcal{E}(\vec{x}) = -\log p(\vec{x}) + \text{const.}$, we would also have obtained samples from the marginal $\{\vec{x}_t\}$ from $p(\vec{x})$.

Now, \vec{x} and \vec{p} evolve according to the Hamiltonian dynamics

$$\dot{\vec{p}} = -\frac{\partial H}{\partial \vec{x}} = -\nabla\mathcal{E}(\vec{x}), \quad \dot{\vec{x}} = \frac{\partial H}{\partial \vec{p}} = \vec{p}, \quad (5)$$

under which the energy $H(\vec{x}, \vec{p})$ is conserved. These dynamics also preserve the volume element $d\vec{x}d\vec{p}$ in the (\vec{x}, \vec{p}) space (this is the content of Liouville's theorem from classical dynamics). These

two facts imply that the continuous-time (deterministic) Markov chain in the (\vec{x}, \vec{p}) space that corresponds to the solution of Eqs. (5) leaves the distribution $p(\vec{x}, \vec{p}) \propto e^{-H(\vec{x}, \vec{p})}$ invariant. Ergodicity of this Markov chain can be assured if we “thermalize” the momentum variables, by sampling \vec{p} from its exact marginal distribution (the standard Gaussian) at regular time intervals. Thus solutions of Eqs. (5) should lead to good transition densities for our MCMC chain. Moreover, we expect these proposals to outperform the basic random-walk proposals for two reasons: first, by construction, the Hamiltonian method will follow the energy gradient and thus propose jumps towards higher-probability regions, instead of just proposing random jumps without paying attention to the gradient of $p(\vec{x})$. Second, the Hamiltonian method will move for many time steps in the same direction, allowing us to move n steps in $O(n)$ time, instead of the $O(n^2)$ time required by the random-walk method.

In practice, a numerical solution to Eqs. (5) is obtained by discretizing the Hamiltonian equations, in which case $H(\vec{x}, \vec{p})$ may change slightly with each step, and the M-H correction is required to guarantee that $p(\vec{x}, \vec{p})$ remains the equilibrium distribution. The discretization customarily used in HMC applications is the “leapfrog” method, which is time reversible and leaves the (\vec{x}, \vec{p}) -volume element invariant. The complete HMC algorithm with the leapfrog discretization has the following steps for generating $\{\vec{x}_{t+1}, \vec{p}_{t+1}\}$, starting from $\{\vec{x}_t, \vec{p}_t\}$, in each step of the Markov chain:

1. Set $\vec{x} := \vec{x}_t$, and sample \vec{p} from the isotropic Gaussian distribution $\mathcal{N}_d(0, \mathbf{1})$.
2. Set $\vec{p}_0 := \vec{p}$, and update (\vec{x}, \vec{p}) by repeating the following leapfrog steps L times
 - $\vec{p} := \vec{p} - \frac{\sigma}{2} \nabla \mathcal{E}(\vec{x})$
 - $\vec{x} := \vec{x} + \sigma \vec{p}$
 - $\vec{p} := \vec{p} - \frac{\sigma}{2} \nabla \mathcal{E}(\vec{x})$
3. With probability $\max\{1, \exp(-\Delta H)\}$, where $\Delta H \equiv H(\vec{x}, \vec{p}) - H(\vec{x}_t, \vec{p}_0)$, accept the proposal \vec{x} as \vec{x}_{t+1} . Otherwise reject it and set $\vec{x}_{t+1} = \vec{x}_t$. (It can be shown that this is a bona fide Metropolis-Hastings rejection rule, ensuring that the resulting MCMC chain indeed has the desired equilibrium density (Duane et al., 1987).)

This chain has two parameters, L and σ , which can be chosen to maximize the mixing rate of the chain while minimizing the number of evaluations of $\mathcal{E}(\vec{x})$ and its gradient. In practice, even a small L (e.g., $L < 5$) often yields a rapidly mixing chain. The special case $L = 1$ corresponds to a chain that has proposals of the form

$$\vec{y} \sim \vec{x} - \frac{\sigma^2}{2} \nabla \mathcal{E}(\vec{x}) + \sigma \vec{p},$$

where \vec{p} is normal with zero mean and identity covariance, and the proposal \vec{y} is accepted according to the usual M-H rule. In the limit $\sigma \rightarrow 0$, this chain becomes a continuous Langevin process with the potential function $\mathcal{E}(\vec{x}) = -\log p(\vec{x})$, whose stationary distribution is the Gibbs measure, $p(\vec{x}) = \exp(-\mathcal{E}(\vec{x}))$, *without* the Metropolis-Hastings rejection step. For a finite σ , however, the Metropolis-Hastings acceptance step is necessary to guarantee that $p(\vec{x})$ is the invariant distribution. The chain is thus referred to as the “Metropolis-adjusted Langevin” algorithm (MALA) (Roberts and Rosenthal, 2001).

The scale parameter σ (which also needs to be adjusted for the RWM chain) sets the average size of the proposal jumps: we must typically choose this scale to be small enough to avoid jumping wildly into a region of low $p(\vec{x})$ (and therefore wasting the proposal, since it will be rejected with high probability). At the same time, we want to make the jumps as large as possible, on average,

in order to improve the mixing time of the algorithm. See (Roberts and Rosenthal, 2001) and (Gelman et al., 2003) for some tips on how to find a good balance between these two competing desiderata for the RWM and MALA chains; we will discuss this issue further in section 6.3 below.

For highly non-isotropic distributions, the HMC chains can also be enhanced by exploiting the Laplace approximation. As in the preceding section, we just sample from the auxiliary distribution $\tilde{p}(\tilde{x}) = |A|p(A\tilde{x})$ (where A is given in Eq. (4)) using the unmodified HMC chain, described above, and subsequently transform the samples by A . As explained above, we can perform this transformation efficiently in $O(T)$ computational time, where T is the stimulus duration. Another practical advantage of this transformation by A is that the process of finding the appropriate scale parameter σ simplifies considerably, since $\tilde{p}(\tilde{x})$ may be approximated as a Gaussian distribution with identity covariance irrespective of the scaling of different dimensions in the original distribution $p(\vec{x})$.

It is worth noting that when sampling from high-dimensional distributions with sharp gradients, the MALA, HMC, and RWM chains have a tendency to be trapped in “corners” where the log-posterior changes suddenly. This is because when the chain eventually ventures close to the corner, a jump proposal will very likely fall on the exterior side of the sharp high-dimensional corner (the probability of jumping to the interior side from the tip of a cone decreases exponentially with increasing dimensionality). Thus most proposals will be rejected, and the chain will effectively stop. (We can, of course, make the stepsize σ very small in the HMC setting to avoid jumping off of these corners, but this slows the chain down and thus defeats the purpose of using HMC.) As we will see below, the “hit-and-run” chain is known to have an advantage in escaping from such sharp corners (Lovasz and Vempala, 2003). We will discuss this point further in Sec. 6.2.4.

HMC methods comprise a significant and growing literature of their own in physics and molecular dynamics. Current research focuses on methods for taking larger steps and for temporally correlating the samples from the momentum variables \vec{p} , to further increase mixing speed; see, e.g., (Izaguirre and Hampton, 2004; Akhmatskaya et al., 2009) for a couple recent examples.

6.2.3 The “Gibbs sampler” is a key special case of the Metropolis-Hastings algorithm

Gibbs sampling (Geman and Geman, 1984) is an important special MCMC scheme. It is particularly efficient when the one-dimensional conditionals $p(x_m|\vec{x}_{\perp m})$ are easy to sample from. Here, x_m is the m -th component of \vec{x} , and $\vec{x}_{\perp m}$ denotes the other components, i.e., the projection of \vec{x} on the subspace orthogonal to the m -th axis. The Gibbs update is defined as follows: first choose the dimension m randomly or in order. Then update \vec{x} along this dimension, i.e., sample x_m from $p(x_m|\vec{x}_{\perp m})$ (while leaving the other components fixed). This is equivalent to sampling a one-dimensional auxiliary variable, s , from

$$s \sim h(s|m, \vec{x}) \propto p(\vec{x} + s\mathbf{e}_m), \quad -\infty < s < \infty, \quad (6)$$

and setting $\vec{y} = \vec{x} + s\mathbf{e}_m$, where \mathbf{e}_m is the unit vector along the m -th axis (we will discuss a couple methods for efficiently sampling from this one-dimensional distribution below).

To show that the Gibbs rule is indeed a special case of the Metropolis-Hastings idea, think of each step in the Gibbs algorithm as a proposal, and note that we always accept this proposal, implying that $\alpha(\vec{y}|\vec{x})$ in the M-H rule must be one if the Gibbs update is, in fact, a M-H update. Is this true? Recall

$$\alpha(\vec{y}|\vec{x}) = \min \left(1, \frac{p(\vec{y})q(\vec{x}|\vec{y})}{p(\vec{x})q(\vec{y}|\vec{x})} \right),$$

which in this case is

$$\min \left(1, \frac{p(\vec{x}_{A_i^c})p(\vec{y}_{A_i}|\vec{x}_{A_i^c})p(A_i^c)p(\vec{x}_{A_i}|\vec{x}_{A_i^c})}{p(\vec{x}_{A_i^c})p(\vec{x}_{A_i}|\vec{x}_{A_i^c})p(A_i^c)p(\vec{y}_{A_i}|\vec{x}_{A_i^c})} \right) = 1,$$

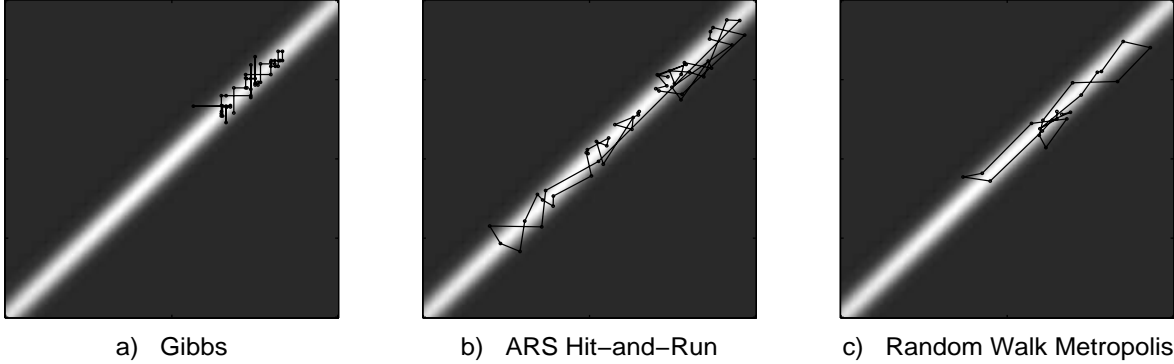


Figure 2: Comparison of different MCMC algorithms in sampling from a non-isotropic truncated Gaussian distribution. Panels (a-c) show 50-sample chains for a Gibbs, isotropic hit-and-run, and isotropic random walk Metropolis (RWM) samplers, respectively. The grayscale indicates the height of the probability density. As seen in panel (a), the narrow, non-isotropic likelihood can significantly hamper the mixing of the Gibbs chain as it chooses its jump directions unfavorably. The hit-and-run chain, on the other hand, mixes much faster as it samples the direction randomly and hence can move within the narrow high likelihood region with relative ease. The mixing of the RWM chain is relatively slower due to its rejections (note that there are fewer than 50 distinct dots in panel (c) due to rejections; the acceptance rate was about 0.4 here). For illustrative purposes, the hit-and-run direction and the RWM proposal distributions were taken to be isotropic here, which is disadvantageous, as explained in the text (also see Fig. 1).

where $p(A_i)$ denotes the probability that we have chosen to update the subset A_i .

It is important to note that the Gibbs update rule can sometimes fail to lead to an ergodic chain (i.e., the chain can get “stuck” and not sample from $p(\vec{x})$ properly) (Robert and Casella, 2005). An extreme case of this is when the conditional distributions $p_m(x_m|\vec{x}_{\perp m})$ are deterministic: then the Gibbs algorithm will never move, clearly breaking the ergodicity of the chain. More generally, in cases where strong correlations between the components of \vec{x} lead to nearly deterministic conditionals, the mixing rate of the Gibbs method can be extremely low (panel (a) of Fig. 2, shows this phenomenon for a 2-dimensional distribution with strong correlation between the two components). Thus, it is a good idea to choose the parameterization of the model carefully before applying the Gibbs algorithm. For example, we can change the basis, or more systematically, exploit the Laplace approximation, as described above, to sample from the auxiliary distribution $\tilde{p}(\vec{x})$ instead.

6.2.4 The hit-and-run algorithm performs Gibbs sampling in random directions

The hit-and-run algorithm (Lovasz and Vempala, 2003) can be thought of as “random-direction Gibbs”: in each step of the hit-and-run algorithm, instead of updating \vec{x} along one of the coordinate axes, we update it along a random general direction not necessarily parallel to any coordinate axis. More precisely, the sampler is defined in two steps: first, choose a direction \vec{n} from some positive density $\rho(\vec{n})$ on the unit sphere $\vec{n}^T \vec{n} = 1$. Then, similar to Gibbs, sample the new point on the line defined by \vec{n} and \vec{x} , with a density proportional to the underlying distribution. That is, we sample s from the one-dimensional density

$$s \sim h(s|\vec{n}, \vec{x}) \propto p(\vec{x} + s\vec{n}), \quad -\infty < s < \infty,$$

and then set $\vec{y} = \vec{x} + s\vec{n}$.

The main gain over RWM or HMC is that instead of taking small local steps (of size proportional to σ), we may take very large jumps in the \vec{n} direction; the jump size is set by the underlying distribution itself, not an arbitrary scale σ . This, together with the fact that all hit-and-run proposals are accepted, makes the chain better at escaping from sharp high-dimensional corners (see (Lovasz and Vempala, 2003) and the discussion at the end of Sec. 6.2.2 above). Moreover, the algorithm is parameter-free, once we choose the direction sampling distribution $r(\cdot)$, unlike the random walk case, where we have to choose the jump size somewhat carefully. The advantage over Gibbs is in situations such as depicted in Fig. 1, where jumps parallel to coordinates lead to small steps but there are directions that allow long jumps to be made by hit-and-run. The price to pay for these possibly long nonlocal jumps, however, is that now (as well as in the Gibbs case) we need to sample from the one-dimensional density $s \sim \frac{1}{Z}p(\vec{x} + s\vec{n})$, which is in general non-trivial. This may be done by rejection sampling, the cumulative inverse transformation, or one-dimensional Metropolis. If the posterior distribution is log-concave, then so are all of its ‘‘slices,’’ and very efficient specialized methods such as slice sampling (Neal, 2003) or adaptive rejection sampling (Gilks and Wild, 1992) can be used; see (Ahmadian et al., 2008) for a discussion of the adaptive rejection sampling method, and section 7.6 below for a brief discussion of the slice sampling method.

To establish the validity of the hit-and-run sampler, we argue as in the Gibbs case: we compute the ratio

$$\frac{p(\vec{y})q(\vec{x}|\vec{y})}{p(\vec{x})q(\vec{y}|\vec{x})} = \frac{p(\vec{y}) \int \rho(\vec{n})q(\vec{x}|\vec{y}, \vec{n})d\vec{n}}{p(\vec{x}) \int \rho(\vec{n})q(\vec{y}|\vec{x}, \vec{n})d\vec{n}} = \frac{\int \rho(\vec{n})p(\vec{y})q(\vec{x}|\vec{y}, \vec{n})d\vec{n}}{\int \rho(\vec{n})p(\vec{x})q(\vec{y}|\vec{x}, \vec{n})d\vec{n}} = 1,$$

since $p(\vec{x})q(\vec{y}|\vec{x}, \vec{n}) = p(\vec{y})q(\vec{x}|\vec{y}, \vec{n}) \propto p(\vec{x})p(\vec{y})$ for any \vec{x} and \vec{y} connected by a line along the direction \vec{n} ; thus we see that the hit-and-run proposal is indeed a M-H proposal which is accepted with probability one.

Regarding the direction density, $\rho(\vec{n})$, the easiest choice is the isotropic $\rho(\vec{n}) = 1$. More generally it is easy to sample from ellipses, by sampling from the appropriate Gaussian distribution and normalizing. Thus, again, a reasonable approach is to exploit the Laplace approximation: we sample \vec{n} by sampling an auxiliary point \tilde{x} from $\mathcal{N}(0, H^{-1})$, and setting $\vec{n} = \tilde{x}/\|\tilde{x}\|$ (see Fig. 1). Sampling from $\mathcal{N}(0, H^{-1})$ may be performed via the efficient Cholesky techniques discussed in the previous sections. This non-isotropic direction density enhances hit-and-run’s advantage over Gibbs by giving more weight to directions that allow larger jumps to be made.

6.3 Comparing mixing speed in the log-concave case: for smooth densities, hybrid methods are best; for densities with ‘‘corners,’’ hit-and-run is more effective

In the last few sections, we have discussed some qualitative arguments supporting various MCMC algorithms, in terms of their mixing rates and computational costs. Here we give a more quantitative account, and also compare the different methods based on their performance in the neural decoding setup.

From a practical point of view, the most relevant notion of mixing is how fast the estimate \hat{g}_N converges to the true expectation of the quantity of interest, $E(g)$. As one always has access to finitely many samples, N , even in the optimal case of i.i.d. samples from p , \hat{g}_N has a finite random error, $(1/N)\text{Var}_p(g)$. For correlated samples from the MCMC chain, the asymptotic error is typically larger (since each sample gives somewhat redundant information), and the error may be written (Robert and Casella, 2005) as

$$\text{Var}(\hat{g}_N) = \frac{\tau_{\text{corr}}}{N} \text{Var}[g(\vec{x})] + o\left(\frac{\tau_{\text{corr}}}{N}\right) \quad (7)$$

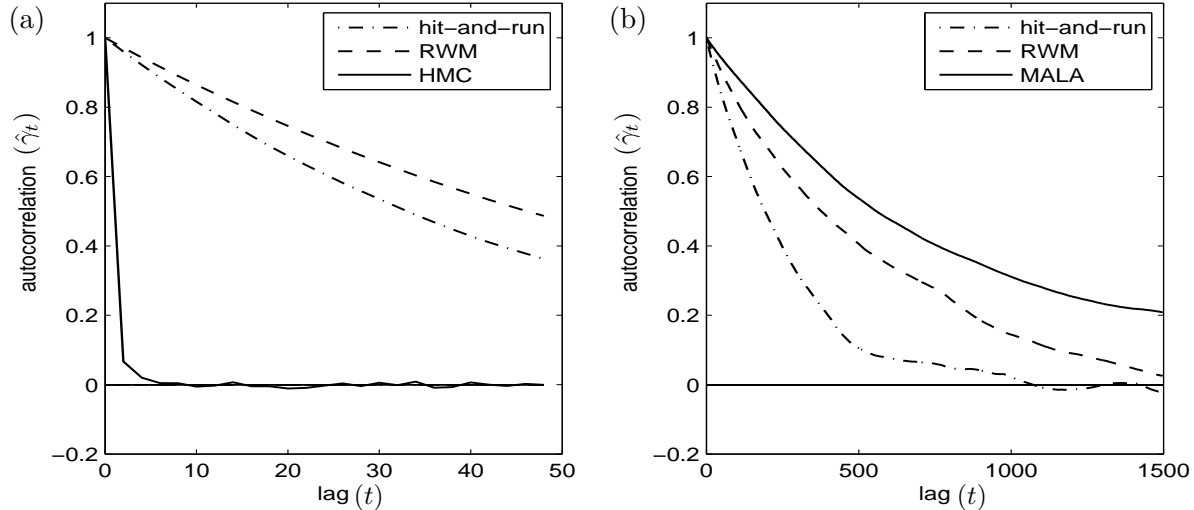


Figure 3: The estimated autocorrelation function for the hit-and-run, Gaussian random-walk metropolis, and HMC chains, based on 7 parallel chains. The chains were sampling from a posterior distribution over a 50-dimensional stimulus (\vec{x}) space with white noise Gaussian (a) and uniform (b) priors with equal marginal variance, and with GLM likelihood based on the response of two simulated retinal ganglion cells. The GLM nonlinearity was exponential and the stimulus filters $k_i(t)$ were taken to be “delta functions” (i.e., no temporal filtering was applied). For the HMC, we used $L = 5$ leapfrog steps in the Gaussian prior case (with σ chosen to obtain a M-H acceptance rate of 60%-70%), and $L = 1$ steps (corresponding to MALA) in the flat prior case. The autocorrelation was calculated for a certain one-dimensional projection of \vec{x} . In general, in the Gaussian prior case, HMC was superior by an order of magnitude. For uniform priors, however, hit-and-run was seen to mix faster than the other two chains over a wide range of parameters such as the stimulus filter strength (unless the filter was strong enough so that the likelihood determined the shape of the posterior, confining its effective support away from the edges of the flat prior). This is mainly because hit-and-run is better in escaping from the sharp, high-dimensional corners of the prior support. For both priors, using non-isotropic proposal or direction distributions improved the mixing of all three chains, and direct Gibbs sampling was seen to be relatively inefficient (data not shown; see (Ahmadian et al., 2008) for details).

for $N \gg \tau_{\text{corr}}$, where τ_{corr} is the equilibrium autocorrelation time of the scalar process $g(\vec{x}_i)$:

$$\tau_{\text{corr}} = \sum_{t=-\infty}^{\infty} \gamma_t \equiv \sum_{t=-\infty}^{\infty} \text{Corr}(g(\vec{x}_i)g(\vec{x}_{i+t})). \quad (8)$$

Thus the smaller the τ_{corr} , the more efficient is the MCMC algorithm, as one can run a shorter chain to achieve a desired expected error.

A slightly simpler measure of mixing speed is the mean squared jump size of the Markov chain

$$a^2 = E(\|\vec{x}_{t+1} - \vec{x}_t\|^2); \quad (9)$$

this has been termed the *first-order efficiency* (FOE) by (Roberts and Rosenthal, 2001). This FOE turns out to be closely related to the negative of the lag-1 autocorrelation γ_1 : maximizing the FOE is equivalent to minimizing this lag-1 correlation. Some helpful analytical results concerning the mixing performance of different MCMC chains were obtained in (Roberts and Rosenthal, 2001),

regarding sampling from some special classes of high-dimensional distributions. Based on their results, the authors argue that in general, the jump scales of RWM and MALA proposals should be chosen such that their acceptance rates are roughly 0.25 and 0.55, respectively. For the special case of sampling from a d dimensional standard Gaussian distribution, $p(\vec{x}) \propto \exp(-\|\vec{x}\|^2/2)$, and for optimally chosen proposal jump scales they show that the FOE of Gaussian MALA and RWM are asymptotically equal to $1.6d^{2/3}$ and 1.33, respectively, for large d ; in comparison, (Ahmadian et al., 2008) compute the FOE of Gaussian hit-and-run as 2. Therefore, while hit-and-run has higher FOE than RWM in this case, we see that for unimodal, nearly Gaussian distributions, MALA will mix much faster (by a factor $\propto d^{2/3}$) than both RWM and hit-and-run in large dimensions, and HMC will mix even faster (see (Kennedy and Pendleton, 1991; Neal, 1993) for further discussion of the mixing speed of HMC in this simple multivariate Gaussian case).

The superiority of HMC over the other chains is clearly visible in panel (a) of Fig. 3, which shows a plot of the estimated autocorrelation function γ_t for the sampling of the three chains from the GLM posterior with standard Gaussian priors. More generally, in simulations with Gaussian priors and smooth GLM nonlinearities, HMC (including MALA) had an order of magnitude advantage over the other chains for most of the relevant parameter ranges (Ahmadian et al., 2008).

The situation can be very different, however, for highly non-Gaussian (but still log-concave) distributions, such as those with sharp boundaries. In our GLM decoding setting this can be the case with flat stimulus priors on convex sets, when the likelihood is broad as a function of \vec{x} and does not restrict the posterior support away from the boundaries and corners of the set on which the prior is supported. In this case, HMC and MALA lose their advantage because they do not take advantage of the information in the prior distribution, which has zero gradient within its support. Furthermore, as mentioned in Secs. 6.2.2 and 6.2.4, when the convex body has sharp corners, hit-and-run will have an advantage over both RWM and HMC in avoiding getting trapped in those corners, which can otherwise considerably slow down the chain (see the arguments in (Lovasz and Vempala, 2003)).

Figure 3, panel (b), shows the estimated autocorrelation function for different chains in sampling from the posterior distribution in GLM-based decoding with a flat stimulus prior distribution with cubic support. For this prior, the correlation time of the hit-and-run chain was consistently lower than those of the RWM, MALA, and Gibbs (not shown in the figure) chains, unless the likelihood was sharp and concentrated away from the boundaries of the prior cube. As we mentioned above (also see the next section), the Laplace approximation is adequate in this latter case. Thus we see that hit-and-run is the faster chain when this approximation fails, which is also the case where MCMC is more indispensable.

Finally, we note that a number of other robust empirical methods of diagnosing mixing and convergence have been introduced in the literature; the so-called \hat{R} statistic discussed in (Gelman et al., 2003) is particularly popular. These alternate methods give consistent results with those based on the autocorrelation time, τ_{corr} , presented here.

6.4 Example: comparing MAP and Monte Carlo stimulus decoding given GLM observations

In this section we compare Bayesian stimulus decoding using the MAP and the MCMC posterior mean estimates, based on the response of a population of neurons modeled via the GLM. We will show that in the flat prior case, the MAP estimate is much less efficient (in terms of its mean squared error) than the posterior mean estimate. We contrast this with the Gaussian prior case, where the Laplace approximation is accurate over a large range of model parameters, and thus the two estimates are close. Furthermore, for both kinds of priors, in the limit of strong likelihoods

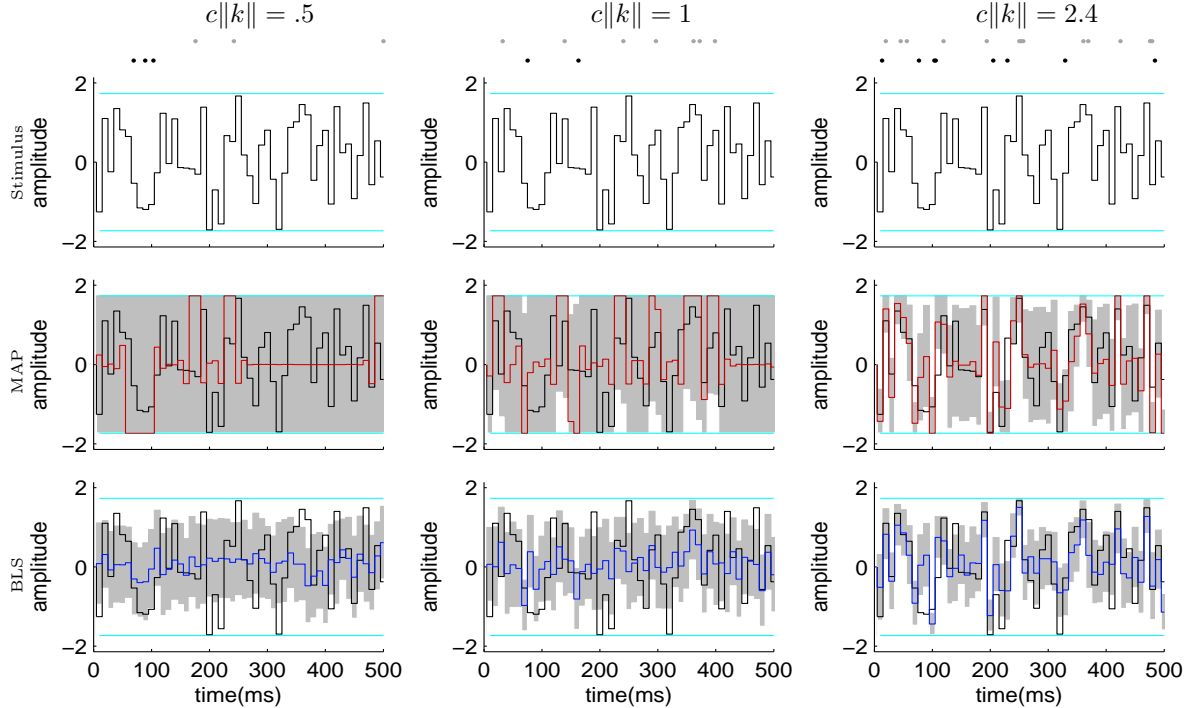


Figure 4: Comparison of MAP and posterior mean estimates, for a pair of ON and OFF model neurons, for different values of the stimulus filter amplitude ($\|k\| = 0.5, 1, \text{ and } 2.4$; i.e., the dependence of the firing rate on the stimulus was increasing from left to right). The stimulus (the black trace shown on all panels) consists of a 500 ms interval of uniformly distributed white noise, refreshed every 10 ms. Thus the stimulus space is 50 dimensional. The cyan horizontal lines mark the boundaries of the flat prior distribution of the stimulus intensity on each 10 ms subinterval. They are set at $\pm\sqrt{3}$, corresponding to intensity variance of 1 and zero mean. Dots on the top row show the spikes of the ON (gray) and the OFF (black) cell. The red traces in the middle row are the MAP estimates, and the blue traces in the bottom rows show the posterior means estimated from 10000 samples of a hit-and-run chain (after burning 2500 samples). The shaded regions in the second and third rows are error bars representing the estimated marginal posterior uncertainty about each stimulus value for the MAP and mean estimates. For the MAP (second rows), these are calculated as the square root of the diagonal of the inverse Hessian H^{-1} , but they have been cut-off where they would have encroached on the zero prior region beyond the horizontal cyan lines. For the posterior mean (third rows), they represent one standard deviation about the mean, calculated as the square root of the diagonal of the covariance matrix, itself estimated from the MCMC chain (the standard error of the posterior mean estimate due to the finite samples of the MCMC were much smaller than these error bars, and are not shown). Note that the errorbars of the mean are in general smaller than those for the MAP, and that all estimate uncertainties decrease as the stimulus informativeness grows.

(e.g., due to a strong stimulus filter or a large number of neurons) the posterior distribution will be sharply concentrated, the Laplace approximation becomes asymptotically more and more accurate, and both estimates will eventually converge to the true stimulus (more precisely the part of the stimulus that is not outside the receptive field of all the neurons; see footnote 5, below).

In the first two examples (Figs. 4–5), the stimulus estimates were computed given the simulated spike trains of a population of pairs of ON and OFF retinal ganglion cells (RGC), in response

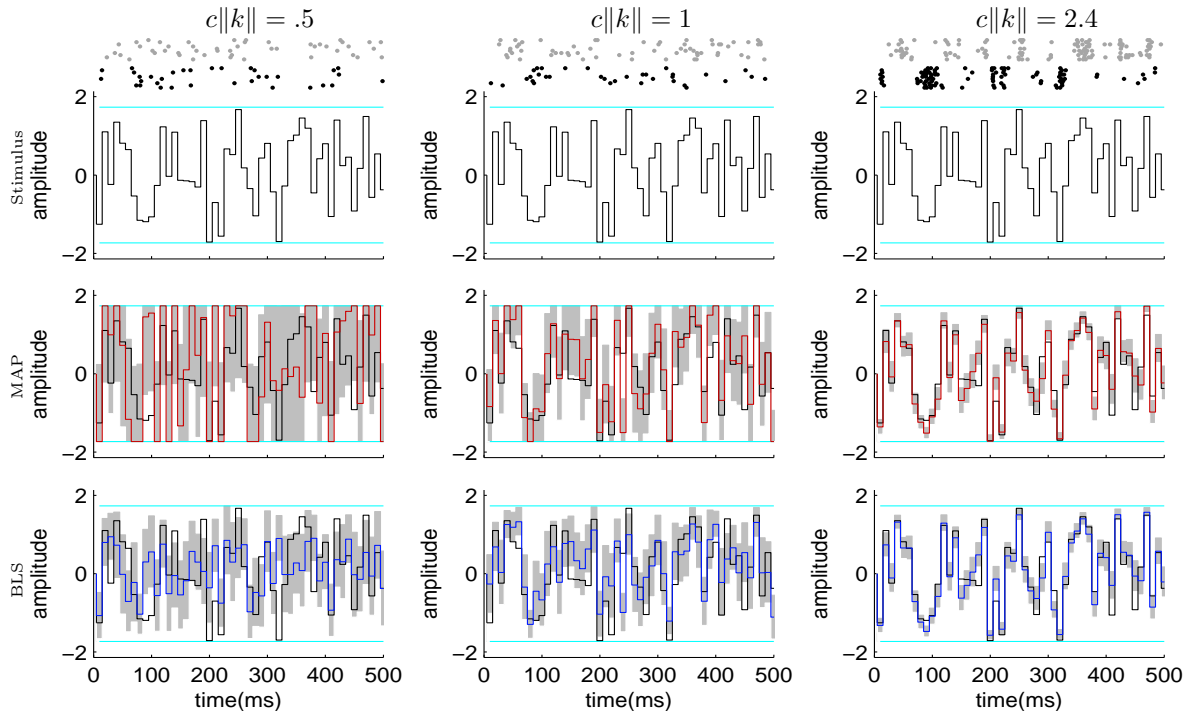


Figure 5: Comparison of MAP and posterior mean estimates, for 10 identical and independent pairs of ON and OFF model neurons, for different values of the stimulus filter. The stimulus and all GLM parameters are the same as in Fig. 4, except for the number of pairs of RGC. The increase in the number of cells leads to the sharpening of the likelihood, leading to smaller error bars on the estimates, and a more accurate Laplace approximate and smaller disparity between the two estimates. Here a 20000 sample long MALA chain (after burning 5000 samples) was used to estimate the posterior mean.

to a spatially uniform, full-field fluctuating light intensity signal. The stimuli were discretized, i.i.d. draws from Gaussian and flat distributions, respectively. Spike responses were generated by simulating the GLM point process encoding model with no temporal filtering (i.e., only the stimulus at time t directly affected the responses at time t); see (Ahmadian et al., 2008) for details. Figure 4 shows the stimulus, the spike trains, and the two estimates for three values of the stimulus filter amplitude based on the response of one pair of model ON and OFF cells. Figure 5 shows the same based on the response of ten identical pairs of model cells.

Because the prior distribution here is flat on the 50-dimensional cube centered at the origin, the Laplace approximation will be justified only when the likelihood is sharp and supported away from the edges of the cube⁴. Moreover, since the flat prior is only “felt” on the boundaries of the cube (the cyan lines in Figs. 4-5), the MAP will lie in the interior of the cube only if the likelihood has a maximum there. When the dependence of the firing rate on the stimulus is small, the log-likelihood becomes approximately linear in the components of \vec{x} . With a flat prior, the near-linear log-posterior will very likely be maximized only on the boundaries of the cube. Thus in the absence of a strong, confining likelihood, the MAP has a tendency to stick to the boundaries, as seen in the first two columns of Fig. 4; in other words, the MAP falls on a corner of the cube,

⁴More precisely, “sharp” here means that the curvature of the log-posterior is large enough so that the Taylor expansion of the log-posterior involved in the Laplace approximation is accurate for deviations from \vec{x}_{MAP} on a scale determined by the inverse square root of the smallest eigenvalue of the Hessian matrix.

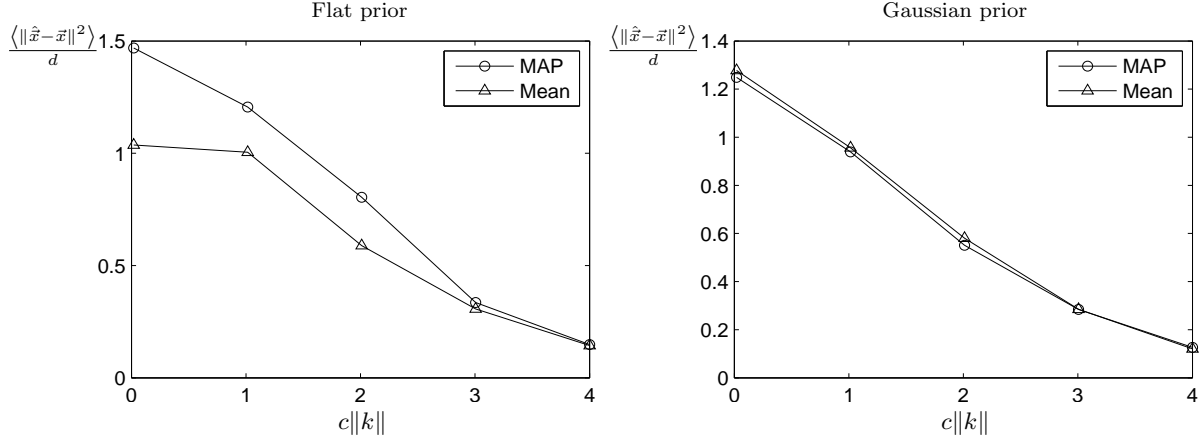


Figure 6: Comparison of mean squared error ($\langle \|\hat{x} - \bar{x}\|^2 \rangle / d$) of MAP and posterior mean estimates for uniform (left panel) and Gaussian (right panel) white-noise stimulus distributions as a function of the stimulus filter strength. In the left panel, the data points at $\|k\| = 0$ were obtained for very small but non-zero $\|k\|$. As seen here, for flat priors, the MAP estimate has a higher average squared error than the posterior mean, except for large values of the stimulus filter where both estimates converge to the true value. For Gaussian priors, on the other hand, the Laplace approximation is accurate and therefore the posterior mean and MAP are very close; thus the MAP estimate is preferable here, since it is cheaper to compute than the posterior mean.

where the Laplace approximation is worst and where MALA and RWM are least efficient. We note that the likelihood is further weakened when we replace the delta function stimulus filters with more realistic filters, since band-pass filtering will remove the dependence of the likelihood on the stimulus subspace orthogonal to the filters.

On the other hand, a sharp likelihood confines the posterior away from the boundaries of the prior support, and solely determines the position of both the MAP and the posterior mean. In this case the Gaussian approximation for the posterior distribution is valid and the two estimates will in fact be very close (as the mean and the mode of a Gaussian are one and the same). This can be seen in the right column of Fig. 4, where the large value of the stimulus filter has sharpened the likelihood. Also, as is generally true in statistical parameter estimation, when the number of data points becomes large the likelihood term gets very sharp, leading to accurate estimates⁵. In our case this corresponds to increasing the number of cells with similar receptive fields, leading to the smaller error bars in Fig. 5 and the more accurate and closer MAP and mean estimates.

To compare the performance of the two estimates more quantitatively, we have plotted the average squared errors of the two estimates, for the same type of stimulus and cell pair as in Fig. 4. The left and right panels in Fig. 6 show plots of the squared error per dimension, for MAP and mean estimates, as a function of the stimulus filter strength for the case of the flat and Gaussian white-noise stimulus ensembles, respectively. As is evident from the plots, in the former ensemble, the MAP is inferior to the mean, due to its higher mean squared error, unless the filter strength is large. For the Gaussian ensemble, the plot shows that the error of the MAP and posterior mean estimates are very close, throughout the range of stimulus filter strength. Thus, due to its much lower computational cost, the MAP-based decoding method is superior for this prior.

⁵This is obviously not the case, however, for parameter directions along which the data is non-informative, and the likelihood function does not vary much. In the RGC case, these correspond to stimulus features (directions in the stimulus space) that fall orthogonal to the cells' spatiotemporal filters.

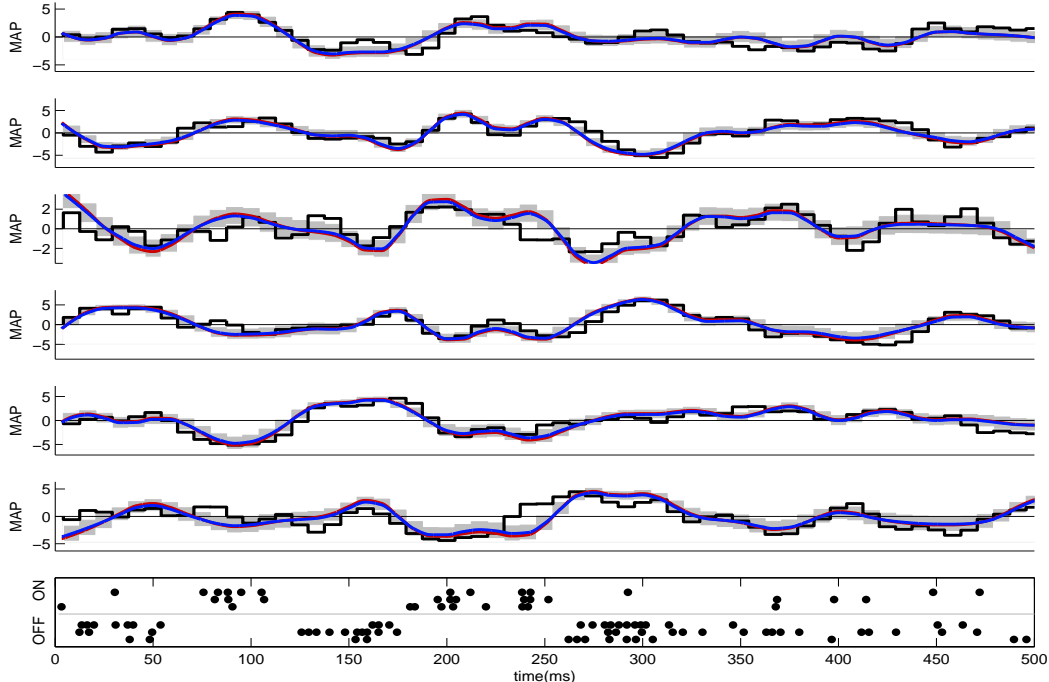


Figure 7: The top six panels show the MAP (red traces) and posterior mean (blue traces) estimates of the stimulus input to 3 pairs of ON and OFF RGC’s given their spike trains from multi-electrode array recordings. The jagged black traces are the actual inputs. The bottom panel shows the recorded spike trains. The posterior means were estimated using an HMC chain with 15000 samples (after an initial 3750 samples were burnt). The gray error-bars around the blue curve are represent its marginal standard deviations which were estimated using the MCMC itself (the error-bars for the MAP based on the Hessian would not be distinguishable in this figure, and are not shown). The closeness of the posterior mean to the MAP is an indication of the accuracy of the Laplace approximation in this case.

Finally, we compared the MAP and posterior mean estimates in decoding of experimentally recorded spike trains. The spike trains were recorded from a group of 11 ON and 16 OFF RGCs (whose receptive fields fully cover a patch of the visual field) in response to presentations of 20 minutes of a spatiotemporally fluctuating binary white-noise light signal, \vec{x} , to the retina. In (Pillow et al., 2008), these data were used to fit the GLM model parameters including cross-couplings, h_{ij} — see that reference for details about the recording, fitting method, and the properties of the fit GLM parameters. Here, we took a 500 ms portion of the recorded spike trains of 6 neighboring RGCs (3 ON and 3 OFF), and using the fit GLM parameters, decoded the filtered inputs,

$$\vec{y}_i(t) \equiv \vec{k}_i \cdot \vec{x}(t), \quad (10)$$

to these cells using the MAP and posterior mean (computed via HMC).

To perform the decoding efficiently here, we need to adapt our non-isotropic $O(T)$ approach somewhat. First we need to discuss the form of our prior $p(\vec{y})$ in this case. The inputs \vec{y}_i are *a priori* correlated due to the overlaps between the cell’s receptive fields, and the covariance matrix of the \vec{y}_i is given by $\mathcal{C}_y^{ij} = \vec{k}_i \mathcal{C}_x \vec{k}_j^T$, where $\mathcal{C}_x = c^2 \mathbf{1}$ is the covariance of the white-noise visual stimulus.

More explicitly

$$\mathcal{C}_y(i, t_1; j, t_2) \equiv \text{Cov}[y_i(t_1), y_j(t_2)] = c^2 \sum_{t,n} \vec{k}_i(t_1 - t, n) \vec{k}_j(t_2 - t, n). \quad (11)$$

Notice that with the experimentally fit \vec{k}_i , which have a finite temporal duration T_k , the covariance matrix, \mathcal{C}_y is banded: it vanishes when $|t_1 - t_2| \geq 2T_k - 1$. Since the spatiotemporal stimulus \vec{x} is binary, \vec{y}_i is not a Gaussian vector. However, because the filters $\vec{k}_i(t, n)$ have a relatively large spatiotemporal dimension, $\vec{y}_i(t)$ are weighted sums of many independent identically distributed binary random variables, and their prior marginal distributions can be well approximated by Gaussian distributions. For this reason, and because the likelihood was relatively strong for this data (and hence the dependence on the prior relatively weak), we replaced the true (highly non-Gaussian) joint prior distribution of \vec{y}_i with a Gaussian distribution with zero mean and covariance Eq. (11).

Now the Hessian for the negative log-posterior on \vec{y} is given by

$$H_y = \mathcal{C}_y^{-1} + J_y, \quad (12)$$

where the Hessian of the negative log-likelihood term, J_y , is now diagonal, because $y_i(t)$ affects the conditional firing rate instantaneously. Let $AA^T = H_y^{-1}$, similar to Eq. (4). The non-isotropic chain requires the calculation of $A\tilde{\vec{y}}$ for some vector $\tilde{\vec{y}}$ at each step of the MCMC. In order to carry this out in $O(T)$ computational time, we proceed as follows. First we calculate the Cholesky decomposition, L , of \mathcal{C}_y , satisfying $LL^T = \mathcal{C}_y$. As mentioned in Sec. 6.2.1, since \mathcal{C}_y is banded this can be performed in $O(T)$ operations. Then we can rewrite Eq. (12) as

$$H_y = L^{-1T}QL^{-1}, \quad Q \equiv \mathbf{1} + L^T J_y L. \quad (13)$$

Since L is banded (due to the bandedness of \mathcal{C}_y) and J_y is diagonal, it follows that Q is also banded. Therefore its Cholesky decomposition, B , satisfying $B^T B = Q$, can be calculated in $O(T)$ time, and is also banded. Using this definition and inverting Eq. (13), we obtain $AA^T = H_y^{-1} = LB^{-1}(LB^{-1})^T$, from which we deduce $A = LB^{-1}$, or

$$A\tilde{\vec{y}} = LB^{-1}\tilde{\vec{y}}. \quad (14)$$

The calculation of L and B can be performed before running the HMC chain. Then at each step we need to perform Eq. (14). As described in the final paragraph of Sec. 6.2.1, calculating $B^{-1}\tilde{\vec{y}}$ and the multiplication of the resulting vector by L , both require only $O(T)$ elementary operations due to the bandedness of B and L . Similarly, the computation of the MAP requires just $O(T)$ time here.

Figure 7 shows the spike trains, as well as the corresponding true inputs and MAP and posterior mean estimates. The closeness of the posterior mean to the MAP (the L_2 norm of their difference is only about 9% of the L_2 norm of the MAP) is an indication of the accuracy of the Laplace approximation in this case.

7 Gibbs sampling is useful in a wide array of applications

As we emphasized above, direct Gibbs sampling is typically not the most efficient way to sample from a log-concave density. However, it would be a major oversight to dismiss this algorithm out of hand; not all densities of interest are log-concave, of course, and in fact Gibbs is easily the most

widely used M-H algorithm. This is because in many models of interest, the posterior distribution can be decomposed into a few conditional densities which are relatively tractable. In these cases, the Gibbs idea is natural, often easy to code⁶, and extremely useful, especially when combined with the fast-mixing M-H methods described above. We discuss a number of examples in this section.

7.1 Example: sampling network spike trains given the observed activity of a subset of neurons in the population

As a first example, imagine sampling the from a population of coupled GLM neurons given the full stimulus sequence X and observed data $D = \{n_{i,t}\}$ consisting of spike counts from some of the neurons at some subset of times $\{t_i\} \in [0, T]$. Recall our coupled GLM model:

$$n_{i,t} \sim Poiss(\lambda_i(t)dt); \quad \lambda_i(t) = f\left(b + \vec{k}_i \cdot \vec{x}(t) + \sum_{i',j>0} h_{i',i}(j)n_{i',t-j}\right); \quad (15)$$

here $h_{i',i}(j)$ represents the influence that cell i' has on cell i , with a delay of j time steps. Now, if we have observed all spike counts from time 0 up to time t , then sampling forward in time is straightforward, either using the time-rescaling technique or by recursively sampling $n_{i,t}$ given the full collection of past spikes $\{n_{i',t'}\}_{t'<t}$ in discrete time, since the spike counts here form a kind of autoregressive process. Thus, iterative MCMC methods are not required in this case.

However, as discussed by (Pillow and Latham, 2007), it is more difficult to sample from $n_{i,t}$ given incomplete past or future information about the other spike counts $n_{i',t'}$, because now we can no longer apply simple direct recursive methods to sample from $n_{i,t}$, and iterative MCMC methods are required instead. We may apply Gibbs here if we just write down the necessary conditional probabilities:

$$\begin{aligned} p(n_{i,t} = c | X, \theta, \{n_{i',t'}\}_{(i',t') \neq (i,t)}) &= \frac{p(n_{i,t} = c, \{n_{i',t'}\}_{(i',t') \neq (i,t)} | X, \theta)}{p(\{n_{i',t'}\}_{(i',t') \neq (i,t)} | X, \theta)} \\ &= \frac{1}{Z} \frac{\exp(-\lambda_i(t)) \lambda_i(t)^c}{c!} \prod_{i',t'} \exp(-\lambda_{i'}(t')) \lambda_{i'}(t')^{n_{i',t'}}, \end{aligned}$$

where the product is taken over all cells i' and times t' for which $\lambda_{i'}(t')$ is affected by $n_{i,t}$. Since it is straightforward to compute and sample from these discrete one-dimensional conditional probabilities, we may now simply apply Gibbs (following some random or deterministic schedule for visiting the nodes (i', t')) to sample from the multivariate spike train.

However, the Gibbs chain can mix quite slowly in this case, since strong dependencies (e.g., refractory effects) typically exist in the variables $n_{i,t}$. We will discuss more efficient methods below.

7.2 Blockwise sampling is often more efficient than Gibbs sampling each variable individually

As emphasized above, the Gibbs sampling Markov chain will in general mix slowly if the sampled variables are highly dependent. In some cases we can develop a more efficient sampling procedure by applying the Gibbs idea to “blocks” of variables that we sample jointly. As a concrete example, let’s return to the case we described in the last subsection. A natural method of “blocking” the spike count variables $n_{i,t}$ is to sample from one neuron at a time, holding all of the spike counts in

⁶The BUGS (Bayesian inference Using Gibbs Sampling) project has been extremely influential in providing easy-to-use Gibbs sampling software for the general public.

the other neurons fixed. The Gibbs sampling theory developed above now tells us that if we can sample from the conditional distributions $p(n_i|\{n_{i'}\}_{i \neq i'})$ (where n_i denotes the full spike train of neuron i), then we can sample from the full desired joint distribution $p(\{n_i\})$.

When should we use this blockwise Gibbs sampler, as opposed to the individual Gibbs sampler described in the last section? This depends on how efficiently we can sample from the blocks n_i : if we can perform the blockwise sampling from $p(n_i|\{n_{i'}\}_{i \neq i'})$ about as efficiently as the individual sampling from $p(n_{i,t}|\{n_{i',t'}\}_{(i',t') \neq (i,t)})$, then typically the blockwise Markov chain will require fewer iterations to mix adequately, and we should use the blockwise approach.

Unfortunately, exact blockwise sampling from $p(n_i|\{n_{i'}\}_{i \neq i'})$ is often intractable. Thus it makes sense to use an “inner” Metropolis-Hastings loop to sample from $p(n_i|\{n_{i'}\}_{i \neq i'})$, within an “outer” Gibbs loop over blocks i . Since both the inner and outer loops are Metropolis-Hastings samplers (and therefore respect detailed balance for $p(\{n_i\})$, by construction), this Metropolis-Hastings-within-Gibbs algorithm is still guaranteed to provide correct MCMC samples from $p(\{n_i\})$.

(Pillow and Latham, 2007) used a blockwise independence sampler for $p(n_i|\{n_{i'}\}_{i \neq i'})$, and introduced a proposal distribution for n_i that takes into account the full activity of the other neurons $n_{i'}$, but which nonetheless permits us to sample from n_i in an efficient, forward recursive fashion. Their idea was to modify the GLM so that both the past and the future of the other neurons’ activity $n_{i'}$ was included as a covariate. In other words, the usual GLM (equation (15)) was modified to

$$\lambda_i^{PL}(t) = f\left(b + \vec{k}_i \cdot \vec{x}(t) + \sum_{j>0} h_{i',i}(j)n_{i,t-j} + \sum_{i' \neq i, j \leq 0} h_{i,i'}(j)n_{i',t-j}\right);$$

they found that this proposal led to a faster-mixing chain.

Inclusion of the extra terms $h_{i,i'}(j)n_{i',t-j}$ for $j \leq 0$ in the model requires that we re-fit all of the GLM parameters (including the new parameters $h_{i,i'}(j)$), but this is rather inefficient. It is also unclear whether this simple GLM form represents a systematic approximation to the correct conditional blockwise sampler in the regime of interest. Thus it is natural to ask whether we can analytically derive an efficient recursive proposal which will serve as a good approximation to the exact conditional sampler. Let’s begin by examining the case of just two linked Poisson variables n_1 and n_2 : imagine that

$$\begin{aligned} n_1 &\sim \text{Poiss}[f(b_1)] \\ n_2|n_1 &\sim \text{Poiss}[f(b_2 + an_1)], \end{aligned}$$

where a is a coupling parameter, which we will assume is small here. We may compute the conditional distribution of n_1 given n_2 :

$$\begin{aligned} \log p(n_1|n_2) &= \log p(n_1, n_2) + \text{const.} \\ &= \log p(n_1) + \log p(n_2|n_1) + \text{const.} \\ &= n_1 \log f(b_1) - f(b_1) - \log n_1! + n_2 \log f(b_2 + an_1) - f(b_2 + an_1) - \log n_2! + \text{const.} \\ &= n_1 \log f(b_1) - \log n_1! + n_2 \left[\log f(b_2) + an_1 \frac{f'}{f}(b_2) + o(a) \right] - f(b_2) - an_1 f'(b_2) - o(a) + \text{const.} \\ &= n_1 \left[\log f(b_1) + a \left(n_2 \frac{f'}{f}(b_2) - f'(b_2) \right) + o(a) \right] - \log n_1! + \text{const.} \\ &= n_1 \left[\log f(b_1) + a \frac{f'}{f}(b_2) (n_2 - f(b_2)) + o(a) \right] - \log n_1! + \text{const.} \end{aligned}$$

where *const.* denotes any term that is constant in n_1 . If we drop the second-order terms in a here, we are left with a Poisson distribution with log-rate

$$\log E(n_1|n_2) \approx \log f(b_1) + a \frac{f'}{f}(b_2)(n_2 - f(b_2)),$$

which is fairly intuitive: we adjust the rate in bin 1 by a term proportional to the deviation of n_2 from its expectation $f(b_2)$, multiplied by the coupling weight a .

These computations generalize in a straightforward manner to our GLM case as long as all the coupling terms $h_{i,j}$ are small. In the canonical case $f(\cdot) = \exp(\cdot)$, the formulas simplify somewhat (since the logarithmic derivative $f'/f(\cdot) = 1$), and we see that a natural proposal is to sample forward recursively from

$$\log \lambda_i^*(t) = \log \lambda_i(t) + \sum_{i' \neq i, j > 0} h_{i,i'}(j) (n_{i',t+j} - \lambda_{i'}(t+j) dt),$$

where $\lambda_i^*(t)$ denotes the proposed rate and $\lambda_i(t)$ is defined in the usual way, as in equation (15). (Note that this formula does not require us to define any new coupling filters $h_{i,i'}(j)$, for $j < 0$; the future behavior of neurons $i' \neq i$ is handled naturally by incorporating the effects of the known $h_{i,i'}(j)$ terms.) Once this proposed sample is obtained, we may compute the Metropolis-Hastings acceptance probability easily. Since this proposal corresponds to a first-order approximation of the exact blockwise Gibbs sampler, the acceptance probability here tends to one in the weak-coupling limit $h \rightarrow 0$.

Of course, in real neurons the self-coupling terms h are typically not small, due to strong refractory effects, and therefore this weak-coupling analysis is of questionable generality. We will introduce more powerful methods for conditional sampling from spike trains later, after we have discussed some theory for hidden Markov models.

7.3 Example: computing errorbars in low-rank stimulus encoding models

We may also apply Gibbs (or blockwise Gibbs) sampling to construct errorbars of parameter estimates. A useful example arises in the context of “separable” receptive fields of the form

$$f(\vec{x}, t) = f_t(t) f_{\vec{x}}(\vec{x}).$$

We previously described a simple alternating optimization method for fitting such a separable model; it is natural to ask how to efficiently quantify the variability of our estimates in this case. As usual, we may simply compute the Hessian of the log-likelihood at the MAP estimate to construct approximate errorbars. But if we want exact errorbars a Gibbs approach is natural (Ahrens et al., 2008). In the case of a Gaussian model,

$$y_i = \sum_{jl} a_j b_l X_{ijl} + \sigma \epsilon_l,$$

with X the appropriate design matrix and ϵ assumed standard i.i.d. normal for simplicity, we may do Gibbs sampling exactly: given \vec{b} we sample from the Gaussian posterior on \vec{a} ,

$$p(\vec{a}|D, \vec{b}) = \mathcal{N}((Z^T Z + \sigma^2 C_a)^{-1}(Z^T Y), (Z^T Z + \sigma^2 C_a)^{-1} \sigma^2 (Z^T Z)(Z^T Z + \sigma^2 C_a)^{-1})$$

and vice versa

$$p(\vec{b}|D, \vec{a}) = \mathcal{N}((W^T W + \sigma^2 C_b)^{-1}(W^T Y), (W^T W + \sigma^2 C_b)^{-1} \sigma^2 (W^T W)(W^T W + \sigma^2 C_b)^{-1}),$$

where the matrices Z and W are formed by multiplying X against \vec{b} and \vec{a} , respectively — i.e.,

$$Z_{ij} = \sum_l b_l X_{ijl}$$

and

$$W_{il} = \sum_j a_j X_{ijl}$$

— and C_a and C_b denote the prior covariance matrices of \vec{a} and \vec{b} , respectively; we have assumed that these priors are zero-mean Gaussian, for simplicity. (Interestingly, (Salakhutdinov and Mnih, 2008) apply a similar model to the machine learning problem of ranking user preferences in large databases.) In the case that the observations D are from a GLM instead of the simpler linear model, to sample from the posteriors $p(\vec{a}|D, \hat{b}^i)$ and $p(\vec{b}|D, \hat{a}^i)$ we need to apply the Metropolis-Hastings algorithm in a subloop to compute each step of the Gibbs method; this may be done using any of the M-H proposals discussed above.

This case provides a nice illustration of the Rao-Blackwellization idea we introduced in section 5. To construct errorbars in this case, we have two options: first, we can compute the marginal variance computed from a sequence of samples \vec{a}^i and \vec{b}^i obtained by alternating between $p(\vec{a}|D, \vec{b})$ and $p(\vec{b}|D, \vec{a})$, as described above. This effectively represents our joint distribution $p(\vec{a}, \vec{b}|D)$ in terms of a normalized sum of “particles,”

$$p(\vec{a}, \vec{b}|D) \approx (1/N) \sum_{i=1}^N \delta(\vec{a} - \vec{a}^i) \delta(\vec{b} - \vec{b}^i).$$

But this is inefficient, since for any \vec{a}^i , we have an analytical formula for the conditional mean and covariance of \vec{b} , and vice versa; the Rao-Blackwell theorem tells us that we should keep track of these conditional expectations instead of noisy, sampled versions of these quantities. Thus, instead of just recording the samples (\vec{a}^i, \vec{b}^i) , we should keep track of the conditional means and variances $(E(\vec{b}|\vec{a}^i), Cov(\vec{b}|\vec{a}^i))$ (and vice versa), which we computed above. This replaces our simple sum-of-particle representation with a mixture-of-Gaussians representation for $p(\vec{a}, \vec{b}|D)$ (since each of the conditionals $p(\vec{b}|\vec{a}^i)$ and $p(\vec{a}|\vec{b}^i)$ are Gaussian, as emphasized above). Since the mixture of Gaussians is inherently a more informative and more accurate representation of $p(\vec{a}, \vec{b}|D)$, estimates of the marginal variance based on the Rao-Blackwellized samples $(E(\vec{b}|\vec{a}^i), Cov(\vec{b}|\vec{a}^i))$ will be more accurate (and require a smaller number of samples N) than the cruder particle-based approach.

7.4 Example: Bayesian adaptive regression splines for time-varying firing rate estimation

We have previously discussed the Bayesian adaptive regression splines (BARS) technique for estimating time-varying firing rates given noisy observations (DiMatteo et al., 2001; Kass et al., 2003; Behseta et al., 2005; Wallstrom et al., 2007). Recall that the observed data here are modeled as Poisson with rate $\lambda(t|\theta, X) = f(X_t, \theta)$, where X is a matrix of spline functions and θ is a vector of spline coefficients. The BARS approach is based a hierarchical model that places a prior on the spline knots (thus implicitly imposing a prior on the spline matrix X) and the coefficients θ . So to compute

$$E(\lambda(t)|D) = E_{p(\theta, X|D)} f(X_t, \theta)$$

here we can simply perform Gibbs sampling on $p(\theta, X|D)$: alternately draw θ^i from $p(\theta|D, X^i)$ and then X^i from $p(X|D, \theta^i)$. In this model $p(\theta|D, X)$ is log-concave with banded Hessian for any X ,

and therefore the efficient HMC methods discussed in section 6.2.2 provide an effective approach for sampling from the conditionals $p(\theta|D, X^i)$; see (Ahmadian et al., 2008) for details. Sampling from $p(X|D, \theta^i)$ requires some more specialized techniques which are beyond the scope of this chapter; see (DiMatteo et al., 2001; Wallstrom et al., 2007).

7.5 Example: estimating hierarchical network spike train models

A much more elaborate example of the use of Gibbs sampling in the context of parameter estimation appears in (Rigat et al., 2006). The authors introduce a hierarchical model for the activity in a network of neurons; there is a prior on connectivity matrices, a GLM for the spike counts given the connectivity matrices, and a model for the observed spike counts given the true spike counts (this observation model takes into account the fact that some spikes are dropped and some false spikes may be added, due to low signal-to-noise in the voltage recordings). Each level of the model — connectivity matrices, true spike counts, and observed spike counts — may be sampled via Gibbs (in some cases M-H is required to perform the Gibbs step), given that the other parameters and data are fixed. See (Rigat et al., 2006) for further details.

7.6 Example: “slice sampling” provides another easy way to sample from a unimodal density

A nice mathematical example of the Gibbs sampling idea is known as “slice sampling” (Neal, 2003). Imagine we would like to sample from a density $p(x)$ (let x be one-dimensional, for simplicity, although the idea can be generalized easily). One way to accomplish this would be to sample uniformly from the *two-dimensional* uniform density

$$(x, y)^i \sim p(x, y) = \frac{1}{Z} 1(0 < y < p(x)),$$

and then simply ignore the y component (i.e., marginalize over the y variable). Sampling directly from $p(x, y)$ is difficult, but it is easy to apply Gibbs here: we alternately sample from

$$p(y|x^i) = \frac{1}{Z} 1(0 < y < p(x^i)),$$

the uniform density on $y \in [0, p(x^i)]$, and $p(x|y^i) = \frac{1}{Z} 1(x \in C_{y^i})$, where C_y is the set of points x such that $p(x) \geq y$. (The term “slice sampling” comes from this latter step — to construct the level set C_y we need to take a “slice” through the density $p(x)$, where the slice is drawn at a vertical height y .) The first step is always trivial, but sampling from C_y can be harder, depending on how complicated the set C_y is. However, in the special case that $p(x)$ is a unimodal (e.g., log-concave) density, C_y is always a nice convex set, and rejection sampling from this set becomes fairly straightforward; see (Neal, 2003) for details. One-dimensional slice sampling is often used in the inner loop of a Gibbs or hit-and-run sampler, though we found adaptive rejection sampling (Gilks and Wild, 1992) to be more efficient in the decoding applications discussed in this chapter (Ahmadian et al., 2008).

7.7 Example: decoding in the presence of model uncertainty

In the previous sections we assumed the values of the model parameters involved in the GLM likelihood were known exactly. Of course, in reality these parameters themselves are obtained by fitting the GLM to experimental data, and thus are only known with a finite accuracy; we would like to know how this uncertainty affects the posterior mean estimate for the stimulus. We represent

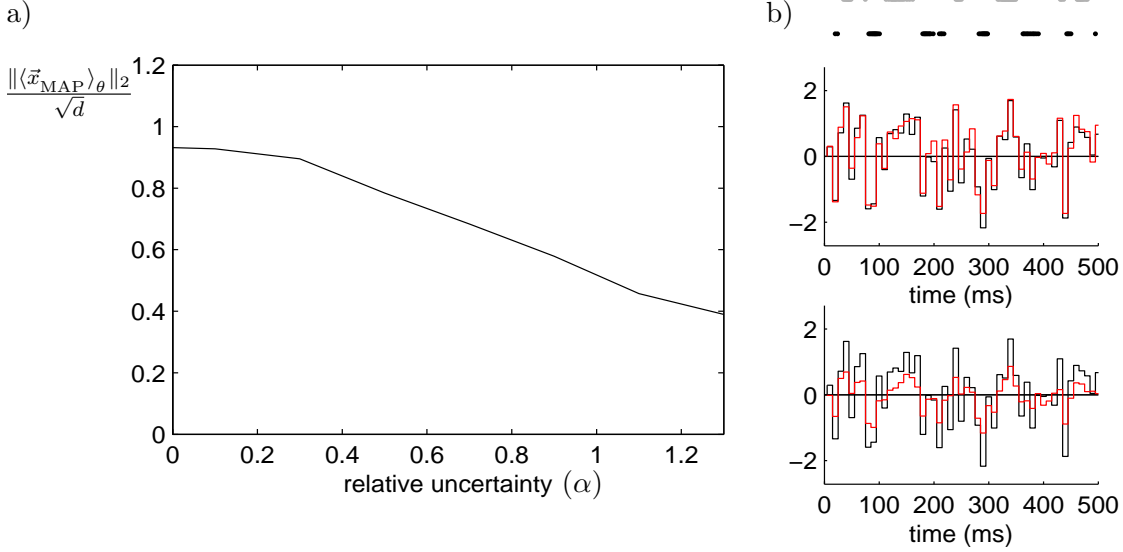


Figure 8: Effect of parameter uncertainty on the posterior estimate for Gaussian white-noise stimuli. Panel (a) is a plot of $\|\langle \vec{x}_{\text{MAP}} \rangle_{\theta}\|_2 / \sqrt{d}$ (where $d = 50$ is the stimulus dimension) versus relative uncertainty, α , in the stimulus filter $k_i(t)$. α is defined through $k_i(t) = (1 + \alpha \varepsilon(t)) \bar{k}_i(t)$, where $\varepsilon(t)$ is standard Gaussian. Unlike in Sec. 6.4, $\bar{k}_i(t)$ was taken to have a time width spreading over a few stimulus frames. Furthermore, its magnitude was taken to be large enough to give rise to a sharp posterior, satisfying the Laplace approximation and thus $E(\vec{x}|\mathbf{r}, \theta) \approx \vec{x}_{\text{MAP}}(\mathbf{r}, \theta)$. For each value of α , 100 samples of $\varepsilon(t)$ were generated, and the MAP was decoded for each using the corresponding $k_i(t)$ and the fixed spike train. The sample average of those MAPs was taken as the estimate for $\langle \vec{x}_{\text{MAP}} \rangle_{\theta} \approx E(\vec{x}|\mathbf{r})$. Panel (b) shows $\langle \vec{x}_{\text{MAP}} \rangle_{\theta}$ (red trace) for $\alpha = 0$ (top plot) and $\alpha = 1$ (bottom plot). It is seen that the main effect of the finite uncertainty is a shrinkage of the estimate towards zero, i.e., the mean of the prior Gaussian distribution.

this uncertainty by a probability distribution, $p(\theta)$. In the presence of parameter uncertainty, the posterior mean of the stimulus, \vec{x} , is modified to

$$E(\vec{x}|D) = \int E(\vec{x}|D, \theta) p(\theta) d\theta = \int \int \vec{x} p(\vec{x}|D, \theta) p(\theta) d\vec{x} d\theta.$$

Now it is straightforward to sample from $p(\vec{x}|D)$ here; we simply alternate samples θ^i from $p(\theta)$ and then \vec{x}^i from $p(\vec{x}|D, \theta^i)$. (Note that $p(\theta)$ does not depend on \vec{x}^i in this case, so this sampler is not exactly of Gibbs form.) We may again exploit the Rao-Blackwell idea to increase the efficiency of this simple procedure: instead of sampling \vec{x}^i on each step, we could record $E(\vec{x}|D, \theta^i)$ and $Cov(\vec{x}|D, \theta^i)$, as sufficient statistics representing our estimate of $E(\vec{x}|D)$ and our posterior uncertainty around this estimate. Of course, in the GLM setting we can not compute these sufficient statistics exactly, but we can record the Laplace-approximate moments $\hat{\vec{x}}_{\text{MAP}}(\theta^i)$ and $H^{-1}(\theta^i)$, where $\hat{\vec{x}}_{\text{MAP}}(\theta^i)$ and $H(\theta^i)$ denote the usual MAP estimate and Hessian based on the parameter value θ^i . (A similar approximate Rao-Blackwellization is useful in the BARS context discussed in section 7.4, as long as the observed firing rates are high enough that the Laplace approximation is accurate (DiMatteo et al., 2001; Behseta et al., 2005; Wallstrom et al., 2007).)

Roughly speaking, uncertainty in θ will broaden the GLM likelihood (as a function of \vec{x}) and therefore, we expect that as this uncertainty increases the posterior estimate $E(\vec{x}|\mathbf{r})$ will move

towards the prior mean $E(\vec{x})$. In other words, as the Bayesian decoder’s knowledge of the encoding mechanism (represented by the parameters θ) decreases, it discounts the information that the observed spike train, \mathbf{r} , carries about the stimulus and instead relies more strongly on its prior information. This effect is illustrated in Fig. 8; see (Ahmadian et al., 2008) for further analytical details.

8 Computing marginal likelihood is nontrivial but necessary in a number of applications

In many cases we don’t just want to compute posterior means and variances; we also want to compute marginal densities, for example in the change-point detection context, or in computing Bayes factors in the hypothesis testing setting. In particular, we frequently need to compute

$$p(D) = \int p(\vec{x})p(D|\vec{x})d\vec{x}.$$

Another example arises in the estimation of mutual information: we need to compute

$$h(\vec{x}|D) = \int p(\vec{x}|D) \log p(\vec{x}|D) d\vec{x} = -\log p(D) + \int p(\vec{x}|D) \log[p(D|\vec{x})p(\vec{x})] d\vec{x};$$

the term on the right may be computed via standard MCMC on $p(\vec{x}|D)$, since $p(D|\vec{x})$ and $p(\vec{x})$ may in many cases be computed explicitly, but we need some way to handle the $\log p(D)$ term. It is natural to ask if we can use MCMC methods to compute this quantity.

One apparently simple approach here is to use the fact that

$$\begin{aligned} 1 &= \int p(\vec{x})d\vec{x} \\ &= \int p(\vec{x})\frac{p(\vec{x}, D)}{p(\vec{x}, D)}d\vec{x} \\ &= p(D) \int p(\vec{x}|D)\frac{1}{p(D|\vec{x})}d\vec{x}, \end{aligned}$$

i.e.,

$$p(D) = \left(E_{p(\vec{x}|D)} \frac{1}{p(D|\vec{x})} \right)^{-1},$$

which could be computed directly via MCMC sampling from $p(\vec{x}|D)$. Sadly, this “harmonic mean” method is known to be highly unstable⁷: the variance of $1/p(D|\vec{x})$ may be very large (even infinite), and the estimate can be completely swamped by a single sample of a very small $p(D|\vec{x})$.

A more stable approach is described in (Chib and Jelizkov, 2001). The key insight is that we may write $p(D)$ in terms of an average of the acceptance probability of the Metropolis sampler:

$$p(\vec{x}^*|D) = \frac{E_{p(\vec{x}|D)}[\alpha(\vec{x}^*|\vec{x})q(\vec{x}^*|\vec{x})]}{E_{q(\vec{x}|\vec{x}^*)}\alpha(\vec{x}|\vec{x}^*)},$$

where \vec{x}^* is any point in the \vec{x} space (a common choice is to use $\vec{x}^* = \vec{x}_{MAP}$), and $\alpha(\vec{y}|\vec{x})$ denotes the standard Metropolis-Hastings probability of accepting a jump from \vec{x} to \vec{y} . The numerator may be estimated by making use of the M-H samples from $p(\vec{x}|D)$, while the denominator may be

⁷In fact, Radford Neal has dubbed this the “Worst Monte Carlo Method Ever.”

estimated by taking samples from $q(\vec{x}|\vec{x}^*)$, given the fixed starting point \vec{x}^* . Note that in general it is not necessary to use the same number of samples to estimate the numerator and denominator; this gives us some flexibility in ensuring the stability of the quotient.

The equality follows upon noting that, by reversibility of the Metropolis-Hastings Markov chain,

$$q(\vec{x}|\vec{x}^*)\alpha(\vec{x}|\vec{x}^*)p(\vec{x}^*|D) = q(\vec{x}^*|\vec{x})\alpha(\vec{x}^*|\vec{x})p(\vec{x}|D),$$

and then integrating both sides with respect to \vec{x} . Once we have $p(\vec{x}^*|D)$, we may obtain $p(D)$ easily since, by Bayes,

$$p(D) = \frac{p(D|\vec{x}^*)p(\vec{x}^*)}{p(\vec{x}^*|D)}.$$

An alternate approach may be developed if we focus on the problem of computing ratios of marginal likelihoods, instead of the absolute value of a single marginal likelihood. The ability to compute likelihood ratios is sufficient in most applications; for example, to compute the mutual information, we don't care about the conditional entropy so much as the difference between the conditional and marginal entropies, which after a little manipulation may be written in terms of a log of a likelihood ratio. Similarly, when performing maximum marginal likelihood estimation (in the context of empirical Bayes approaches for hierarchical models or, more specifically, the change-point problem we discussed in the last chapter), we only care about relative likelihoods, not their absolute values.

So imagine we have two unnormalized densities,

$$q_i(\vec{x}) = Z_i p_i(\vec{x}), \quad i = 1, 2;$$

here $q_i(\vec{x})$ may be computed exactly, and we would like to compute the ratio

$$\frac{Z_1}{Z_2} = \frac{\int q_1(\vec{x})d\vec{x}}{\int q_2(\vec{x})d\vec{x}}.$$

Now note that, for any well-behaved function $g(\vec{x})$, we have

$$\frac{E_{p_1}[q_2(\vec{x})g(\vec{x})]}{E_{p_2}[q_1(\vec{x})g(\vec{x})]} = \frac{\int \frac{1}{Z_1} q_1(\vec{x}) q_2(\vec{x}) g(\vec{x}) d\vec{x}}{\int \frac{1}{Z_2} q_2(\vec{x}) q_1(\vec{x}) g(\vec{x}) d\vec{x}} = \frac{Z_2 \int q_1(\vec{x}) q_2(\vec{x}) g(\vec{x}) d\vec{x}}{Z_1 \int q_2(\vec{x}) q_1(\vec{x}) g(\vec{x}) d\vec{x}} = \frac{Z_2}{Z_1}.$$

The numerator and denominator on the left can in principle be computed by Monte Carlo sampling from p_1 and p_2 , respectively, so this gives us a way to compute our ratio Z_1/Z_2 . As a special case, if we choose p_1 to be our target distribution $p(\vec{x}|D)$ and q_2 to be a convenient density whose normalization Z_2 we can compute easily (we have in mind the Laplace approximation to $p(\vec{x}|D)$) we can use this method to compute Z_1 directly.

The only problem is that p_1 and p_2 may place most of their mass in very different parts of the \vec{x} space, which will make direct Monte Carlo integration of the numerator and denominator difficult (since most of the samples from p_1 may evaluate to very small q_2 values, and vice versa, leading to high variance in our estimates of these integrals). But remember that $g(\vec{x})$ is arbitrary here, so we can choose this function to optimize the efficiency of our estimates. This idea was first developed by (Bennett, 1976), and was further refined by (Meng and Wong, 1996), who showed that an asymptotically optimal choice of $g(\vec{x})$ here is fairly easy to compute; this optimal $g(\vec{x})$ turns out to amplify the contribution from the region of overlap of the two distributions p_1 and p_2 , thus acting like a bridge connecting the two supports. This method is therefore known as “bridge sampling” in the statistics literature. See (Ahmadian et al., 2008) for applications to estimating the mutual information in a neural decoding setting.

The problem of computing the marginal likelihoods remains a very active research topic in statistics; it is worth emphasizing that both of the approaches discussed here can break down in certain high-dimensional settings, and more involved techniques are necessary. For example, instead of using just a single “bridge” between the two densities p_1 and p_2 , a number of researchers in the physics and statistics literature have advocated using a number of small intermediate steps to bridge the two densities instead. See e.g. (Gelman and Meng, 1998; Neal, 2001).

9 Further reading

MCMC is an enormous field. Other algorithms of interest include “perfect” sampling methods (Casella et al., 2001), which allow exact samples to be drawn (i.e., not just asymptotically exact) from Markov chains with the desired equilibrium distribution $p(\vec{x})$. Another important subfield concerns sampling between models of varying dimensionality, e.g. the “reversible jump” algorithm (Green, 1995; Nguyen et al., 2003). See the books (Liu, 2002; Robert and Casella, 2005) for more details.

References

- Ahmadian, Y., Pillow, J., and Paninski, L. (2008). Efficient Markov Chain Monte Carlo methods for decoding population spike trains. *Under review, Neural Computation*.
- Ahrens, M., Paninski, L., and Sahani, M. (2008). Inferring input nonlinearities in neural encoding models. *Network: Computation in Neural Systems*, 19:35–67.
- Akhmatskaya, E., Bou-Rabee, N., and Reich, S. (2009). A comparison of generalized hybrid monte carlo methods with and without momentum flip. *J. Comput. Phys.*, 228(6):2256–2265.
- Behseta, S., Kass, R., and Wallstrom, G. (2005). Hierarchical models for assessing variability among functions. *Biometrika*, 92:419–434.
- Bennett, C. H. (1976). Efficient estimation of free energy divergences from Monte Carlo data. *Journal of Computational Physics*, 22:245–268.
- Billera, L. and Diaconis, P. (2001). Geometric interpretation of the Metropolis-Hastings algorithm. *Statistical Science*, 16:335–339.
- Casella, G., Lavine, M., and Robert, C. (2001). Explaining the perfect sampler. *The American Statistician*, 55:299–305.
- Chib, S. and Jelizkov, I. (2001). Marginal likelihood from the Metropolis-Hastings output. *Journal of the American Statistical Association*, 96:270–281.
- Davis, R. and Rodriguez-Yam, G. (2005). Estimation for state-space models: an approximate likelihood approach. *Statistica Sinica*, 15:381–406.
- DiMatteo, I., Genovese, C., and Kass, R. (2001). Bayesian curve fitting with free-knot splines. *Biometrika*, 88:1055–1073.
- Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216 – 222.

- Gelman, A., Carlin, J., Stern, H., and Rubin, D. (2003). *Bayesian Data Analysis*. CRC Press.
- Gelman, A. and Meng, X. (1998). Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical Science*, 13:163–185.
- Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.
- Gilks, W. and Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling. *Applied Statistics*, 41:337–348.
- Green, P. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732.
- Ishwaran, H. (1999). Applications of hybrid Monte Carlo to Bayesian generalized linear models: quasicomplete separation and neural networks. *Journal of Computational and Graphical Statistics*, 8:779–799.
- Izaguirre, J. A. and Hampton, S. S. (2004). Shadow hybrid Monte Carlo: an efficient propagator in phase space of macromolecules. *J. Comput. Phys.*, 200(2):581–604.
- Jungbacker, B. and Koopman, S. (2007). Monte Carlo estimation for nonlinear non-Gaussian state space models. *Biometrika*, 94:827–839.
- Kass, R., Ventura, V., and Cai, C. (2003). Statistical smoothing of neuronal data. *Network: Computation in Neural Systems*, 14:5–15.
- Kennedy, A. D. and Pendleton, B. (1991). Acceptances and autocorrelations in hybrid Monte Carlo. *Nuclear Physics B - Proceedings Supplements*, 20:118 – 121.
- Liu, J. (2002). *Monte Carlo Strategies in Scientific Computing*. Springer.
- Lovasz, L. and Vempala, S. (2003). The geometry of logconcave functions and an $O^*(n^3)$ sampling algorithm. Technical Report 2003-04, Microsoft Research.
- Meng, X. L. and Wong, W. H. (1996). Simulating ratios of normalizing constants via a simple identity: A theoretical exploration. *Statistica Sinica*, 6:831–860.
- Nadarajah, S. and Kotz, S. (2008). Estimation methods for the multivariate t distribution. *Acta Applicandae Mathematicae*, 102:99–118.
- Neal, R. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, Canada.
- Neal, R. (1996). *Bayesian Learning for Neural Networks*. Number 118 in Lecture Notes in Statistics. Springer.
- Neal, R. (2003). Slice sampling. *Annals of Statistics*, 31:705–767.
- Neal, R. (2004). Improving asymptotic variance of MCMC estimators: Non-reversible chains are better. *University of Toronto Statistics Tech Report*, 0406.
- Neal, R. M. (2001). Annealed importance sampling. *Statistics and Computing*, 11:125–139.

- Nguyen, D., Frank, L., and Brown, E. (2003). An application of reversible-jump Markov chain Monte Carlo to spike classification of multi-unit extracellular recordings. *Network*, 14:61–82.
- Pillow, J. and Latham, P. (2007). Neural characterization in partially observed populations of spiking neurons. *NIPS*.
- Pillow, J., Shlens, J., Paninski, L., Sher, A., Litke, A., Chichilnisky, E., and Simoncelli, E. (2008). Spatiotemporal correlations and visual signaling in a complete neuronal population. *Nature*, 454:995–999.
- Portilla, J., Strela, V., Wainwright, M. J., and Simoncelli, E. P. (2003). Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans Image Processing*, 12(11):1338–1351.
- Rigat, F., de Gunst, M., and van Pelt, J. (2006). Bayesian modelling and analysis of spatio-temporal neuronal networks. *Bayesian Analysis*, 1:733–764.
- Robert, C. and Casella, G. (2005). *Monte Carlo Statistical Methods*. Springer.
- Roberts, G. and Rosenthal, J. (2001). Optimal scaling for various metropolis-hastings algorithms. *Statistical Science*, 16:351–367.
- Salakhutdinov, R. and Mnih, A. (2008). Bayesian probabilistic matrix factorization using MCMC. *ICML*.
- Schervish, M. (1995). *Theory of statistics*. Springer-Verlag, New York.
- Shephard, N. and Pitt, M. K. (1997). Likelihood analysis of non-Gaussian measurement time series. *Biometrika*, 84(3):653–667.
- Wallstrom, G., Liebner, J., and Kass, R. E. (2007). An implementation of Bayesian adaptive regression splines (BARS) in C with S and R wrappers. *Journal of Statistical Software*, 26:1–21.