Notes on tridiagonal methods in state-space models

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

February 11, 2008

A number of important models in neuroscience may be described in "state-space" form with point-process observations: a hidden state variable evolves according to some continuous Markovian dynamics, and the rate of the observed spike trains is some function of this underlying hidden state. Examples include the integrate-and-fire model (Paninski et al., 2008) and models used in a number of spike train decoding applications (Brown et al., 1998; Truccolo et al., 2005).

It is of interest to compute the most likely (ML) or maximum a posteriori (MAP) path that the hidden state variable traversed on a given trial, given the observed spike trains. The point-process filter algorithm introduced in (Brown et al., 1998) computes an approximation to this ML path, but this approximation is rigorously accurate only in the so-called "high-information" limit (when we have a large number of spikes per unit time, or when the hidden state dynamics are nearly deterministic). Moreover, this algorithm relies on a Gaussian approximation which may be highly inaccurate in some cases, for example when nonnegativity constraints come into play.

A more direct approach is well-known in the state space literature (Davis and Rodriguez-Yam, 2005; Jungbacker and Koopman, 2007). Assume the hidden process q_t satisies the usual recursive model,

$$q_{t+1} = Aq_t + \eta_t,$$

and that the density $p(\eta_t)$ of the i.i.d. innovation noise η_t is log-concave. In addition, assume that the initial density $p(q_0)$ is log-concave and the observation density $p(y_t|q_t)$ is also log-concave in q_t. Then it is easy to see that the log-posterior

$$\log p(Q|Y) = c + \log p(q_0) + \sum_t \log p(y_t|q_t) \sum_t \log p(q_{t+1}|q_t)$$

is log-concave in Q; i.e., computing the MAP path is a concave problem. Further, if $\log p(q_0), \log p(y_t|q_t)$, and $\log p(q_{t+1}|q_t)$ are all smooth functions of Q, then we may apply standard approaches such as Newton's algorithm to solve this optimization problem.

Now the key idea is to exploit the special structure of this problem, by noting that the Hessian matrix $J = \nabla\nabla_Q \log p(Q|Y)$ is a block tridiagonal matrix, with blocks of size $d = \dim(q_t)$, since $\log p(Q|Y)$ does not have any terms of the form $f(q_s, q_{s'})$ for $|s - s'| > 1$. Thus the Newton step

$$\hat{Q}^{(i+1)} = \hat{Q}^i - J^{-1}\nabla_Q \log p(Q|Y)\Big|_{Q=\hat{Q}^i}$$

may be computed in $O(d^3T)$ time (e.g., via Gaussian elimination, which in the tridiagonal setting is also known as the Thomas algorithm) instead of the usual $O((dT)^3)$ required to solve a problem of size $\dim(Q) = dT$. (Note that we do not want to compute the inverse matrix J^{-1} here, but rather just to solve the linear system $J^{-1}\nabla$. This may be done efficiently in Matlab via the command $J\backslash\nabla$; note that J must be represented as a sparse matrix to exploit the $O(T)$ algorithm.) Thus we may compute the MAP path exactly using this direct method, in time comparable to that required by the approximate point-process smoothing algorithm.

It turns out that this tridiagonal Newton-Raphson method also has a helpful interpretation in terms of an iterative Kalman filter. In each Newton iteration, we make a second-order approximation of the terms $\log p(q_{t+1}|q_t)$ and $\log p(y_t|q_t)$ about the current estimate $\hat{Q}^{(i)}$. This second-order approximation corresponds exactly to a linear Gaussian approximation of the transition and observation densities. Thus each Newton iteration corresponds to a forward-backward sweep of an inhomogeneous linear-Gaussian Kalman filter, where the coefficients of the Kalman model are updated once per Newton iteration as we update the Hessian and gradient of $\log p(Q|Y)$ at $\hat{Q}^{(i)}$. Of course, in the case of linear Gaussian observations and transitions, the algorithm terminates after one step, since the Hessian and gradient remain constant with each iteration, and we are left with the standard Kalman filter. See (Davis and Rodriguez-Yam, 2005; Jungbacker and Koopman, 2007) and references therein for further discussion of the connections with the Kalman theory.

Once we have obtained $\hat{Q}_{MAP}$, it is straightforward to develop any of the Laplace-approximation applications discussed in (Pillow and Paninski, 2007). For example, approximating the marginal likelihood of the observed data Y via Laplace's method requires that we compute a (block-tridiagonal) quadratic function of Q (which may be done in $O(T)$ time) and a determinant of the block-tridiagonal Hessian matrix J (this can also be done in $O(T)$ time, via a standard recursion). The E step of the EM algorithm in this model requires that we compute the first and second moments $E(q_t|Y), E(q_t q_{t+1}^T|Y)$, etc.; computing the Laplace approximation of these quantities again requires a single run of a linear-Gaussian Kalman filter once $\hat{Q}_{MAP}$ has been obtained. Finally, the MAP path can be used to provide a good initialization for iterative algorithms (Monte Carlo or expectation propagation) for computing conditional expectations in these models. It can be shown that each EP iteration, for example, requires just a single Kalman sweep (Ypma and Heskes, 2003). See (Koyama et al., 2008) for details. We will discuss Monte Carlo approaches in a bit more depth in the next section.

Example: applications to spike-train decoding

As a first application of these ideas, let's take another look at the problem of decoding a stimulus $\vec{x}$ given spike train responses, as discussed in (Pillow and Paninski, 2007). Recall our model for the spike train response,

$$\lambda_i(t) = f(\vec{k}_i^T \vec{x}_t + h_i(t)),$$

with $h_i(t)$ denoting the summed spike-history effects in neuron i at time t; these functions $h_i(t)$ may be considered fixed in this case, since we have observed the spike train Y.

To apply the tridiagonal methods discussed above, first we must cast our model in state-

space form. For simplicity, assume that the stimulus is a scalar function of time, i.e.,

$$\vec{k}_i^T \vec{x}_t = x * k(t) = \int_{-\infty}^{t} x(s) k_i(t-s) ds;$$

the generalization to vector stimuli $x(t)$ is straightforward. Now further assume that each filter $k_i(.)$ may be represented as a weighted sum of exponentials:

$$k_i(t) = \sum_{j=1}^{d} a_{ij} e^{-t/\tau_j} 1(t \geq 0),$$

for some choice of time constants $\{\tau_1, \tau_2, \ldots, \tau_d\}$. (This is not a major assumption, since we may always add another function e^{-t/τ_j} to our basis set to improve the approximation if necessary.) Now if we define the convolved variables

$$q_j(t) = \int_{-\infty}^{t} x(s) e^{(t-s)/\tau_j} ds,$$

then clearly

$$\lambda_i(t) = f(h_i(t) + \sum_j a_{ij} q_j(t));$$

now if we may model $x(t)$ as an autoregressive process — e.g., in the simplest case, in a white-noise paradigm, $x(t)$ will be i.i.d. Gaussian — then $\vec{q}_t$ evolves according to an autoregressive process (with a low-rank innovations covariance matrix) and we have successfully cast our model in the standard point-process filter setting. Thus the MAP estimate of the stimulus $x(t)$ may be computed in $O(T)$ time using the tridiagonal method described above.

(Ahmadian et al., 2008) discuss Markov chain Monte Carlo methods for decoding given spike train data in this setting. A key step in the hit-and-run and Metropolis-adjusted Langevin algorithm (Robert and Casella, 2005; Lovasz and Vempala, 2003) algorithms discussed there is to draw Gaussian samples from the Laplace approximation density. An $O(T)$ algorithm for drawing these samples is given in (Jungbacker and Koopman, 2007).

Applications to constrained problems via the barrier method

In the above we have assumed that the MAP path may be computed via an unconstrained smooth optimization. In many examples of interest we have to deal with constrained optimization problems instead. In particular, nonnegativity constraints arise frequently on physical grounds. To handle these constrained problems while exploiting the fast tridiagonal techniques discussed above, we can employ "barrier" methods (Boyd and Vandenberghe, 2004). The idea is to replace the constrained problem

$$\hat{Q}_{MAP} = \arg \max_{Q: q_t \geq 0} \log p(Q|Y)$$

with a sequence of unconstrained problems

$$\hat{Q}_\epsilon = \arg \max_Q \log p(Q|Y) + \epsilon \sum_t \log q_t;$$

clearly, $\hat{Q}_\epsilon$ satisfies the nonnegativity constraint, since $\log u \to -\infty$ as $u \to 0$. (We have specialized to the nonnegative case for concreteness, but the idea may be generalized easily to any convex constraint set; see (Boyd and Vandenberghe, 2004) for details.) Furthermore, it is easy to show that if $\hat{Q}_{MAP}$ is unique, then $\hat{Q}_\epsilon$ converges to $\hat{Q}_{MAP}$ as $\epsilon \to 0$. Finally, the Hessian of the objective function $\log p(Q|Y) + \epsilon \sum_t \log q_t$ retains the tridiagonal properties of the original objective $\log p(Q|Y)$; thus we may use our fast Newton iteration to obtain $\hat{Q}_\epsilon$, for any ϵ.

We give three applications of this barrier approach below.

Integrate-and-fire model with hard threshold

As discussed above, the integrate-and-fire model may be written in state space form (Paninski, 2006; Koyama et al., 2008; Paninski et al., 2008). Computing the most likely voltage path in the hard-threshold IF model given an observed sequence of spikes via the barrier method is straightforward: we simply need to maximize the tridiagonal quadratic loglikelihood

$$-\frac{1}{2\sigma^2 dt} \sum_t \left[V(t+dt) - (V(t) - gV(t) + I(t))\right]^2$$

under the linear constraint $V(t) \leq V_{th}$ for all times t. The Hessian of the barrier term enforcing this subthreshold constraint is diagonal, and therefore the Newton step may be computed in $O(T)$ time. This method may also be easily applied to the common-noise model discussed in (Iyengar, 1985; Paninski, 2006; de la Rocha et al., 2007), in which a common noise source drives a network of IF cells, and to the multidimensional linear IF neuron considered by (Badel et al., 2005) and others; in each of these cases, the loglikelihood may be written as a block tridiagonal quadratic form (where in the one-dimensional case described above, the block size was just one).

This $O(T)$ method for computing the MAP voltage path may be used to develop a fast method for fitting the parameters in this model. Assume the input current $I(t)$ may be written in linear form:

$$I(t) = \sum_i k_i I_i(t).$$

Computing the likelihood $p(Y|\vec{k})$ in this model remains challenging (Nikitchenko and Paninski, 2007); our standard Gaussian approximations for $P(V|Y)$ here are rather inaccurate, due to the sharp "corners" induced by the thresholding $p(y_t|V_t)$. However, in the low-noise limit $\sigma \to 0$ it is possible to show that the quadratic term in the Laplace approximation dominates the likelihood:

$$\log p(Y|\vec{k}) = -\frac{1 + o(1)}{2\sigma^2 dt} \sum_t \left[V_{MAP}(t+dt) - (V_{MAP}(t) - gV_{MAP}(t) + I(t))\right]^2, \ \sigma \to 0$$

(i.e., we may drop the determinant in the approximate Gaussian term). Note that this is a positive semidefinite quadratic form, jointly in V_{MAP} and $\vec{k}$. Thus to compute $\hat{k}_{MLE}$ in this low-noise limit we need to solve

$$\hat{k}_{MLE} = \arg\max_{\vec{k}} \max_{V: V(t) \leq V_{th}} -\frac{1}{2\sigma^2 dt} \sum_t \left[V(t+dt) - (V(t) - gV(t) + I(t))\right]^2.$$

(This may be considered a "profile likelihood" approach to inferring the parameter $\vec{k}$ (Murphy and van der Vaart, 2000).) If we order the parameter vector as $\{\vec{k}, V\}$, this quadratic form may be written in block form $J = \begin{pmatrix} J_{\vec{k}\vec{k}} & J_{\vec{k}V}^T \\ J_{\vec{k}V} & J_{VV} \end{pmatrix}$, where J_{VV} is tridiagonal. We cannot directly apply our $O(T)$ methods to obtain $\hat{k}_{MLE}$, since J itself is not tridiagonal. However, if we note that J may be written as a sum of a tridiagonal matrix and a matrix of low rank (at most $2\dim(\vec{k})$), then applying the Woodbury lemma leads to an $O(T)$ method for computing the Newton step for simultaneously obtaining $\{\hat{k}_{MLE}, \hat{V}_{MAP}\}$. (This Newton approach is, as usual, much faster than the coordinate descent method of iteratively optimizing V given $\vec{k}$ and vice versa, which leads to slow, jagged solution paths.)

Fast nonnegative deconvolution methods for inferring spike times from noisy, intermittent calcium traces

Calcium-imaging methods have become popular recently for investigating the behavior of populations of neurons (Cossart et al., 2003; Kerr et al., 2005; Ohki et al., 2006). However, calcium dynamics evolve over a much slower timescale than do spike trains; hence deconvolution techniques are necessary to recover the spike train from the available noisy, intermittent calcium observations.

A reasonable first-pass model of calcium dynamics is as follows. Let y_t denote the observed fluorescence signal, and q_t the underlying, unobserved calcium signal. (Note that q_t might be measured on a finer time scale than y_t.) For the dynamics, we use a first-order model,

$$q_{t+dt} = aq_t + bn_t,$$

for some positive constants a, b; here n_t denotes the spike train which is driving these calcium dynamics, so the calcium jumps up with each spike and then decays exponentially according to the constant a. Finally, assume linear Gaussian observations

$$y_t = q_t + e_t;\ e_t \sim \mathcal{N}(0, \sigma^2).$$

Finally, we assume a log-concave, nonnegative prior on n_t. See (Vogelstein and Paninski, 2007) for a much more detailed discussion of the assumptions involved with this simple model.

Now maximizing $p(Q|Y)$ is nearly routine. The likelihood term $\log p(Y|Q)$ contributes a diagonal term to the Hessian, while the dynamics term $p(q_{t+dt}|q_t)$ (including a barrier term to enforce the nonnegativity of n_t) contributes a tridiagonal term. Each Newton iteration may once again be done in $O(T)$ time.

Inferring presynaptic inputs given postsynaptic voltage recordings

Given voltage recordings at a postsynaptic site, can we recover the time course of the presynaptic conductance inputs (Huys et al., 2006; Paninski, 2007)? We may write down a simple state-space model:

$$V(t+dt) = V(t) + dt\,[g_L(V_L - V(t)) + g_E(t)(V_E - V(t)) + g_I(t)(V_I - V(t))] + \epsilon(t)$$

$$g_E(t+dt) = g_E(t) - \frac{g_E(t)}{\tau_E} + N_E(t)$$

$$g_I(t+dt) = g_I(t) - \frac{g_I(t)}{\tau_I} + N_I(t).$$

Here $g_E(t)$ denotes the excitatory presynaptic conductance at time t, and $g_I(t)$ the inhibitory conductance; V_L, V_E, and V_I denote the leak, excitatory, and inhibitory reversal potentials, respectively. Finally, $\epsilon(t)$ denotes an unobserved i.i.d. current noise with a log-concave density, and $N_E(t)$ and $N_I(t)$ denote the presynaptic excitatory and inhibitory inputs (which must be nonnegative on physical grounds); we assume these inputs also have a log-concave density.

Assume $V(t)$ is observed noiselessly for simplicity. Then let our observed variable $y_t = V(t+dt) - V(t)$ and our state variable $q_t = (g_E(t)\ g_I(t))^T$. Now applying our barrier $O(T)$ method is straightforward; the Hessian in this case is block-tridiagonal, with blocks of size two (since our state variable q_t is two-dimensional). If $V(t)$ is observed with noise, on the other hand, then we may no longer write the system (V, g_E, g_I) in terms of a linear state-space model, and different techniques (e.g., particle filtering) are required (Paninski, 2007).

Applications to spatial models

The applications discussed above all involve state-space models which evolve through time. However, these tridiagonal ideas are also quite useful in the context of spatial models. We give a few examples here.

Estimating two-dimensional firing rate maps

(Rahnama Rad and Paninski, 2008)

Imagine we would like to estimate some two-dimensional rate function $z(\vec{x}) = z(x, y)$ from point process observations. We will consider several somewhat distinct cases:

1. We observe a spatial point process whose rate is given by $\lambda(\vec{x}) = f(z(\vec{x}) + \sum_i a_i g_i(\vec{x}))$, where $g_i(\vec{x})$ is some set of known functions and a_i is a set of scalar weights. (Similar models have beend considered in forestry or astronomy applications, for example (Moeller and Waagepetersen, 2004).)

2. We observe a temporal process whose rate is given by $\lambda(t) = f\left(z(\vec{x}(t)) + \vec{k}^T\vec{y}(t)\right)$, where $\vec{x}(t)$ is some known time-varying path through space (e.g., the time-varying position of a rat in a maze (Brown et al., 1998) or the hand position in a motor experiment (Paninski et al., 2004)), $\vec{y}(t)$ is a vector of fully-observed covariates (possibly including spike history terms (Paninski, 2004; Truccolo et al., 2005)) and $\vec{k}$ is a vector of weights.

3. We make repeated observations of a temporal point process whose mean rate function may change somewhat from trial to trial[1]; in this case we may model the rate as $\lambda(t, i)$, where t denotes the time within a trial and i denotes the trial number (Czanner et al., 2008).

4. We observe a temporal process whose rate is given by $\lambda(t) = f\left(z(x(t)) + \vec{k}^T\vec{y}(t)\right)$, where $x(t)$ is some known time-varying path through a one-dimensional space (e.g., the time-varying position of a rat in a linear maze), and $z(.)$ may change somewhat as a function of time (Frank et al., 2002).

[1] Thanks to C. Shalizi for pointing out this example.

In each case, the function $f(.)$ is assumed to be convex and log-concave (Paninski, 2004). Now we further assume that z is generated by sampling from a Gaussian process with mean zero and covariance function $C(\vec{x}, \vec{x}')$. (The mean-zero assumption entails no loss of generality, since we can simply include a mean offset change in the collection of functions $g_i(\vec{x})$ or $y(t)$.) In this setting, the resulting (marginal) point process is doubly-stochastic and is known as a Cox process (Snyder and Miller, 1991; Moeller and Waagepetersen, 2004), and in the special case that $f(.) = \exp(.)$, the process is called a log-Gaussian Cox process (Moeller et al., 1998).

In many applications, we would like to compute the conditional mean of z given observed spike train data D. It is well-known that under the above conditions on $f(.)$, the posterior

$$p(z|D) \propto p(D|z)p(z)$$

is log-concave as a function of z (Paninski, 2004; Paninski, 2005), since both the prior $p(z)$ and the point-process likelihood $p(D|z)$ are log-concave in z, and log-concavity is preserved under multiplication. We will exploit this log-concavity in this paper to develop efficient approximation and sampling algorithms for the posterior $p(z|D)$.

Our basic approximation is a standard Laplace approximation (Kass and Raftery, 1995; Paninski et al., 2007) for the posterior:

$$p(z|D) \approx \frac{1}{(2\pi)^{d/2}|C_D|^{1/2}} \exp\left(-\frac{1}{2}(z-\hat{z}_D)^T C_D^{-1}(z-\hat{z}_D)\right),$$

where $d = \dim(z)$,

$$\hat{z}_D = \arg\max_z p(z|D)$$

and

$$C_D^{-1} = C^{-1} + H_D,$$

with

$$H_D = -\nabla\nabla_z \log p(z|D)_{z=\hat{z}_D}.$$

Note that C_D^{-1} is quite easy to compute once we have $\hat{z}_D$, since H_D is diagonal. The key, then, is to develop efficient methods for computing $\hat{z}_D$. Log-concavity of $p(\vec{x}|D)$ ensures that $\arg\max_z p(z|D)$ is unique and may be found by simple ascent methods. The standard Newton-Raphson ascent method requires that we solve

$$\left(C^{-1} - \nabla\nabla_z \log p(z|D)_{z=\hat{z}^{(i)}}\right) x = \nabla_z \log p(z|D)_{z=\hat{z}^{(i)}} \tag{1}$$

for x, where $\hat{z}^{(i)}$ denotes our estimate of z after i iterations of Newton-Raphson. For example, in the simplest setting (case 1 above, with all the coefficients a_i set to zero), we have the standard point-process likelihood (Snyder and Miller, 1991)

$$\log p(D|z) = \sum_j \log f[z(\vec{x}_j)] - \int f[z(\vec{x})]dxdy + const.,$$

where j indexes the location $\vec{x}_j$ where the j-th spike was observed, and so

$$[\nabla_z \log p(z|D)]_{\vec{x}} = \sum_j \frac{f'}{f}[z(\vec{x})]\delta(\vec{x}-\vec{x}_j) - f'[z(\vec{x})]dxdy$$

and

$$[\nabla\nabla_z \log p(z|D)]_{\vec{x},\vec{x}} = \sum_j \frac{f''f - (f')^2}{f^2}[z(\vec{x})]\delta(\vec{x} - \vec{x}_j) - f''[z(\vec{x})]dxdy,$$

where $f'(.)$ and $f''(.)$ denote the first and second (scalar) derivatives of the function $f(.)$. Thus we see that if $f(.) \neq \exp(.)$, the Hessian may have some non-smooth terms (from the $\delta(\vec{x} - \vec{x}_j)$ terms), though these terms are always matched with non-smooth behavior in the corresponding elements of the gradient.

So clearly the feasibility of this smoothing method will rest on the tractability of the Newton step (1), which in turn rests on our ability to solve equations of the form

$$\left(C^{-1} + H\right) x = b,$$

for diagonal matrices H. For general matrices C^{-1}, this will require $O(d^3)$ time, which is intractable for reasonably-sized z. Thus we need to choose C^{-1} more carefully, so that we can exploit fast solvers. If $[C^{-1}]_{\vec{x},\vec{y}} = 0$ whenever $\vec{x}$ and $\vec{y}$ are not nearest neighbors on the discrete two-dimensional grid, then C^{-1} may be written in block tridiagonal form (where each block is of size $\sqrt{d} \times \sqrt{d}$), and our equation resembles a discrete Poisson equation, for which efficient multigrid solvers are available which require just $O(d)$ time (Press et al., 1992). Even standard Gaussian elimination methods for solving the equation are quite efficient here, requiring just $O(d^{3/2})$ time.

It is worth mentioning a few nice properties of the estimator $f(\hat{z}_D)$ for the rate. First, the Bayesian approach has all the usual advantages: our estimator comes equipped with errorbars (as we will discuss at more length below), and it is straightforward to incorporate our prior knowledge about the smoothness of z in the definition of the covariance function C (as well as the covariate information in $g_i(\vec{x})$ or $y(t)$). Moreover, the estimator $f(\hat{z}_D)$ behaves as an adaptive smoother: regions of low firing rate are smoothed more than regions of high firing rate, where in a sense more data is available (and therefore less smoothing is desirable): speaking colloquially, the estimator "listens to the data" more closely in regions of high firing rate, where the Fisher information (Hessian) is larger.

It is also worth noting that the generalization to non-Gaussian priors of the form

$$p(z) \propto \exp\left(\sum_{ij} h_{ij}[z(\vec{x}_i) - z(\vec{x}_j)]\right),$$

for some collection of smooth, symmetric, concave functions $h_{ij}(.)$, is straightforward, since the log-posterior remains smooth and concave. This is useful because by using sub-quadratic penalty functions $h_{ij}(.)$ we can capture sharper edge effects than in the Gaussian prior (Gao et al., 2002). Once again, to maintain sparseness of the Hessian we choose $h_{ij}(.)$ to be uniformly zero for non-neighboring $(\vec{x}_i, \vec{x}_j)$. (We recover the Gaussian case if we choose $h_{ij}(.)$ to be quadratic with negative curvature; in this case, the nonzero elements of the inverse prior covariance matrix $(C^{-1})_{ij}$ correspond to the pairs (i, j) for which the functions $h_{ij}(.)$ are not uniformly zero.)

As in the temporal case, we can use the hit-and-run or Metropolis-adjusted Langevin algorithm (Robert and Casella, 2005; Lovasz and Vempala, 2003) to efficiently sample from $p(z|D)$; all we need is an efficient algorithm for solving

$$Rz = \eta,$$

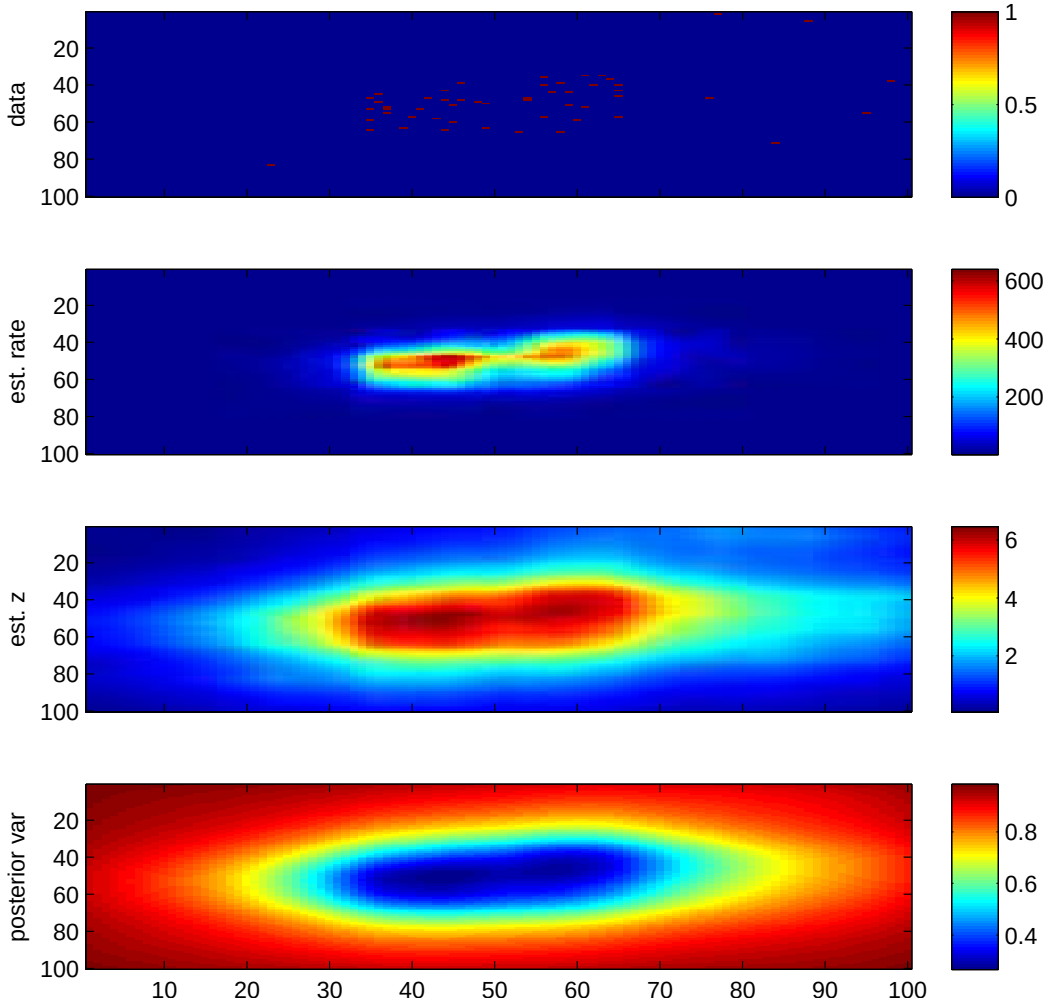

Figure 1: Example of the smoother applied to simulated spatial point-process data. The bottom panel shows the posterior marginal variance of z given the observed data. The dimensionality $d = 10^4$ here; the solution was obtained in ~ 10 seconds on a laptop computer.

where η is a standard normal sample and R is the Cholesky decomposition of C_D^{-1}. R may be computed in $O(d^2)$ time (and only requires $O(d^{3/2})$ space), and $Rz = \eta$ again requires just $O(d^{3/2})$ time. Alternatively, the Kalman approach of (Jungbacker and Koopman, 2007) may be employed.

Note that we have not specified how to choose C; in the case that C^{-1} is non-zero only for nearest neighbors, we must specify four parameters to uniquely specify C, and in the case that the covariance in the x direction and in the y direction are the same (i.e., C is invariant with respect to rotations of the $\vec{x}$-space of size $\pi/2$), we must specify three parameters. To optimize these parameters of C, we can derive an expectation-maximization (EM) algorithm (Dempster et al., 1977; Smith and Brown, 2003); this requires that we compute the diagonal and off-diagonal of C_D. These terms may be computed without ever computing the full matrix C_D (which would require $O(d^2)$ storage) by making use of the block-tridiagonal structure of

C_D^{-1}, via the Asif-Moura or Kalman smoother algorithm. This still requires $O(d^2)$ time, but the parameters of C could be optimized on a coarser scale (making d smaller) and then the smoothing could be performed with fixed d at the desired fine scale; thus this $O(d^2)$ time scaling is not a major limitation.

Reconstructing color images given cone responses

As a final example, we discuss an interesting application to color vision described in (Brainard et al., 2008): how can we reconstruct a color image given just a few observed cone responses?

Let N be the number of pixels of the image we want to reconstruct; 3 is the number of color elements.

The prior on the image $\vec{x}$ is truncated Gaussian:

$$p(\vec{x}) = (1/Z)\mathcal{N}_{\mu,C}(x)1(\vec{x} \geq 0),$$

so the log-prior is

$$\log p(\vec{x}) = c - \frac{1}{2}(\vec{x} - \mu)^T C^{-1}(\vec{x} - \mu)$$

on the nonnegative orthant $\vec{x} \geq 0$, where c is a constant we can ignore.

The covariance matrix C has a special form that we can exploit quite fruitfully here. We model the covariance as separable in the space and color domain, and also separable in its horizontal and vertical components, i.e.,

$$C = C_x \otimes C_y \otimes C_c,$$

where $\otimes$ denotes the Kronecker product. Thus, by the standard properties of the Kronecker product,

$$C^{-1} = C_x^{-1} \otimes C_y^{-1} \otimes C_c^{-1}.$$

Furthermore, C_x and C_y have a special symmetric Toeplitz Markov form:

$$C_x = C_y = \sigma^2 \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 & \dots \\ \rho & 1 & \rho & \rho^2 & \dots \\ \rho^2 & \rho & 1 & \rho & \dots \\ & \vdots & & & \ddots \end{bmatrix},$$

with $\rho = e^{-dx/\tau}$, where τ denotes the spatial correlation width and dx is the discretization width. (Note that the corresponding spatial power spectrum has the correct $1/f^2$ behavior. The horizontal and vertical correlation scales are assumed to be the same here; this may easily be generalized, of course.) This is the stationary covariance of the Ornstein-Uhlenbeck process, and has a well-known inverse:

$$C_x^{-1} = \frac{1}{\sigma^2(1-\rho^2)} \begin{bmatrix} 1 & -\rho & 0 & 0 & \dots \\ -\rho & 1+\rho^2 & -\rho & 0 & \dots \\ 0 & -\rho & 1+\rho^2 & -\rho & \dots \\ & \vdots & & & \ddots \\ \dots & 0 & -\rho & 1+\rho^2 & -\rho \\ \dots & 0 & 0 & -\rho & 1 \end{bmatrix}.$$

Note in particular that this matrix is tridiagonal. Finally, the color covariance C_c is small — just 3×3 — so this is easy enough to invert directly.

Observation model

Images are then mapped linearly through some anisotroptic blurring lens; this transformation is positive and linear, corresponding to some nonnegative matrix K.

The number of isomerization events on the photoreceptor array given an image $\vec{x}$ can be modeled as independent Gaussian,

$$p(response = \{n_i\}|\vec{x}) = \mathcal{N}_{K\vec{x},\sigma^2_{resp}I}(\vec{n}).$$

A more accurate model might be that the cone responses are independent Poisson,

$$p(response = \{n_i\}|\vec{x}) = \prod_i (e^{-\lambda_i dt}(\lambda_i dt)^n_i)/n_i!$$

with the rates λ_i defined as

$$\lambda_i = b + (K\vec{x})_i;$$

b is the baseline dark-adapted isomerization rate. Remember, even in the case that $b = 0$, λ_i is guaranteed to be nonnegative, by the positivity of K and the nonnegativity of $\vec{x}$.

MAP decoding

Now we need to decode an observed array $\{n_i\}$ of isomerizations into an image $\vec{x}$. One simple way to do this is to compute the maximum *a posteriori* (MAP) estimator,

$$\vec{x}_{MAP} \equiv \arg\max_{\vec{x}:\vec{x}\geq 0} p(\vec{x}|\{n_i\}) = \arg\max_{\vec{x}:\vec{x}\geq 0} \log p(\{n_i\}|\vec{x}) + \log p(\vec{x})$$

Here $\log p(\vec{x})$ is a (negative definite) quadratic function of $\vec{x}$, as discussed above. The likelihood term, on the other hand, is

$$\log p(\{n_i\}|\vec{x}) = c - \frac{1}{2\sigma^2_{resp}} \sum_i (n_i - (K\vec{x})_i)^2$$

in the Gaussian case and

$$\log p(\{n_i\}|\vec{x}) = c + \sum_i (n_i \log(b + (K\vec{x})_i) - (b + (K\vec{x})_i)\, dt)$$

in the Poisson case.

Since the space $\{\vec{x} : \vec{x} \geq 0\}$ is convex and the function $\log p(\vec{x}|\{n_i\})$ is strictly concave, this optimization problem may be solved via standard ascent techniques: there is a single global maximum, no matter what data $\{n_i\}$ are observed.

We can solve this constrained problem via the barrier method: to make the notation explicit, we are replacing our constrained problem,

$$\vec{x}_{MAP} = \arg\max_{\vec{x}:\vec{x}\geq 0} \log p(\vec{x}|\{n_i\}),$$

with a series of simpler unconstrained problems,

$$\vec{x}_\epsilon = \arg\max_{\vec{x}} \log p(\vec{x}|\{n_i\}) + \epsilon \sum_j \log \vec{x}_j,$$

where j indexes the elements of the $3N$-dimensional vector $\vec{x}$. Note that computing this objective function requires just $O(N)$ time and memory, due to the sparse form of C^{-1}.

We use Newton's method to solve for $\vec{x}_\epsilon$. Computing the gradient and Hessian of our objective function is straightforward. For the Gaussian model,

$$\begin{aligned}\nabla_{\vec{x}} &\left(\log p(\vec{x}) + \log p(\{n_i\}|\vec{x}) + \epsilon \sum_j \log \vec{x}_j \right) \\ &= \nabla_{\vec{x}} \left(-\frac{1}{2}(\vec{x}-\mu)^T C^{-1}(\vec{x}-\mu) - \frac{1}{2\sigma_{resp}^2} \sum_i (n_i - (K\vec{x})_i)^2 + \epsilon \sum_j \log \vec{x}_j \right) \\ &= -C^{-1}(\vec{x}-\mu) - (K^T K\vec{x} - K^T\vec{n}) + \epsilon \vec{x}.^{-1},\end{aligned}$$

where $\vec{x}.^{-1}$ denotes the vector $\vec{x}$ raised pointwise to the -1 power. The Hessian (second-derivative matrix) of this objective function may be computed as

$$H = -C^{-1} - K^T K - \epsilon \text{ diag}\left[\vec{x}.^{-2}\right].$$

In the Poisson case, we have

$$\begin{aligned}\nabla_{\vec{x}} &\left(-\frac{1}{2}(\vec{x}-\mu)^T C^{-1}(\vec{x}-\mu) + \sum_i (n_i \log(b + (K\vec{x})_i) - (b + (K\vec{x})_i)dt) + \epsilon \sum_j \log \vec{x}_j \right) \\ &= -C^{-1}(\vec{x}-\mu) + K^T \text{diag}\left[(b+K\vec{x}).^{-1}\right]\vec{n} + K^T 1 + \epsilon \vec{x}.^{-1},\end{aligned}$$

and

$$H = -C^{-1} - K^T \text{diag}\left[\vec{n}./(b+K\vec{x}).^{-2}\right] K - \epsilon \text{ diag}\left[\vec{x}.^{-2}\right].$$

The Hessian is clearly negative definite, as expected, given the concavity of the objective function. More importantly, this matrix has a very useful form: it is block tridiagonal (recall the form of C^{-1} discussed above) plus a low-rank matrix[2], and therefore only requires $O(N)$ storage (we never store the full $3N \times 3N$ matrix explicitly). Moreover, computing the Newton direction $-H^{-1}\nabla$ may be done in just $O(N^{3/2})$ time, by making use of the block tridiagonal structure and the Woodbury lemma: denoting $A = C^{-1} + \epsilon \text{ diag}\left[\vec{x}.^{-2}\right]$, we need to compute

$$(A + K^T D K)^{-1}\nabla = A^{-1}\nabla - A^{-1}K_1^T(D_1^{-1} + K_1 A^{-1} K_1^T)^{-1} K_1 A^{-1}\nabla$$

for an appropriate diagonal matrix D and gradient vector ∇; K_1 and D_1 denote the submatrices of K and D corresponding to the nonzero elements of D. The block tridiagonal structure of A ensures that $A^{-1}\nabla$ and $A^{-1}K_1^T$ may be solved in $O(N^{3/2})$ time via Gaussian elimination; the key is that we do not need to compute A^{-1} explicitly, but only the solution y to the equation $Ay = \nabla$ (this may be done efficiently by the notation $A\backslash\nabla$ in Matlab). Finally, the inner matrix inverse $(D^{-1} + K_1 A^{-1} K_1^T)^{-1}$ is much easier to compute, due to the low rank of K_1.

Two more details of our implementation are worth noting. First, it is helpful to transform linearly to the color-whitened coordinates $\vec{z} = R\vec{x}$, where R denotes the inverse square root

[2] $K^T K$ is low-rank because the number of observed cones is much smaller (on the order of a hundred or so) than the dimensionality of the image $\vec{x}$. Moreover, in the Poisson case $K^T \text{diag}\left[\vec{n}./(b+K\vec{x}).^{-2}\right] K$ has even lower rank, since all but a few of the elements n_i are zero.

of the 3×3 color covariance matrix. This has the effect of increasing the sparseness of the A matrix (since the prior covariance matrix in the whitened space is a Kronecker product of the spatial covariance with an identity matrix, instead of with the full matrix C_c), resulting in a speedup of about a factor of two.

Second, it turns out that many of the early iterations of the Newton algorithm have to be truncated, since the Newton direction $H^{-1}\nabla$ frequently points out of the constraint space $\vec{x} \geq 0$. (In this case, we perform a line search along the direction $\vec{x}^{old} - H^{-1}\nabla$ instead of taking the full Newton step $\vec{x}^{new} = \vec{x}^{old} - H^{-1}\nabla$; this is still fast, since evaluation of the objective requires just $O(N)$ time.) Thus the most time-consuming step of the Newton optimization code is getting to the Newton regime in the first place. To reduce the total number of iterations we need to find a good initialization for the optimization. An effective method here is to use a coarse-to-fine approach: we perform the optimization at a coarse spatial discretization, then iteratively refine the discretization (optimizing the objective at each new discretization) until the desired level of spatial resolution is obtained. (Note that K here should be replaced by Kdx^2, where recall dx denotes the width of the spatial bins; this makes the filtering operation an integral operator, whose scaling behaves correctly as dx is decreased.)

This approach makes computation of $\vec{x}_{MAP}$ feasible for $\dim(\vec{x}) \sim 10^4$.

References

Ahmadian, Y., Pillow, J., and Paninski, L. (2008). Efficient Markov Chain Monte Carlo methods for decoding population spike trains. *Under review, Neural Computation*.

Badel, L., Richardson, M., and Gerstner, W. (2005). Dependence of the spike-triggered average voltage on membrane response properties. *Neurocomputing*, 69:1062–1065.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Oxford University Press.

Brainard, D., Williams, D., and Hofer, H. (2008). Trichromatic reconstruction from the interleaved cone mosaic: Bayesian model and the color appearance of small spots. *Vision Research*, In press.

Brown, E., Frank, L., Tang, D., Quirk, M., and Wilson, M. (1998). A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells. *Journal of Neuroscience*, 18:7411–7425.

Cossart, R., Aronov, D., and Yuste, R. (2003). Attractor dynamics of network up states in the neocortex. *Nature*, 423:283–288.

Czanner, G., Eden, U., Wirth, S., Yanike, M., Suzuki, W., and Brown, E. (2008). Analysis of between-trial and within-trial neural spiking dynamics. *Journal of Neurophysiology*, In press.

Davis, R. and Rodriguez-Yam, G. (2005). Estimation for state-space models: an approximate likelihood approach. *Statistica Sinica*, 15:381–406.

de la Rocha, J., Doiron, B., Shea-Brown, E., Josic, K., and Reyes, A. (2007). Correlation between neural spike trains increases with firing rate. *Nature*, 448:802–806.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal Royal Stat. Soc., Series B*, 39:1–38.

Frank, L. M., Eden, U. T., Solo, V., Wilson, M. A., and Brown, E. N. (2002). Contrasting Patterns of Receptive Field Plasticity in the Hippocampus and the Entorhinal Cortex: An Adaptive Filtering Approach. *J. Neurosci.*, 22(9):3817–3830.

Gao, Y., Black, M., Bienenstock, E., Shoham, S., and Donoghue, J. (2002). Probabilistic inference of arm motion from neural activity in motor cortex. *NIPS*, 14:221–228.

Huys, Q., Ahrens, M., and Paninski, L. (2006). Efficient estimation of detailed single-neuron models. *Journal of Neurophysiology*, 96:872–890.

Iyengar, S. (1985). Hitting lines with two-dimensional Brownian motion. *SIAM Journal on Applied Mathematics*, 45:983–989.

Jungbacker, B. and Koopman, S. (2007). Monte Carlo estimation for nonlinear non-Gaussian state space models. *Biometrika*, 94:827–839.

Kass, R. and Raftery, A. (1995). Bayes factors. *Journal of the American Statistical Association*, 90:773–795.

Kerr, J. N. D., Greenberg, D., and Helmchen, F. (2005). Imaging input and output of neocortical networks in vivo. *PNAS*, 102(39):14063–14068.

Koyama, S., Kass, R., and Paninski, L. (2008). Efficient computation of the most likely path in integrate-andfire and more general state-space models. *COSYNE*.

Lovasz, L. and Vempala, S. (2003). The geometry of logconcave functions and an $O^*(n^3)$ sampling algorithm. Technical Report 2003-04, Microsoft Research.

Moeller, J., Syversveen, A., and Waagepetersen, R. (1998). Log-Gaussian Cox processes. *Scandinavian Journal of Statistics*, 25:451–482.

Moeller, J. and Waagepetersen, R. (2004). *Statistical inference and simulation for spatial point processes*. Chapman Hall.

Murphy, S. and van der Vaart, A. (2000). On profile likelihood. *Journal of the American Statistical Association*, 95:449–465.

Nikitchenko, M. and Paninski, L. (2007). An expectation-maximization Fokker-Planck algorithm for the noisy integrate-and-fire model. *COSYNE*.

Ohki, K., Chung, S., Kara, P., Hubener, M., Bonhoeffer, T., and Reid, R. C. (2006). Highly ordered arrangement of single neurons in orientation pinwheels. *Nature*, 442(7105):925–928.

Paninski, L. (2004). Maximum likelihood estimation of cascade point-process neural encoding models. *Network: Computation in Neural Systems*, 15:243–262.

Paninski, L. (2005). Log-concavity results on Gaussian process methods for supervised and unsupervised learning. *Advances in Neural Information Processing Systems*, 17.

Paninski, L. (2006). The most likely voltage path and large deviations approximations for integrate-and-fire neurons. *Journal of Computational Neuroscience*, 21:71–87.

Paninski, L. (2007). Inferring synaptic inputs given a noisy voltage trace via sequential Monte Carlo methods. *Journal of Computational Neuroscience*, Under review.

Paninski, L., Fellows, M., Shoham, S., Hatsopoulos, N., and Donoghue, J. (2004). Superlinear population encoding of dynamic hand trajectory in primary motor cortex. *J. Neurosci.*, 24:8551–8561.

Paninski, L., Iyengar, S., Kass, R., and Brown, E. (2008). Statistical models of spike trains. In *Stochastic Methods in Neuroscience*. Oxford University Press.

Paninski, L., Pillow, J., and Lewi, J. (2007). Statistical models for neural encoding, decoding, and optimal stimulus design. In Cisek, P., Drew, T., and Kalaska, J., editors, *Computational Neuroscience: Progress in Brain Research*. Elsevier.

Pillow, J. and Paninski, L. (2007). Model-based decoding, information estimation, and change-point detection in multi-neuron spike trains. *Under review, Neural Computation*.

Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1992). *Numerical recipes in C*. Cambridge University Press.

Rahnama Rad, K. and Paninski, L. (2008). Efficient estimation of two-dimensional firing rate surfaces via Gaussian process methods. *COSYNE*.

Robert, C. and Casella, G. (2005). *Monte Carlo Statistical Methods*. Springer.

Smith, A. and Brown, E. (2003). Estimating a state-space model from point process observations. *Neural Computation*, 15:965–991.

Snyder, D. and Miller, M. (1991). *Random Point Processes in Time and Space*. Springer-Verlag.

Truccolo, W., Eden, U., Fellows, M., Donoghue, J., and Brown, E. (2005). A point process framework for relating neural spiking activity to spiking history, neural ensemble and extrinsic covariate effects. *Journal of Neurophysiology*, 93:1074–1089.

Vogelstein, J. and Paninski, L. (2007). Model-based optimal inference of spike times and calcium dynamics given noisy and intermittent calcium-fluorescence imaging. *Biophysical Journal*, Under review.

Ypma, A. and Heskes, T. (17-19 Sept. 2003). Iterated extended kalman smoothing with expectation-propagation. *Neural Networks for Signal Processing, 2003*, pages 219–228.