Supplemental Material for "Approximation Methods for State-Space Models"

Shinsuke Koyama[1,3], Lucia Castellanos Pérez-Bolde[2,3],

Cosma Rohilla Shalizi[1] and Robert E. Kass[1,2,3]

[1]Department of Statistics

[2]Department of Machine Learning

[3]Center for the Neural Basis of Cognition

Carnegie Mellon University, Pittsburgh, PA 15213

email: koyama@stat.cmu.edu

August 11, 2009

1. NUMERICAL COMPUTATION FOR SECOND DERIVATIVES

We describe the numerical algorithm for computing the Hessian matrix, as promised in Section 2.3 of the main manuscript.

The Laplace approximation requires the second derivative (or the Hessian matrix) of the log-likelihood function evaluated at its maximum. However, it is often difficult, and even more often tedious, to get correct analytical derivatives of the log-likelihood function. In such cases accurate numerical computations of the derivative may be used, as follows. Consider calculating the second derivative of $l(x)$ at x_0 for the one-dimensional case. For $n = 0, 1, 2, \ldots$ and $c > 1$, define the second central difference quotient,

$$A_{n,0} = [l(x_0 + c^{-n}h_0) + l(x_0 - c^{-n}h_0) - 2l(x_0)]/(c^{-n}h_0)^2,$$

and then for $k = 1, 2, \ldots, n$ compute

$$A_{n,k} = A_{n,k-1} + \frac{A_{n,k-1} - A_{n-1,k-1}}{c^{2(k+1)} - 1}. \tag{1}$$

When the value of $|A_{n,k} - A_{n-1,k}|$ is sufficiently small, $A_{n,k+1}$ is used for the second derivative.

This algorithm is an iterated version of the second central difference formula, often called *Richardson extrapolation*, producing an approximation with an error of order $O(h^{2(k+1)})$ (Dahlquist & Bjorck 1974).

In the d-dimensional case of a second-derivative approximation at a maximum, Kass (1987) proposed an efficient numerical routine which reduces the computation of the Hessian matrix to a series of one-dimensional second-derivative calculations. The trick is to apply the second-difference quotient to suitably-defined functions f of a single variable s as follows.

1. Initialize the increment $\boldsymbol{h} = (h_1, \ldots, h_d)$.

2. Find the maximum of $l(\boldsymbol{x})$, and call it $\hat{\boldsymbol{x}}$.

3. Get all unmixed second derivatives for each $i = 1$ to d, using the function

$$\begin{aligned} x_i &= \hat{x}_i + s \\ x_j &= \hat{x}_j \quad \text{for } j \text{ not equal to } i \\ f(s) &= l(\boldsymbol{x}(s)). \end{aligned}$$

   Compute the second difference quotient; then repeat and extrapolate until the difference in successive approximations meets a relative error criterion, as in (1); store as diagonal elements of the Hessian matrix array, $l''_{i,i} = f''(0)$.

4. Similarly, get all the mixed second derivatives. For each $i = 1$ to d, for each $j = i + 1$ to d, using the function

$$\begin{aligned} x_i &= \hat{x}_i + s/\sqrt{l''_{i,i}} \\ x_j &= \hat{x}_j + s/\sqrt{l''_{j,j}} \\ x_k &= \hat{x}_k \quad \text{for } k \text{ not equal to } i \text{ or } j \\ f(s) &= l(\boldsymbol{x}(s)). \end{aligned}$$

   Compute the second difference quotient; then repeat and extrapolate until difference in successive approximations is less than relative error criterion as in (1); store as off-diagonal elements of the Hessian matrix array, $l''_{i,i} = (f''(0)/2 - 1)\sqrt{l''_{i,i} l''_{j,j}}$.

In practice, the increment for computing the Hessian at time t would be taken to be $h_i = 0.1 \times \sqrt{v^{(i,i)}_{t|t-1}}$, $i = 1, 2, \ldots, d$, where $v^{(i,i)}_{t|t-1}$ is the (i,i)-element of the covariance matrix of the predictive distribution at time t.

2. LAPLACE'S METHOD

Here, we briefly describe Laplace's method, especially the details used in the proofs of Lemma 6 and Proposition 7.

We consider the following integral,

$$I(\gamma) = \int g(x) e^{-\gamma h(x)} dx, \tag{2}$$

where $x \in \mathbb{R}$; γ, the expansion parameter, is a large positive real number; $h(x)$ and $g(x)$ are independent of γ (or very weakly dependent on γ); and the interval of integration can be finite or infinite. Laplace's method approximates $I(\gamma)$ as a series expansion in descending powers of γ. There is a computationally efficient method to compute the coefficients in this infinite asymptotic expansion (Theorem 1.1 in (Wojdylo 2006)). Suppose that $h(x)$ has an interior minimum at x_0, and $h(x)$ and $g(x)$ are assumed to be expandable in a neighborhood of x_0 in series of ascending powers of x. Thus, as $x \longrightarrow x_0$,

$$h(x) \sim h(x_0) + \sum_{s=0}^{\infty} a_s (x - x_0)^{s+2},$$

and

$$g(x) \sim \sum_{s=0}^{\infty} b_s (x - x_0)^s,$$

in which $a_0, b_0 \neq 0$.

Let us introduce two dimensionless sets of quantities, $A_i \equiv a_i/a_0$ and $B_i \equiv b_i/b_0$, as well as the constants $\alpha_1 = 1/a_0^{1/2}$ and $c_0 = b_0/a_0^{1/2}$. Then the integral in 2) can be asymptotically expanded as

$$I(\gamma) \sim c_0 e^{-\gamma h(x_0)} \sum_{s=0}^{\infty} \Gamma(s + \frac{1}{2}) \alpha_1^{2s} c_{2s}^{*} \gamma^{-s-\frac{1}{2}},$$

where

$$c_s^* = \sum_{i=0}^{s} B_{s-i} \sum_{j=0}^{i} \left(\begin{array}{c} -\frac{s+1}{2} \\ j \end{array} \right) \mathcal{C}_{i,j}(A_1, \ldots) \ ,$$

where $\mathcal{C}_{i,j}(A_1, \ldots)$ is a partial ordinary Bell polynomial, the coefficient of x^i in the formal expansion of $(A_1 x + A_2 x^2 + \cdots)^j$. $\mathcal{C}_{i,j}(A_1, \ldots)$ can be computed by the following recursive formula,

$$\mathcal{C}_{i,j}(A_1, \ldots) = \sum_{m=j-1}^{i-1} A_{i-m} \mathcal{C}_{m,j-1}(A_1, \ldots) \ ,$$

for $1 \geq j \geq i$. Note that $\mathcal{C}_{0,0}(A_1, \ldots) = 1$, and $\mathcal{C}_{i,0}(A_1, \ldots) = \mathcal{C}_{0,j}(A_1, \ldots) = 0$ for all $i, j > 0$.

3. THE POPULATION VECTOR ALGORITHM

The *population vector algorithm* (PVA) is a standard method for neural decoding, especially for directionally-sensitive neurons like the motor-cortical cells recorded from in the experiments we analyze (Dayan & Abbott 2001, pp. 97–101). Briefly, the idea is that each neuron i, $1 \leq i \leq N$, has a preferred motion vector $\boldsymbol{\theta}_i$, and the expected spiking rate λ_i varies with the inner product between this vector and the actual motion vector $\boldsymbol{x}(t)$,

$$\frac{\lambda_i(t) - r_i}{\Lambda_i} = \boldsymbol{x}(t) \cdot \boldsymbol{\theta}_i \ , \tag{3}$$

where r_i is a baseline firing rate for neuron i, and Λ_i a maximum firing rate. ((3) corresponds to a cosine tuning curve.) If one observes $y_i(t)$, the actual neuronal counts over some time-window Δ, then averaging over neurons and inverting gives the *population vector*

$$\boldsymbol{x}_{\mathrm{pop}}(t) = \sum_{i=1}^{N} \frac{y(t) - r_i\Delta}{\Lambda_i\Delta}\boldsymbol{\theta}_i \ ,$$

which the PVA uses as an estimate of $\boldsymbol{x}(t)$. If preferred vectors $\boldsymbol{\theta}_i$ are uniformly distributed, then $\boldsymbol{x}_{\mathrm{pop}}$ converges on a vector parallel to $\boldsymbol{x}$ as $N \to \infty$, and is in that sense unbiased (Dayan & Abbott 2001, p. 101). If preferred vectors are not uniform, however, then in general the population vector gives a biased estimate.

4. REAL DATA ANALYSIS

Figure 1 shows trajectories of the true and estimated (by LGF, PF-100 and PVA) cursor position of the real data analysis. It is seen that the LGF provides better estimation than either the PF-100 or the PVA.

REFERENCES

Dahlquist, G., & Bjorck, A. (1974), *Numerial Methods*, New Jersey, Prentice Hall: Englewood.

Dayan, P., & Abbott, L. F. (2001), *Theoretical Neuroscience*, Cambridge, Massachusetts: MIT Press.

Kass, R. E. (1987), "Computing observed information by finite differences," *Communication in Statistics: Simulation and Computation*, 2, 587–599.

Wojdylo, J. (2006), "On the coefficients that arise from Laplace's method," *Journal of Computational and Applied Mathematics*, 196.

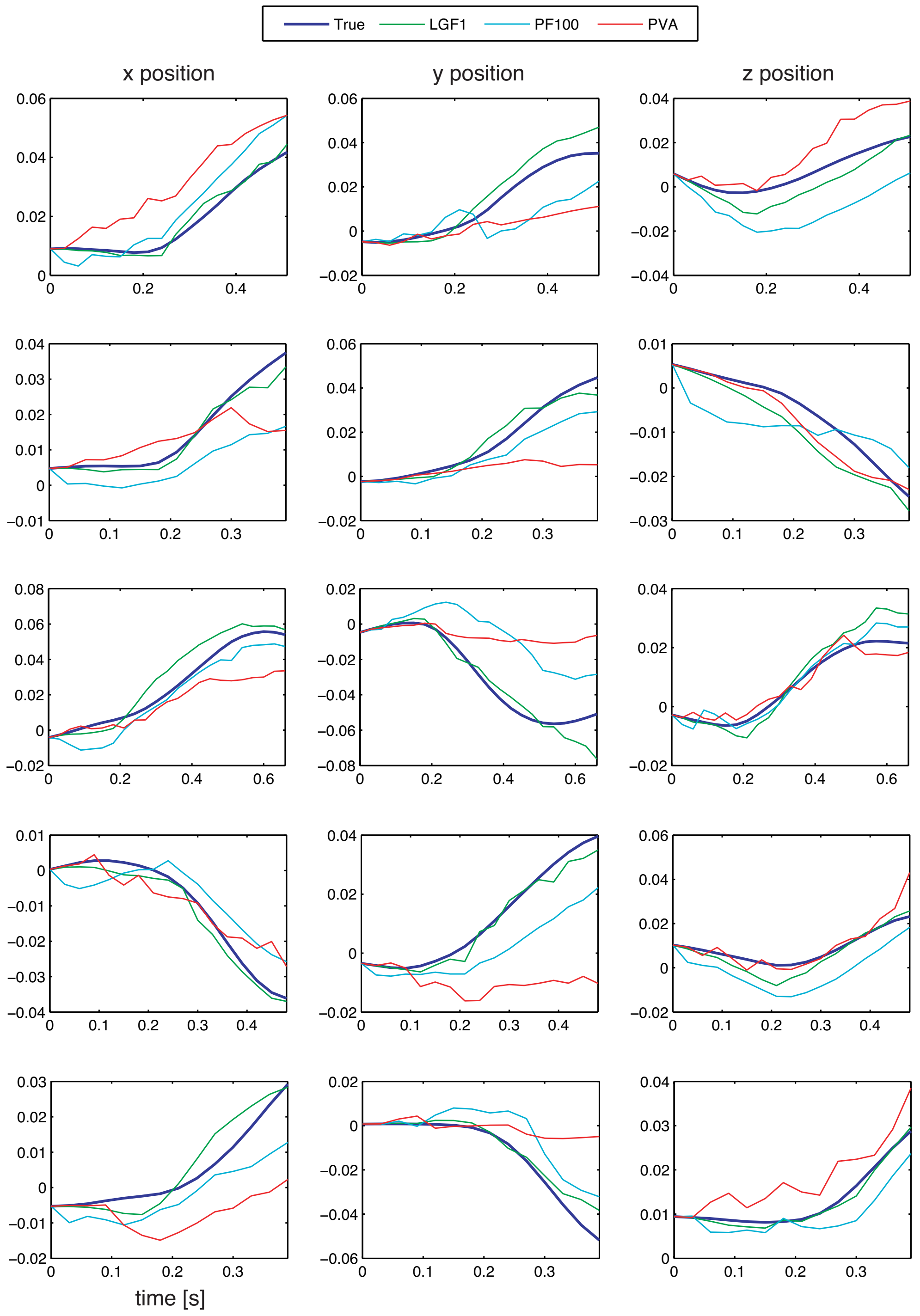

Figure 1: The trajectories of the cursor position. “True”: actual trajectory. “LGF1”: trajectories estimated by first-order LGF, respectively. “PF100”: trajectory estimated by the particle filter with 100 particles. “PVA”: trajectory estimated by the population vector algorithm. The trajectories estimated by the LGF2 are not shown; they are similar to those estimated by the LGF1.