Kalman filtering for maximum likelihood estimation given corrupted observations.

E. E. Holmes, National Marine Fisheries Service

Introduction

The Kalman filter is used to extend likelihood estimation to cases with hidden states, such as when observations are corrupted and the true population size is unobserved. The following algorithm is based on section 3.4 in Harvey (1989), which was used by Lindley (2003) for estimation for population processes. The Kalman filter is well-known and widely used in engineering and computer science applications. There are a multitude of books on the Kalman filter, including Harvey (1989). One of the more penetrable introductions of the Kalman filter alone (but not on maximum likelihood estimation) is chapter 1 of Maybeck (1979).

The state-space model

The diffusion approximation for a stochastic exponential growth model can be written as a linear state space model (written in the notation familiar in the engineering literature):

$$x_{t+1} = x_t + B + w_t, \quad \text{where} \quad w_t \sim \text{normal}(0, Q) \qquad [1]$$

$$y_t = x_t + v_t, \quad \text{where} \quad v_t \sim \text{f}(0, R) \qquad [2]$$

where $x_t = \log N_t$ is the true log population size and $y_t = \log O_t$ is the log observations of the population size. B is μ, the mean population growth rate. Q is the σ^2, otherwise known as the process error or environmental variability. R is the variability associated with sampling error or other non-process error. Only y_t, is observed; the underlying parameters, B, Q, and R, and the underlying true population size, x_t, is hidden. If we make the assumption that v_t is normally distributed, then the model is a linear Gaussian state-space model**.**

We can calculate the probability of the observed time series, $\{y\}_1^T \equiv \{y_1, y_2, \ldots, y_T\}$, as follows:

$$P(\{y\}_1^T) = \prod_{t=1}^{T} P(y_t \mid \{y\}_1^{t-1}) \qquad [3]$$

where $P(y_t \mid \{y\}_1^{t-1})$ is the probability of y_t given all the observations before time t and $P(y_1 \mid \{y\}_1^0) \equiv P(y_1)$. Denote the expected value of $(y_t \mid \{y\}_1^{t-1})$ as $\tilde{y}_t^{t-1}$. The conditional probability $(y_t \mid \{y\}_1^{t-1})$ is distributed normal with a mean $\tilde{y}_t^{t-1}$ and some variance, denoted F_t^{t-1}, which depends on the particular parameters, $\Psi = \{B, Q, R\}$, that generated the data. Thus, the probability of the time series given a particular set of parameters, Ψ, is

$$P(\{y\}_1^T \mid \Psi) = \prod_{t=1}^{T} \exp\left\{-\frac{(y_t - \tilde{y}_t^{t-1})^2}{2F_t^{t-1}}\right\}(2\pi|F_t^{t-1}|)^{-1/2}\, d\vartheta \qquad [4]$$

from the probability density of a normal with mean $\tilde{y}_t^{t-1}$ and variance F_t^{t-1}. The log likelihood of ψ given the data, $\{y\}_1^T$, is

$$\log L(\Psi \mid \{y\}_1^T) = -\frac{T}{2}\log 2\pi - \frac{1}{2}\sum_{t=1}^{T}\log|F_t^{t-1}| - \frac{1}{2}\sum_{t=1}^{T}\frac{(y_t - \tilde{y}_t^{t-1})^2}{F_t^{t-1}} + \text{a constant.} \qquad [5]$$

For Eqn. 5, we need estimates of $\tilde{y}_t^{t-1} = E(y_t \mid \{y\}_1^{t-1})$ and $F_t^{t-1} = E(y_t y_t \mid \{y\}_1^{t-1})$. Observe from Eqn. 1 that

$$\begin{aligned} E(y_t \mid \{y\}_1^{t-1}) &= E(x_t \mid \{y\}_1^{t-1}) \\ E(y_t y_t \mid \{y\}_1^{t-1}) &= E(x_t x_t \mid \{y\}_1^{t-1}) + R \end{aligned} \qquad [6]$$

The Kalman filter below gives optimal estimates of $E(x_t \mid \{y\}_1^{t-1})$ and $E(x_t x_t \mid \{y\}_1^{t-1})$ which are then used in Eqn. 5 to calculate the log likelihood of ψ.

The maximum likelihood estimates of B, Q, and R are found by using some type of maximization routine on Eqn. 6 to find the set of parameters $\psi = \{B, Q, R\}$ that maximize the likelihood. Matlab code for this algorithm is given at the end of this appendix.

The Kalman filter

First, some notation:

$$\{y\}_1^\tau \equiv \{y_1, y_2, \ldots, y_\tau\}$$
$$x_t^\tau \equiv E[x_t \mid \{y\}_1^\tau]$$
$$V_t^\tau \equiv \mathrm{E}[x_t x_t \mid \{y\}_1^\tau]$$

The Kalman recursion: Start at t = 1 and step forward to *T*. Assume an initial $x_t \equiv \pi_1$ and initial $V_1^0 \equiv V_1$ to start the recursion. One could let these be free variables and find the maximum likelihood values when maximizing Eqn. 6, but that is not done here and the algorithm should not be very sensitive to these starting values. At each step, compute:

$$x_t^{t-1} = \begin{cases} \pi_1 & \text{for } t = 1 \\ x_{t-1}^{t-1} + B & \text{for } t > 1 \end{cases}$$
$$V_t^{t-1} = \begin{cases} V_1 & \text{for } t = 1 \\ V_{t-1}^{t-1} + Q & \text{for } t > 1 \end{cases}$$
$$K_t = \frac{V_t^{t-1}}{\left(V_t^{t-1} + R\right)}$$
$$x_t^t = x_t^{t-1} + K_t\left(y_t - x_t^{t-1}\right)$$
$$V_t^t = V_t^{t-1} - K_t V_t^{t-1}$$

This is the well-known Kalman filter, but it looks a little different than what you'll see in engineering texts. First generally it is assumed that y_t is a series of measurements from multiple instruments, thus the Kalman filter is always written in matrix form. Here since y_t is one measurement, it can be written in scalar form. Second, the Kalman filter is usually presented for

the model $x_{t+1} = Ax_t + Bu_t + w_t$, $y_t = Cx_t + v_t$. In this application, A=1, C=1 and u_t =1, so the filter is simplified quite a bit.

References

Harvey, A. C. 1991. Forecasting, structural time series models and the Kalman filter. Cambridge University Press, Cambridge, UK.

Maybeck, P. S. 1979. Stochastic models, estimation and control. Volume 1. Academic Press, New York, USA.

Matlab code

```
function [mu,sigma2,sigma2np]=kalman_ests(data)

y=log(data);

%Start with some reasonable initial parameter estimates
muest=mean(y(2:end)-y(1:(end-1)));
tmp1=var(y(2:end)-y(1:(end-1)));
tmp4=var(y(5:end)-y(1:(end-4)));
sigma2est=(tmp4-tmp1)/3;
sigma2npest=(var(y(2:end)-y(1:(end-1)))-max(0.0001,sigma2est))/2;
pi1=max(0.0001,sigma2est)+max(0.0001,sigma2npest); %var of y(1)

%log transform the variances so the search algorithm doesn't give negative
% variances
startvals=[muest log(max(0.0001,sigma2est)) log(max(0.0001,sigma2npest))];
%fminsearch is a Nelder-Mead minimization matlab function
a=fminsearch('kalman_loglik',startvals,[],y,y(1),pi1);

MLmuest=a(1);
MLsigma2est=exp(a(2));
MLsigma2npest=exp(a(3));

function negloglik = kalman_loglik(v,y,initx,V1)

T=length(y);
B = v(1); %mu
Q = exp(v(2)); %s2
R = exp(v(3)); %s2np

%initialize
xtt=zeros(1,T);  Vtt=zeros(1,T); xtt1=zeros(1,T); Vtt1=zeros(1,T);
xtT=zeros(1,T);  VtT=zeros(1,T); J=zeros(1,T); Vtt1T=zeros(1,T);
```

```
Ft=zeros(1,T); vt=zeros(1,T);

%forward pass gets you E[x(t) given y(1:t)]
x10=initx;
V10=V1;
for(t=1:T)
   if(t==1)
      xtt1(1) = initx; %denotes x_1^0
      Vtt1(1) = V1; %denotes V_1^0
   else
      xtt1(t) = xtt(t-1) + B; %xtt1 denotes x_t^(t-1); Harvey 3.2.2a
      Vtt1(t) = Vtt(t-1) + Q; %Harvey 3.2.2b
   end
   Kt = Vtt1(t)/(Vtt1(t)+R);
   Ft(t) = Vtt1(t)+R;
   vt(t) = y(t)-xtt1(t);
   xtt(t) = xtt1(t) + Kt*(y(t) - xtt1(t)); %Harvey 3.2.3a
   Vtt(t) = Vtt1(t)-Kt*Vtt1(t); %Harvey 3.3.3b
end

%Calculate negative log likelihood
negloglik = (1/2)*sum(vt.^2./Ft) + (1/2)*sum(log(abs(Ft))) + (T/2)*log(2*pi);
```