Neural Decoding by Kalman Filter and

Sequential Monte Carlo Methods

Department of Statistics, FSU

Xue Huang

Feb 26, 2012

1. **Problem Statement**

Neurons are remarkable among the cells of the body in their ability to propagate signals rapidly. Neural coding and decoding is a neuroscience-related field concerned with how sensory and other information is represented in the brain by networks of neurons. The main goal of studying neural coding is to characterize the relationship between the stimulus and the individual or ensemble neuronal responses and the relationship among electrical activity of the neurons in the ensemble.

In this report, we are given the observed neural activity from brain cortex in the research animals, we perform a Bayesian analysis on these data to understand the brain mechanism and make statistical inferences about the external behaviors.

1. **Methodology**

Both for the train set and test set, they have two variables: *kin* and *rate*. *Kin* () is the 2-d kinematic state of hand movement at time t, which includes *x-position, y-position, x-velocity* and *y-velocity*. *Rate ()* is the spiking rates of 42 neurons in the primary motor cortex at the same time, where the rate at each time is the number of spikes within 70*ms*.

A classical Kalman Filter can be used to model the hand kinematic state and neural activity as follows:



where 

In the training data set, both hand state and neural activity are known. We can use the close-form formula to estimate the model parameter A, H, W, Q from the training data set.

Once the parameters are identified, we can perform the neural decoding on the testing data. That is, we will use neural activity to infer the movement behaviors of the hand. Two inference methods are used in this report:

1. *Kalman Filter Algorithm*

The Kalman Filter addresses the general problem of trying to estimate the state of a discrete-time controlled process that is governed by the linear stochastic difference equation



With a measurement that is



The random variables and represent the process and measurement noise. They are assumed to be independent of each other with normal probability 

The Kalman Filter estimated a process by using a form of feedback control: the filter estimates the process at some time and then obtains feedback in the forms of measurements. So, the algorithms for the Kalman Filter can be divided into two parts: time update and measurement update. The time update are responsible for projecting forward the current state and error covariance estimated to obtain the priori estimated for the next step. The measurement update step is responsible for the feedback that is for incorporating a new measurement into the priori estimate to obtain an improved posteriori estimate.

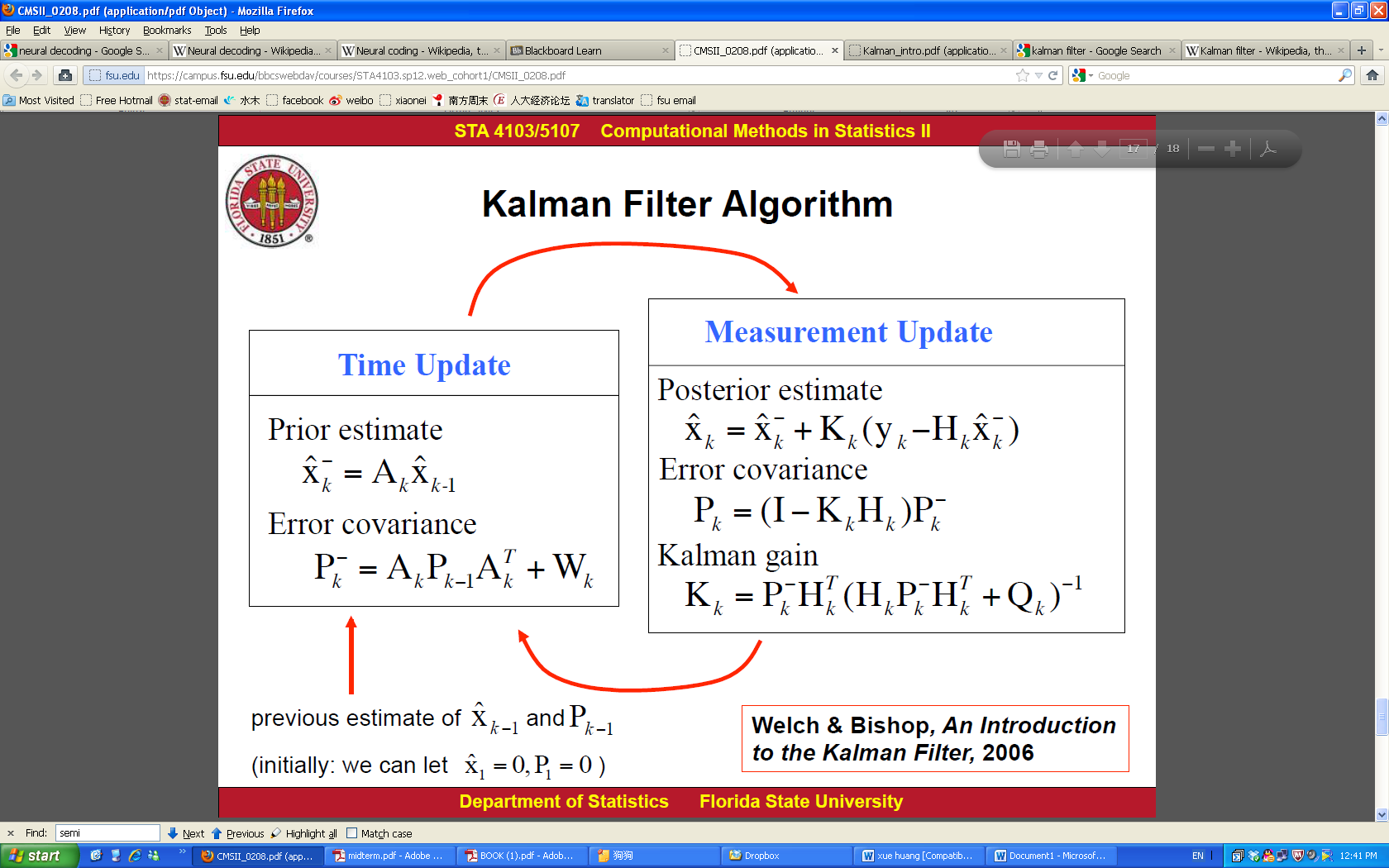


Figure 1: The ongoing discrete Kalman Filter cycle. The time update projects the current state estimate ahead in time. The measurement update adjusts the projected estimate by an actual measurement at that time.

1. *Sequential Monte Carlo Method (SMCM)*

The Sequential Monte Carlo aims to estimate the sequence of hidden parameters  based only on the observed data. The algorithm of Sequential Monte Carlo is in this form:

1. Generate n samples. Set t=0.
2. Prediction: Generate the prediction set using:



1. Update: Compute the weights, and normalize them using.

* Estimate using.
* Resample from the set with probabilities n times to obtain the samples 

1. Set t=t+1, and return to Step (ii).
2. *Comparison between Kalman filter and Sequential Monte Carlo Method*

* Estimation accuracy

Error is commonly used as a criterion to measure the estimation accuracy:

Let denote the true state and denote the estimate. Then,



We compute the estimation accuracy of the positions using Error.

* Computation time

1. **Experiment Results**
2. *Kalman Filter Algorithm*

Using the Kalman Filter, we get the estimate for the hand movement. Figure 2 shows the comparison between the estimation from Kalman filter and the true data value; we can see that the estimated value is very close to true hand position.



Figure 2: Plot of the true and estimated hand positions using Kalman Filter

1. *Sequential Monte Carlo Method (SMCM)*

Figure 3-6 shows the plots of the true data value and estimated hand positions using SMCM with sample size n= 20, 50, 100 and 500. From the evolution of the estimate of the hand positions, we can see that as the sample size increase, Sequential Monte Carlo Method performs better.



Figure 3: Plot of the true and estimated hand positions using SMCM with n=20



Figure 4: Plot of the true and estimated hand positions using SMCM with n=50



Figure 5: Plot of the true and estimated hand positions using SMCM with n=100



Figure 6: Plot of the true and estimated hand positions using SMCM with n=500

1. *Comparison between Kalman filter and Sequential Monte Carlo Method*

From Table 1 and 2, we can see Kalman filter method over perform sequential Monte Carlo methods both in estimation accuracy and computation time, which is a more accurate and fast method.

* Estimation accuracy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | x-position | y-postion |
|  |  | Kalman Filter | 0.4908 | 0.8384 |
|  |  | Sequential MCM n=20 | 0.2557 | 0.7638 |
|  |  | n=50 | 0.361 | 0.8071 |
|  |  | n=100 | 0.4223 | 0.8214 |
|  |  | n=500 | 0.4705 | 0.8329 |

Table 1: Estimation accuracy of Kalman filter and SMCM (n=20, 50, 100 and500)

* Computation time

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | time |
|  |  | Kalman Filter | 2.1023s |
|  |  | Sequential MCM n=20 | 6.1118s |
|  |  | n=50 | 14.7782 |
|  |  | n=100 | 29.9702s |
|  |  | n=500 | 151.7672s |

Table 2: Computation time of Kalman filter and SMCM (n=20, 50, 100 and500)

1. **Summary**

In this report, we perform a Bayesian analysis on the data to make statistical inferences about the external behaviors of the research animals. Two inference methods are used: Kalman filter and Sequential Monte Carlo Method. Both of them are good estimate for the hand movement.

At last, we make comparisons between these two methods from two aspects: estimation accuracy and computation time. We find that Kalman filter method performs better and consistent result than Sequential Monte Carlo Methods. But as sample size increase, Sequential Monte Carlo Methods has increasing higher estimation accuracy. Besides, Kalman filter compute faster than Sequential Monte Carlo method, which shows Kalman filter is a more efficient Bayesian analysis method under linear assumptions.

1. **Appendix (Matlab Code)**

clear all; close all; clc;

load('C:\Users\xue\Desktop\midterm\_2\_train.mat');

[M,N]=size(rate);

mean\_pos = mean(kin(:,1:2));

kin(:,1:2) = kin(:,1:2) - ones(M,1)\*mean\_pos;

mean\_rate = mean(rate);

rate = rate - ones(M,1)\*mean\_rate;

x= kin';

y= rate';

% estimates of A,W,H,Q

Ahat= x(:,2:M)\*x(:,1:M-1)' \* inv(x(:,1:M-1)\*x(:,1:M-1)');

What=(x(:,2:M)\*x(:,2:M)'-Ahat\*x(:,1:M-1)\*x(:,2:M)')/(M-1);

Hhat=y\*x'\*inv(x\*x');

Qhat=(y\*y'-Hhat\*x\*y')/M;

% Using Kalman Filter to infer the hand movement

clear ('x','kin','y','rate','M','N');

load('C:\Users\xue\Desktop\midterm\_2\_test.mat');

[M,C]=size(rate);

mean\_pos = mean(kin(:,1:2));

kin(:,1:2) = kin(:,1:2) - ones(M,1)\*mean\_pos;

mean\_rate = mean(rate);

tic;

rate = rate - ones(M,1)\*mean\_rate;

x2=kin';

y2=rate';

M=910;

xh=[11;11;0.3;-0.5];

xmh=xh; P=1\*eye(4); Pm=P; K=1\*ones(4,42);

for k=2:910

xmh(:,k)=Ahat\*xh(:,k-1);

Pm(:,:,k)=Ahat\*P(:,:,k-1)\*Ahat'+What;

K(:,:,k)=Pm(:,:,k)\*Hhat'\*inv(Hhat\*Pm(:,:,k)\*Hhat'+Qhat);

P(:,:,k)=(eye(4)-K(:,:,k)\*Hhat)\*Pm(:,:,k);

xh(:,k)=xmh(:,k)+K(:,:,k)\*(y2(:,k)-Hhat\*xmh(:,k));

end

figure(1)

hold on;

tl = ['px','py','vx','vy'];

for i=1:4

subplot(4,1,i)

plot(1:M,x2(i,:), 1:M,xh(i,:));

xlabel('time');

ylabel(tl(2\*(i-1)+1:2\*(i-1)+2));

xlim([0,M]);

legend('True Data','Kalman Filter Estimation');

end

t1=toc;

% Estimation accuracy for Kalman Filter using R-square Error

R2m = 1-sum((x2-xh).^2,2)./sum((x2-mean(x2,2)\*ones(1,M)).^2,2);

% Using Sequential Monte Carlo Methods to infer the hand movement

clear ('x2','kin','y2','rate','N','xh','xmh','P','Pm','K','k');

load('C:\Users\xue\Desktop\midterm\_2\_test.mat');

[M,C]=size(rate);

rate = rate - ones(M,1)\*mean\_rate;

d=size(kin,2);

x3=kin';

y3=rate';

tic;

% n=20;

% n=50;

% n=100;

n=500;

% initialize x as random numbers

X=zeros(d,M);

S=zeros(d,n,M);

xh=ones(d,n,M);

xh(:,:,1)=randn(d,1,n);

S(:,:,1)= randn(d,1,n);

theta=zeros(d,M);

w=ones(n,M)/n;

w2=ones(n,M);

weight=ones(M,1);

for t=2:M

%prediction, generate the prediction set

for j=1:n

mu=ones(4,n);

mu(:,j)=Ah\*xh(:,j,t-1);

S(:,j,t)=mvnrnd(mu(:,j),(What+What')/2);

%update compute the weight and normalize

w(j,t)=mvnpdf(y3(:,t),Hhat\*S(:,j,t),(Qhat+Qhat')/2);

end

weight(t)=sum(w(:,t));

w2(:,t)=w(:,t)/weight(t);

theta(:,t)=S(:,:,t)\*w2(:,t);

% resample

for j=1:n

r=randsample([1:n],1,true,w2(:,t));

xh(:,j,t)=S(:,r,t);

end

end

X=theta;

X(1:2,:) = X(1:2,:) + mean\_pos'\*ones(1,M);

figure(2);

subplot(2,1,1);

xp=[X(1,:);x3(1,:)];

plot(xp');

legend('SMCM Estimation','True data')

title('x-position');

subplot(2,1,2);

yp=[X(2,:);x3(2,:)];

plot(yp');

legend('SMCM Estimation','True data')

title('y-position');

t2=toc;

R\_SMCM=zeros(2,1);

for i=1:2;

r1=0;

for k=1:M

r1=r1+(x3(i,k)-X(i,k))^2;

end

r2=0;

for k=1:M

r2=r2+(x3(i,k)-mean(x3(i,:)))^2;

end

R\_SMCM(i)=1-(r1/r2);

end