732A96 Lab 4

Emil K Svensson

5 October 2017

The Implementation

```r
set.seed(12345)




sample_emission <- function(z, sd = 1){
  mean_adjust <- round(runif(1,-1,1))
  rnorm(1, mean = z + mean_adjust, sd = sd)

}

sample_transition <- function(z, sd = 1){
  mean_adjust <- round(runif(1,0,2))
  rnorm(1, mean = z + mean_adjust, sd = sd)
}

#Time <- 100

generate_data <- function(nobs, sd_emission = 1, sd_transition = 1){

X <- rep(NA,nobs)
Z_true <- c(runif(1,0,100),rep(NA,nobs-1))

X[1] <- sample_emission(Z_true[1], sd = sd_emission)

for (i in 2:nobs){

  Z_true[i] <- sample_transition(Z_true[i-1 ], sd_transition)
  X[i] <- sample_emission(Z_true[i], sd_emission)
}

return(list(X = X,
            Z_true = Z_true,
            sd_used = c(emission = sd_emission, transition = sd_transition)))
}


particle_filter <- function(nobs,X, sd_emission,sd_transition){
# Now we discard Z and try to estimate it using only X
Wt <- matrix(NA,ncol = nobs, nrow = nobs)
Z <- matrix(NA, ncol = nobs, nrow = nobs + 1)
Z[1,] <- runif(100, min = 0, max = 100)

for (t in 1:nobs){
```

```
# Emission model, the probability of the observation x_t given z_t
emission <- sapply(Z[t,], function(zt) {
     (dnorm(X[t],zt , sd_emission) +
     dnorm(X[t],zt - 1, sd_emission) +
     dnorm(X[t],zt + 1, sd_emission)) / 3
})

Wt[t,] <- emission / sum(emission)


Z_old <- sample(x = Z[t,], size = nobs, replace = TRUE, prob = Wt[t,])

Z[t+1,] <- sapply(Z_old, function(x) sample_transition(x, sd  = sd_transition))

}

return(list(X = X,
            Z_est = Z[1:nobs,1:nobs],
            Wt = Wt)
)

}

# Calculate the weight Wt[t] for each particle
# Sample Z[t] particles with the weights
# Sample Z[t+1] with the sampled Z[t] particles

# Transition model, the probability of the hidden state z_{t+1} given z_t, the uncertainty of the obser
```

The special case

```
special_filter <- function(nobs,X, sd_emission,sd_transition){
# Now we discard Z and try to estimate it using only X
Wt <- matrix(NA,ncol = nobs, nrow = nobs)
Z <- matrix(NA, ncol = nobs, nrow = nobs + 1)
Z[1,] <- runif(100, min = 0, max = 100)

for (t in 1:nobs){

# Emission model, the probability of the observation x_t given z_t
emission <- sapply(Z[t,], function(zt) {
     (dnorm(X[t],zt , sd_emission) +
     dnorm(X[t],zt - 1, sd_emission) +
     dnorm(X[t],zt + 1, sd_emission)) / 3
})

# All weights are equal in the special case.

Wt[t,] <- rep(1/nobs, nobs)
```

```
Z_old <- sample(x = Z[t,], size = nobs, replace = TRUE, prob = Wt[t,])

Z[t+1,] <- sapply(Z_old, function(x) sample_transition(x, sd  = sd_transition))

}

return(list(Z_est = Z[1:nobs,1:nobs],
            Wt = Wt)
)

}
```

Running the functions

```
gen_data_sd1 <- generate_data(nobs = 100, sd_emission = 1, sd_transition = 1)

est_data_sd1 <- particle_filter(nobs = 100, X = gen_data_sd1$X,
                sd_emission = 1, sd_transition = 1)

est_data_sd5 <- particle_filter(nobs = 100, X = gen_data_sd1$X,
                sd_emission = 5, sd_transition = 1)

est_data_sd50 <- particle_filter(nobs = 100, X = gen_data_sd1$X,
                sd_emission = 50, sd_transition = 1)

est_data_special <- special_filter(nobs = 100, X = gen_data_sd1$X,
                sd_emission = 50, sd_transition = 1)
```

```
library(rgdal)
library(rasterImage)
library(png)
library(animation)


r2d2 <- readPNG("r2d22.png")
#pman <- readPNG("pman.png")

#off <- 0.1

#Good advice, dont mess with the off parameter.
plot_robot <- function(img,X,Z,Wt, off = 0.1){

# Some preparations before ploting

# Making the image to a raster object
r2d2 <- as.raster(img)

#Coordinates for the true position of the robot
C <- coordinates(data.frame(X = X, y = 0.01))

# The Expected position of the robot
rm_z <- rowSums(Wt * Z)
```

```
for (i in 1:length(rm_z)){
plot(x = 1:250, y = rep(0,250),
     type = "l",
     col = "white",
     ylim = c(-0.6,1),
     xlab = "Position x",
     ylab = "",
     main = paste("Time :",i))

abline(h = 0, col = "black")
points(x = rm_z[i], y = 0.5, col = "white")
points(x = Z[i,], y = rep(-0.5,length(Z[i,])))

rasterImage(r2d2,
            xleft = C[i,1]-off ,
            ybottom = C[i,2]-off,
            xright = C[i,1]+off+50,
            ytop = C[i,2]+off+0.25,
            interpolate = FALSE)

points(x = C[i,1], y = -0.1, col = "green")
points(rm_z[i], y = -0.2, col = "red")

if(i == 1){
  Sys.sleep(5)
}

Sys.sleep(0.1)
}

}
```

All plots are avaliable at my GitHub LINK in form of gifs. With a standard deveation of 1 the expected value, represented by the red dot is close to the true position (the green dot) basicly from the begining of the observations.

For the standard deviation equal to 5 we have similar results as for sd of 1 but with a bit more uncertainty and a expected Z that is not always very close to the true state.

For the standard deviation equal to 50 we have a harder time following the true location but it is still a good aproximation after a couple of itterations.

For the special case with equal weights the predictions are useless, it is like running another independent robot and will continue doing that since we are not using the information from the emission model in our sampling.

Ploting Gifs

```
plot_robot(img = r2d2,
           X = gen_data_sd1$X,
           Z = est_data_sd1$Z,
           Wt = est_data_sd1$Wt,
           off = 0.1)

plot_robot(img = r2d2,
           X = gen_data_sd1$X,
           Z = est_data_sd5$Z,
           Wt = est_data_sd5$Wt,
           off = 0.1)

plot_robot(img = r2d2,
           X = gen_data_sd1$X,
           Z = est_data_sd50$Z,
           Wt = est_data_sd50$Wt,
           off = 0.1)

plot_robot(img = r2d2,
           X = gen_data_sd1$X,
           Z = est_data_special$Z,
           Wt = est_data_special$Wt,
           off = 0.1)
```

Saving Gifs

```
animation::saveGIF({
  plot_robot(img = r2d2,
           X = gen_data_sd1$X,
           Z = est_data_sd1$Z,
           Wt = est_data_sd1$Wt,
           off = 0.1)
},
movie.name = "robot_sd1.gif")

animation::saveGIF({
  plot_robot(img = r2d2,
           X = gen_data_sd1$X,
           Z = est_data_sd5$Z,
           Wt = est_data_sd5$Wt,
           off = 0.1)
},
movie.name = "robot_sd5.gif")

animation::saveGIF({
  plot_robot(img = r2d2,
           X = gen_data_sd1$X,
           Z = est_data_sd50$Z,
           Wt = est_data_sd50$Wt,
```

```
           off = 0.1)
},
movie.name = "robot_sd50.gif")

animation::saveGIF({
  plot_robot(img = r2d2,
           X = gen_data_sd1$X,
           Z = est_data_special$Z,
           Wt = est_data_special$Wt,
           off = 0.1)
},
movie.name = "robot_special.gif")
```

A Kalman filter implementation

```
update_mu <- function(A, B=0, mu=0, mu_previous) {
  return(A%*%mu_previous)
}

update_sigma <- function(A, R, sigma_previous) {
  return(A%*%sigma_previous%*%t(A) + R)
}

calculate_kalman_gain <- function(sigma, C, Q) {
  sigma%*%t(C) %*% solve(C%*%sigma%*%t(C) + Q)
}

scale_mu_with_kalman_gain <- function(mu_bar, K, z, C) {
  mu_bar + K%*%(z-C%*%mu_bar)
}

scale_sigma_with_kalman_gain <- function(K, C, sigma_bar) {
  dimensions <- dim(K)
  I <- diag(1, nrow=dim(K)[1], ncol=dim(C)[2])

  return((I-K%*%C)%*%sigma_bar)
}

kalman_filter <- function(A, B, C, R, Q, mu_0, sigma_0, z) {
  n <- length(z)
  mu_list <- list()
  sigma_list <- list()

  mu_list[[1]] <- mu_0
  sigma_list[[1]] <- sigma_0

  for (t in 2:(n+1)) {
    mu_bar <- update_mu(A=A, mu_previous=mu_list[[t-1]])
    sigma_bar <- update_sigma(A=A, R=R, sigma_list[[t-1]])
    K <- calculate_kalman_gain(sigma=sigma_bar, C=C, Q=Q)
    mu_list[[t]] <- scale_mu_with_kalman_gain(mu_bar, K, z[t-1], C)
```

```r
    sigma_list[[t]] <- scale_sigma_with_kalman_gain(K, C, sigma_bar)
  }
  return(structure(list(
    mu <- mu_list,
    sigma <- sigma_list
  )))
}


# Test
A <- matrix(c(1, 1, 0, 1), byrow=TRUE, nrow=2)
B <- matrix(c(0, 0, 0, 0), byrow=TRUE, nrow=2)
C <- matrix(c(1, 0), byrow=TRUE, nrow=1)
R <- matrix(c(0.035, 0.035+3.06*10^(-12), 3.06*10^(-12), 3.06*10^(-12)), byrow=TRUE, nrow=2)
Q <- 0.035
mu_0 <- matrix(c(10, 0), byrow=TRUE, nrow=2)
sigma_0 <- matrix(c(10^2, 0, 0, 10^2), byrow=TRUE, nrow=2)

load("../data/Radiation_data.Rda")
z <- Radiation_data$dose

debugonce(kalman_filter)
kalman_values <- kalman_filter(A, B, C, R, Q, mu_0, sigma_0, z)
kalman_mean <- unlist(lapply(kalman_values[[1]], function(x) x[[1]]))

plot(Radiation_data$dose, pch=16)
lines(kalman_mean, col="red")
```