732A96: Lab 4
Advanced Machine Learning

Carles Sans Fuentes

October 9, 2017

Assignment

Question 1

Question The purpose of the lab is to put in practice some of the concepts covered in the lectures. To do so, you are asked to implement the particle filter for robot localization. For the particle filter algorithm, please check Section 13.3.4 of Bishop's book and/or the slides for the last lecture on SSMs.

Run it for $T = 100$ time steps to obtain $z_{1:100}$ (i.e. states) and $x_{1:100}$ (i.e. observations). Use the observations (i.e. sensor readings) to identify the state (i.e. robot location) via particle filtering. Use 100 particles. For each time step, show the particles, the expected location and the true location. Repeat the exercise after replacing the standard deviation of the emission model with 5 and then with 50. Comment on how this affects the results. Finally, show and explain what happens when the weights in the particle filter are always equal to 1, i.e. there is no correction.

Answer:

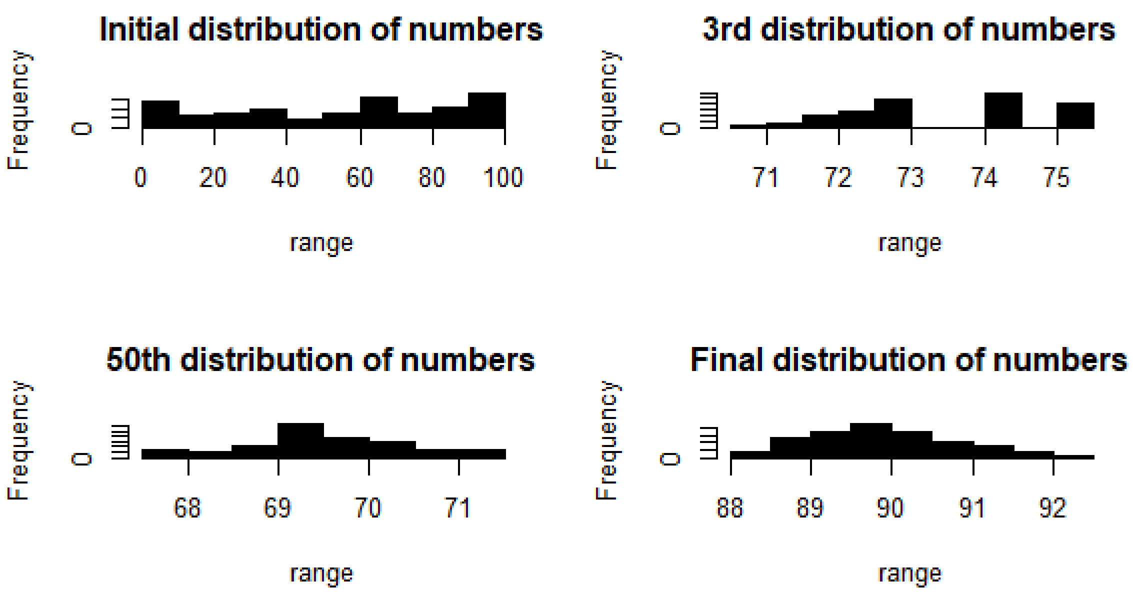

Figure 1: Histogram of the distribution of parameters with sd equal to 1

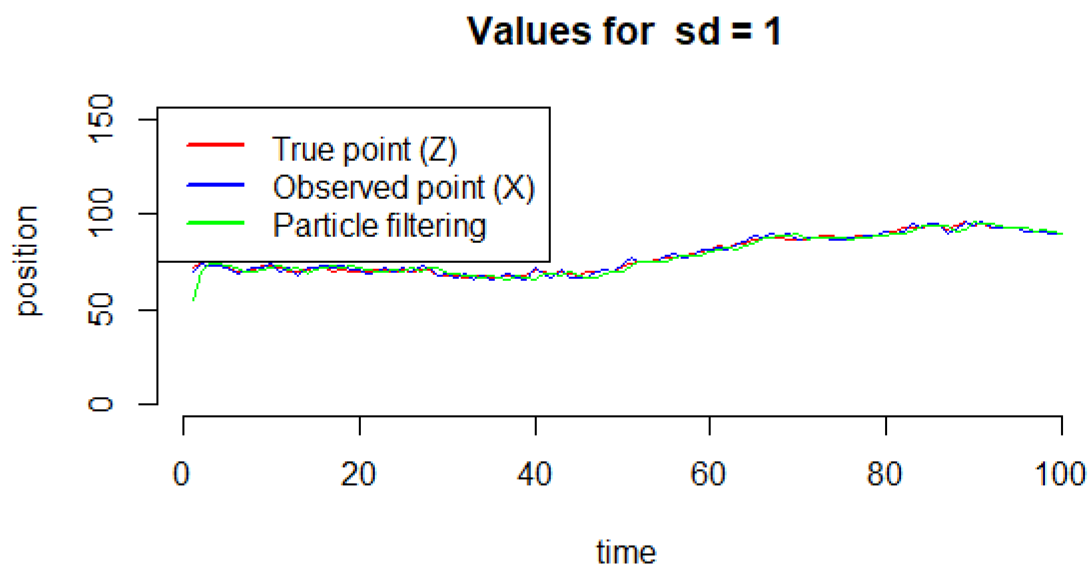

Figure 2: Plot X, Z and the particle filtering for 100 simulations with sd equal to 1

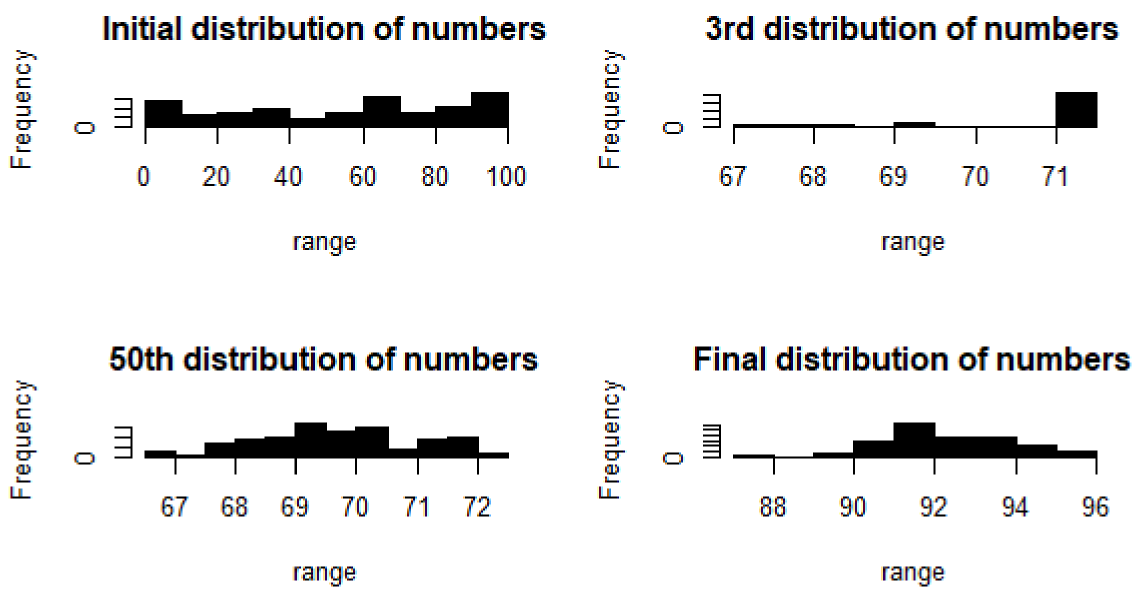

Figure 3: Histogram of the distribution of parameters with sd equal to 5

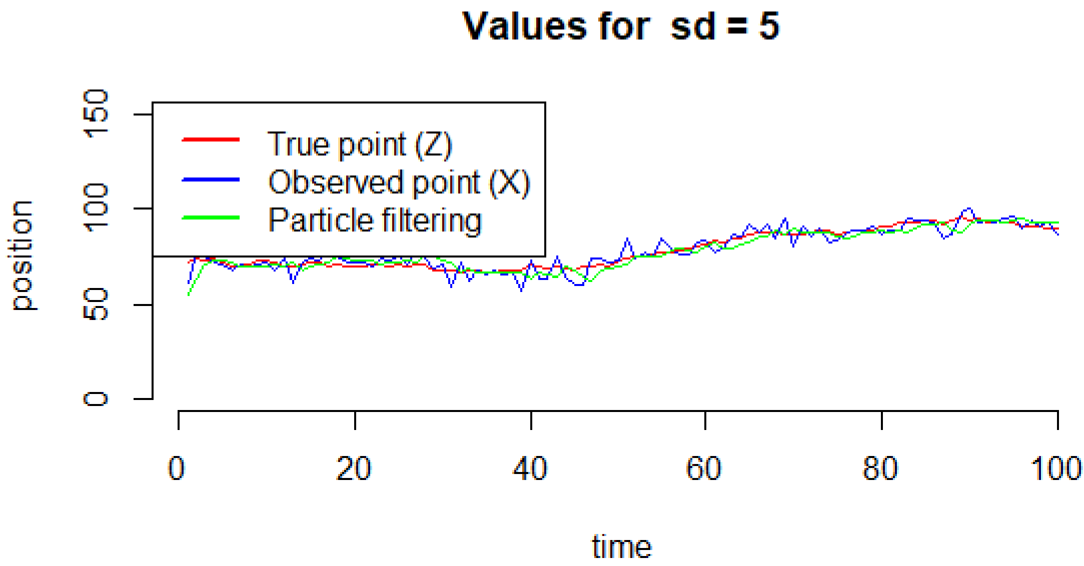

Figure 4: Plot X, Z and the particle filtering for 100 simulations with sd equal to 5

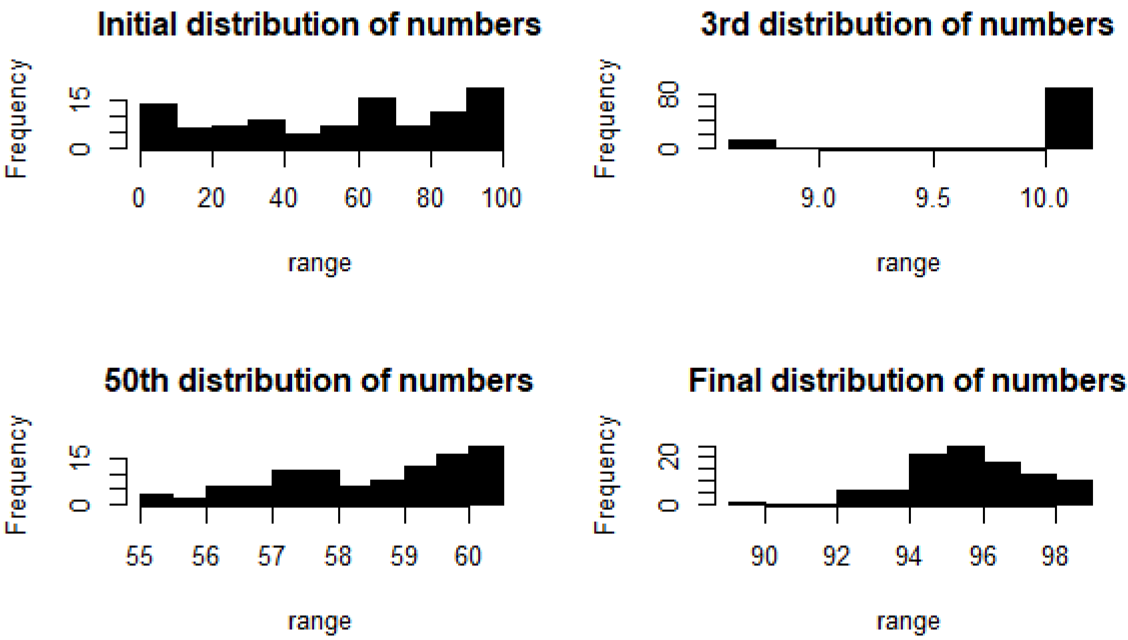

Figure 5: Histogram of the distribution of parameters with sd equal to 50

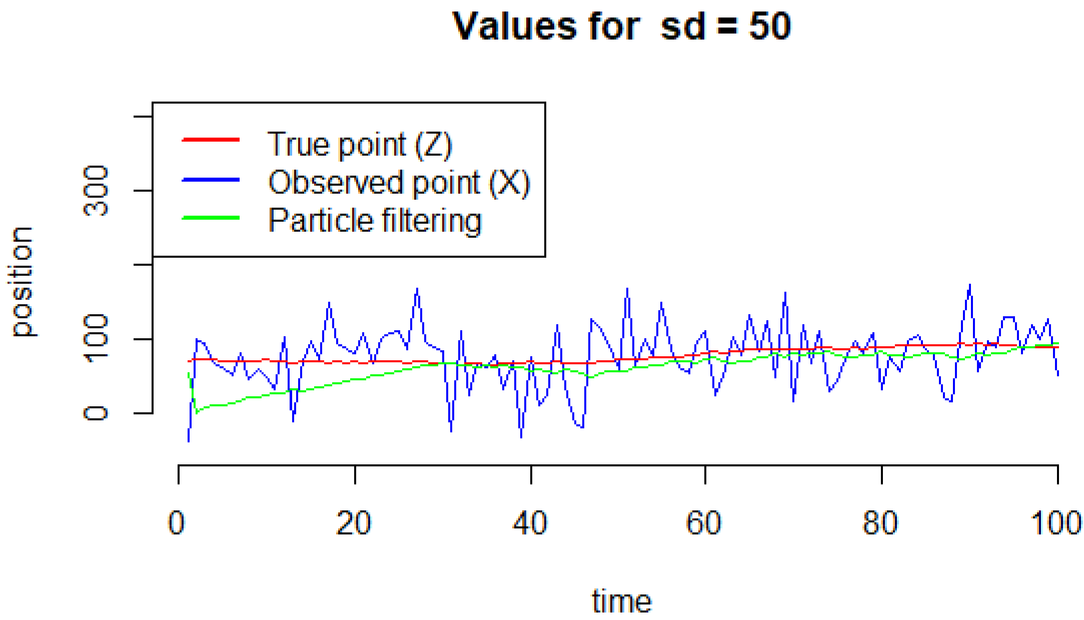

Figure 6: Plot X, Z and the particle filtering for 100 simulations with sd equal to 50

When increasing the standard deviation of the emission function, we are both less certain about the location of the robot givebn that our sensor is worse. Intuitively, if the sensor is worse (e.g. higher variance), our prediction will also be worse. For that , we can see that the higher the sd, the worse and more bumped our X is (blue line) and the more the particle filtering lasts to converge to the real position. Nevertheless, it ends up giving a good prediction.

If we set every time the weight to be 1 in all cases (e.g. it means all numbers have the same weights), our prediction will be much worse since we are not taking into account the conditional probabilities of the previous points.

Contributions

All results and comments presented have been developed and discussed together by the members of the group.

Appendix

Poisson regression-the MCMC way

```
##########
#####1
set.seed(12345)
#######Generating model
#initial model
p1<-runif(1,0,100)

# Transition

sd_T<-1
sd_E<-50

Transition<- function(n,p0, sd_2){
  z<- integer(n)
  z[1]<-p0
  for(i in 2:n){
    p<- sample(c(z[i-1]-1,z[i-1],z[i-1]+1),1)
    z[i]<-rnorm(1,p,sqrt(sd_2))
    }
  return(z)
}
myZ<-Transition(100, p0= p1, sd_2= sd_T**2)
##Emission model


Emission<- function(vectorz, sd_2){
  n<- length(vectorz)
  x<- integer(n)
  for(i in 1:n){
    p<- sample(c(vectorz[i]-1,vectorz[i],vectorz[i]+1),1)
    x[i]<-rnorm(1,p,sqrt(sd_2))
  }
  return(x)
}
x_t<- Emission(myZ, sd_2 = sd_E**2)

#####################
weights<- rep(0.01,100)
X<-runif(100,0,100)
calculationprob<- function(simulations, init_weights, grid, sd_E, sd_T, x_t= x_t){
  n<- length(grid)
  parameters<- matrix( ncol = n,nrow = simulations)
  newweigths<- matrix(ncol = n,nrow = simulations)

  newweigths[1,]<-init_weights/sum(init_weights)
  parameters[1,]<-sample(x =grid,size = n, replace=TRUE, prob = newweigths[1,])
 ### sample(dnorm(parameters[1,],grid) Left that
  xt<- integer(100)
  for(i in 2:simulations){

    for(j in 1:simulations){
      xt[j]<-Transition(2,parameters[i-1,j],sd_T)[2]
    }
    newweigths[i,]<-dnorm(x_t[i-1], xt, sqrt(sd_E))/sum(dnorm(x_t[i-1], xt,sqrt(sd_E)))
    parameters[i,]<-sample(xt, size =n, replace=TRUE, prob = newweigths[i,])

  }
  result<- list(parameters = parameters, weights=newweigths)
  return(result)
}
sd_2T<-1
sd_2E<-1

trial<-calculationprob(simulations =100, init_weights = weights, grid= X,
                       sd_T = sd_T, sd_E = sd_E,x_t= x_t)
parameter_est<-trial$parameters
parameter_est

plotting<- function(Z=myZ,  X=x_t , particle= trial$parameters){
  n<-1:length(Z)
  for(i in n){
    plot(0, xlim= c(1,100), ylim = c(0,150 ),bty='n',pch='',ylab='sep representation for clearness',xlab='position')
    points(y = 20, x = Z[i], col ="red")
    points(y = 30, x = X[i], col ="blue")
    points(y =rep(40, 100), x = particle[i,], col = "green")
    legend("topleft", # places a legend at the appropriate place
           c("True point (Z) ", "Estimated point (X)", "Particle filtering"), # puts text in the legend
           lty=c(1,1), # gives the legend appropriate symbols (lines)
```

```
            lwd=c(2.5,2.5),col=c("Red"," blue", "Green")) # gives the legend lines the correct color and width
             Sys.sleep(0.2)
  }
}
par(mfrow=c(1,1))
plotting()


par(mfrow=c(1,1))
plot(0, xlim= c(1,100), ylim = c(-50,400),
     bty='n',pch='',ylab='position',xlab='time', main = paste0("Values for  sd = ", sd_E))
lines(x = 1:100, y = myZ, col ="red")
lines(x = 1:100, y = x_t, col ="blue")
lines(x =1:100, y = apply(trial$parameters,1,mean), col = "green")
legend("topleft", # places a legend at the appropriate place
       c("True point (Z) ", "Observed point (X)", "Particle filtering"), # puts text in the legend
       lty=c(1,1), # gives the legend appropriate symbols (lines)
       lwd=c(2.5,2.5),col=c("Red"," blue", "Green")) # gives the legend lines the correct color and width

par(mfrow=c(2,2))
hist(trial$parameters[1,], main= "Initial distribution of numbers", xlab = "range", col = "black")
hist(trial$parameters[3,], main= "3rd distribution of numbers", xlab = "range", col = "black")
hist(trial$parameters[50,], main= "50th distribution of numbers", xlab = "range", col = "black")
hist(trial$parameters[100,], main= "Final distribution of numbers", xlab = "range", col = "black")
```