Advanced Machine Learning, Lab 4

Rebin Hosini (rebho 150)

2017-10-05

Implementation of particle filter

We the links presented below we can visualize how the true position, the expected and the particles moves as a function of time. when we have a standard deviation of 1 the particles becomes compressed around the robot position after time unit and it follows this pattern for the rest of the time units. When we instead increase the standard deviation the particles stays wide spread throughout the entire time. The reason for this is that an increase in standard deviation implies that we become more uncertain about the position and the same happens. When we set equal weights we can see that the particles does not converge to the true position of the robot. This can also be seen in the (Sigma = 1, W = equal) video.

Sigma = 1
Sigma = 5
Sigma = 50
Sigma = 1, W = Equal

```
sigma <- 1
sigmaE <- 1 #change sigma for each

sTransModel <- function(sd = 1, z_p){
  pos <- sample(1:3, size = 1, prob = c(1/3,1/3,1/3))
  dist <- list("d1" = rnorm(1, mean = z_p, sd = sd),
               "d2" = rnorm(1, mean = z_p+1, sd = sd),
               "d3" = rnorm(1, mean = z_p+2, sd = sd))
 return(dist[[pos]])
}

sEmissModel <- function(sd = sigmaE, z_t){
  pos <- sample(1:3, size = 1, prob = c(1/3,1/3,1/3))
  dist <- list("d1" = rnorm(1, mean = z_t, sd = sd),
               "d2" = rnorm(1, mean = z_t+1, sd = sd),
               "d3" = rnorm(1, mean = z_t-2, sd = sd))
 return(dist[[pos]])
}

emissModel <- function(sd = sigmaE, z_t, x_t){
    (dnorm(x_t, mean = z_t , sd = sd)+
     dnorm(x_t, mean = z_t + 1 , sd = sd)+
     dnorm(x_t, mean = z_t - 1, sd = sd))/3
}

init <- function(n = 1, min = 1, max = 100){
  runif(n = n, min = min, max = max)
}

steps <- 100
z <- c()
```

```
x <- c()
z[1] <- init() # z_1
x[1] <- sEmissModel(z_t = z[1])

for (i in 2:steps){
  z[i] <- sTransModel(z_p = z[i-1], sd = sigma) # z_t
  x[i] <- sEmissModel(z_t = z[i], sd = sigmaE) # x_t

}

PF <- function(steps, sd = c(sigma,sigmaE)){
  weights <- matrix(0, ncol = steps, nrow = steps)
  Z <- matrix(ncol = steps, nrow = steps+1)
  Z[1,] <- init(n = steps, 0, 100)
  for (i in 1:steps){
    emission <- vapply(Z[i,, drop = TRUE],
                       FUN = function(zVal){emissModel(sd = sd[2], z_t = zVal, x_t = x[i])},
                       FUN.VALUE = numeric(1))
    weights[i,] <- emission/sum(emission)
    #Sample from current Z with probabilities weighted Z
    WZ <- sample(Z[i,], replace = TRUE, prob = weights[i,, drop = TRUE], size = steps)
    Z[i+1,] <- sapply(WZ, function(zprev) {sTransModel(sd = sd[1], z_p = zprev)})
  }
  return(list("Z" = Z, "W" = weights, "X" = x))
}

PFW <- function(steps, sd = c(sigma, sigmaE)){
  weights <- matrix(0.01, ncol = steps, nrow = steps)
  Z <- matrix(ncol = steps, nrow = steps+1)
  Z[1,] <- init(n = steps, 0, 100)
  for (i in 1:steps){
    emission <- vapply(Z[i,, drop = TRUE],
                       FUN = function(zVal){emissModel(sd = sd[2], z_t = zVal, x_t = x[i])},
                       FUN.VALUE = numeric(1))
    #Sample from current Z with probabilities weighted Z
    WZ <- sample(Z[i,], replace = TRUE, prob = weights[i,, drop = TRUE], size = steps)
    Z[i+1,] <- sapply(WZ, function(zprev) {sTransModel(sd = sd[1], z_p = zprev)})
  }
  return(list("Z" = Z, "W" = weights, "X" = x))
}

plotData <- function(outModel){
  E <- apply(outModel$W*outModel$Z[-nrow(outModel$Z),], MARGIN = 1, sum)
  xPos <- matrix(ncol = 2, nrow = 100)
  xPos[,1] <- outModel$X #Do not forget to change sigma outside of the PF fun
  xPos[,2] <- 0.2
  return(list("pos" = xPos, "E" = E))
}

############################ Run robot

plotRobot <- function(outModel,steps = 100){
  o <- outModel
```

```
  X <- o$X
  Z <- o$Z
  for (i in 1:steps){
    plot(x = 1:250, y = rep(0,250), type = "l",
         col = "white",ylab = "y", xlab = "x")
    abline(h = 0, col = "black")
    #Expected
    points(x = plotData(o)$E[i], y = 0.8, type = "o")
    text(x = plotData(o)$E[i], y = 0.8, labels = "Expected", cex = 1)
    #True robot state
    abline(v = plotData(o)$pos[i], col = "orange")
    text(x = plotData(o)$pos[i], y = 0.11, labels = "Robot", cex = 1, col = "orange")
    #Particles
    points(x = Z[i,], y = rep(-0.1,length(Z[i,])), col = "red", pch = 16)
    Sys.sleep(0.3)
  }
}
#debugonce(PF)
set.seed(123456)
out <- PF(steps = 100)
outW <- PFW(steps = 100)

#setwd("/Users/Rebin/Desktop/Statistics & Datamining/Advanced Machine Learning/AdvanceML-732A96 - group
#animation::saveHTML({plotRobot(out)}, img.name = "sd1", htmlfile = "SD1.html")
#setwd("/Users/Rebin/Desktop/Statistics & Datamining/Advanced Machine Learning/AdvanceML-732A96 - group
#animation::saveHTML({plotRobot(out)}, img.name = "sd5", htmlfile = "SD5.html")
#setwd("/Users/Rebin/Desktop/Statistics & Datamining/Advanced Machine Learning/AdvanceML-732A96 - group
#animation::saveHTML({plotRobot(out)},img.name = "sd50", htmlfile = "SD50.html")
#setwd("/Users/Rebin/Desktop/Statistics & Datamining/Advanced Machine Learning/AdvanceML-732A96 - group
#animation::saveHTML({plotRobot(outW)}, img.name = "sd1w", htmlfile = "SD1W.html")

############################
```

‘‘‘