Advanced Machine Learning - Lab 03

G. Hari Prasath (hargo729), Julius Kittler (julki029), Hector Plata (hecpl268) and Maximilian Pfundstein (maxpf364)

2019-10-04

Contents

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 state space models (SSMs). The robot moves along the horizontal axis according to the following SSM:

Transition Model:

$$p(z_t|t_{t-1}) = \frac{\mathcal{N}(z_t|z_{t-1}, 1) + \mathcal{N}(z_t|z_{t-1} + 1, 1) + \mathcal{N}(z_t|z_{t-1} + 2, 1)}{3}$$

Emission Model:

$$p(x_t|z_t) = \frac{\mathcal{N}(x_t|z_t, 1) + \mathcal{N}(x_t|z_t - 1, 1) + \mathcal{N}(x_t|z_t + 1, 1)}{3}$$

Initial Model:

$$p(z_1) = \text{Uniform}(0, 100)$$

1 Implementing the State Space Model

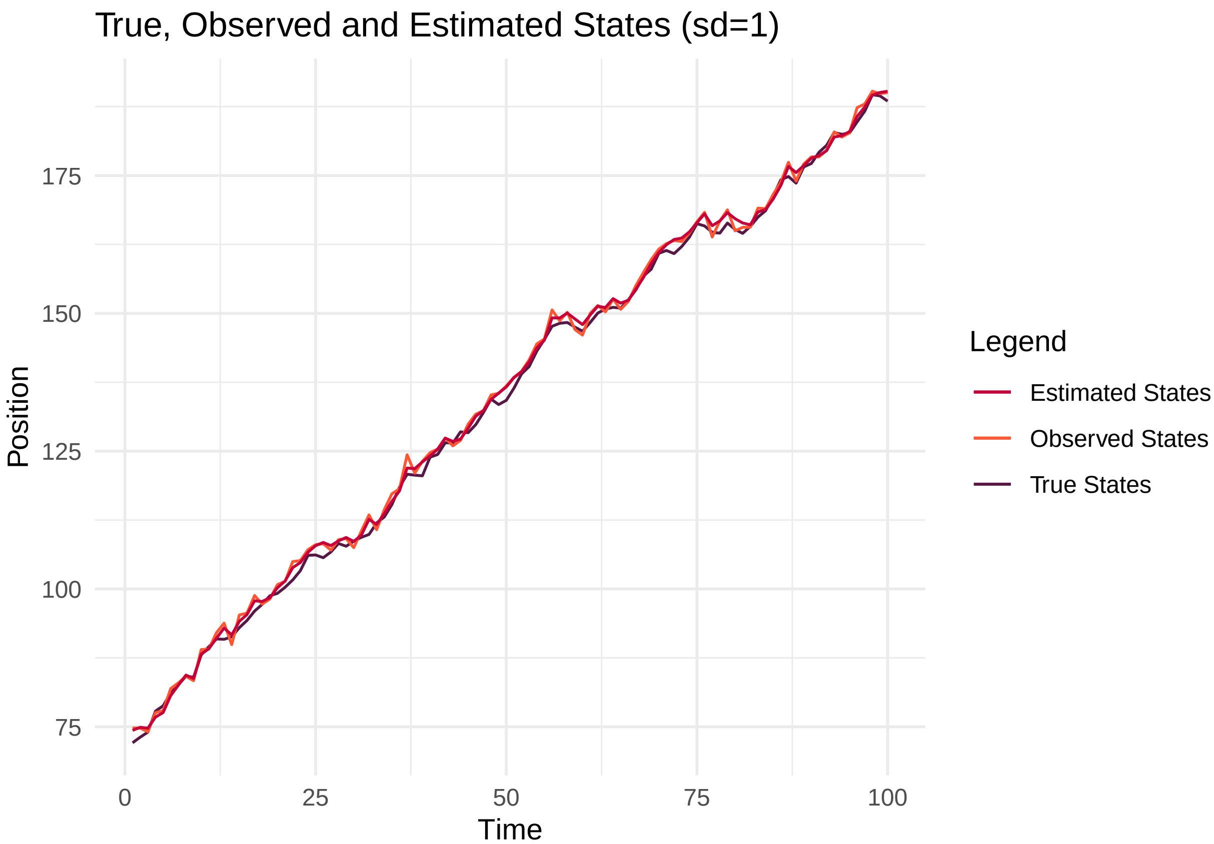
Task: Implement the SSM above. Simulate it for $T = 100$ time steps to obtain $z_{1:100}$ (i.e., states) and $x_{1:100}$ (i.e., observations). Use the observations (i.e., sensor readings) to identify the state (i.e., robot location) via particle filtering. Use 100 particles. Show the particles, the expected location and the true location for the first and last time steps, as well as for two intermediate time steps of your choice.

```
# Parameter setup
partice_sample_size = 100 # M
iterations = 100 # T

rtransition = function(z_t_1, sd=1) {
  return(rnorm(1, sample(c(0,1,2), size=1)+z_t_1, sd=sd))
}

remission = function(n, z_t, sd=1) {
```

```r
  return(rnorm(1, sample(c(0,1,2), size=1)+z_t, sd=sd))
}

demission = function(x_t, z_t, sd=1) {
  return(sum(dnorm(x_t, c(0, -1, 1)+z_t, sd=sd))/3)
}

rinit = function(n) {
  return(runif(n, 0, 100))
}

sample_observations = function(n, sd_emission=1) {
  # First observation
  states = vector(length = n) # Z
  observations = vector(length = n) # X

  states[1] = rinit(1)
  observations[1] = remission(1, states[1], sd_emission)

  for (i in 2:n) {
    states[i] = rtransition(states[i-1])
    observations[i] = remission(1, states[i], sd_emission)
  }

  return(data.frame(states = states, obs = observations))
}

particle_filter = function(observations, T=100, M=100, sd_emission=1, corr=TRUE) {

  X = matrix(nrow = T, ncol = M) # Posterior believe
  X_bar = matrix(nrow = T, ncol = M) # Prior believe
  W = matrix(nrow = T, ncol = M)
  Z = vector(length = T)

  for (t in 1:T) {

    if (t == 1) {
      # Initialization
      X_temp = rinit(M)
      # Prediction
      X_bar[t,] = sapply(X_temp, rtransition)
    }
    else {
      # Prediction
      X_bar[t,] = sapply(X[t-1,], rtransition)
    }

    # Importance Weight
    W[t, ] = sapply(X_bar[t,], demission, z_t = observations[t], sd=sd_emission)
    # Normalize
    W[t, ] = W[t, ]/sum(W[t, ])
    # Correction
    if (corr) {
```

```r
      prob = W[t,]
    }
    else {
      prob = rep(1, length(W[t,]))
    }
    X[t,] = sample(X_bar[t,], M, prob = prob, replace = TRUE)
    # Taken from Bishop
    Z[t] = as.numeric(W[t, ] %*% X[t, ])

  }

  return(list(X=X, X_bar=X_bar, W=W, Z=Z))
}
```

```r
sd_emission = 1

observations = sample_observations(iterations, sd_emission)

res = particle_filter(observations$obs,
                      T = iterations,
                      M = partice_sample_size,
                      sd_emission = sd_emission)
```

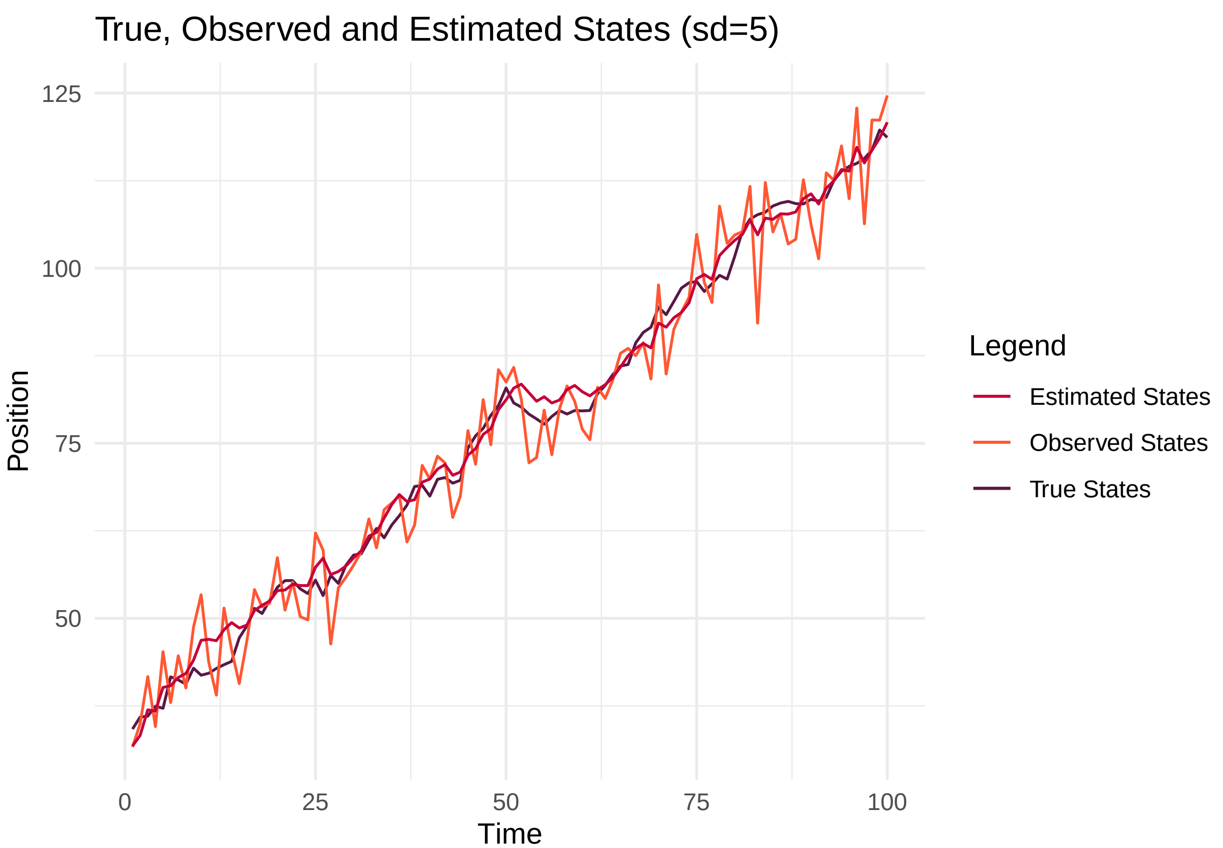


2 Different Standard Deviations

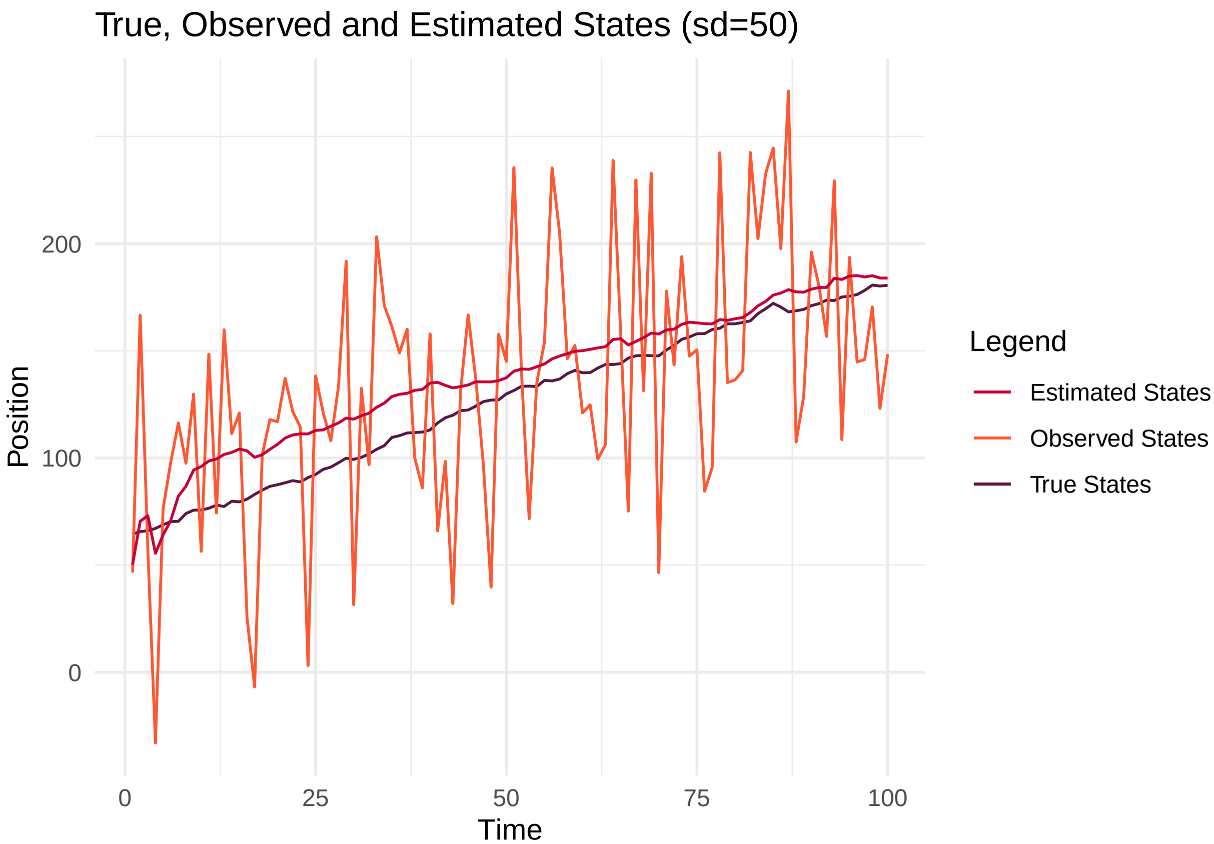
Task: Repeat the exercise above replacing the standard deviation of the emission model with 5 and then with 50. Comment on how this affects the results.

Answer: Increasing the standard deviation of the emission model increases the variance of the observed values. The conclusion is that our particle filter needs more iterations to converge to the true state. Especially during the first iterations it can be seen, that the uncertainty is high and thus the estimate of the robot is inaccurate.

```r
sd_emission = 5

observations = sample_observations(iterations, sd_emission)

res = particle_filter(observations$obs,
                      T = iterations,
                      M = particle_sample_size,
                      sd_emission = sd_emission)
```




```r
sd_emission = 50

observations = sample_observations(iterations, sd_emission)

res = particle_filter(observations$obs,
                      T = iterations,
                      M = partice_sample_size,
                      sd_emission = sd_emission)
```



3 Omit Correction

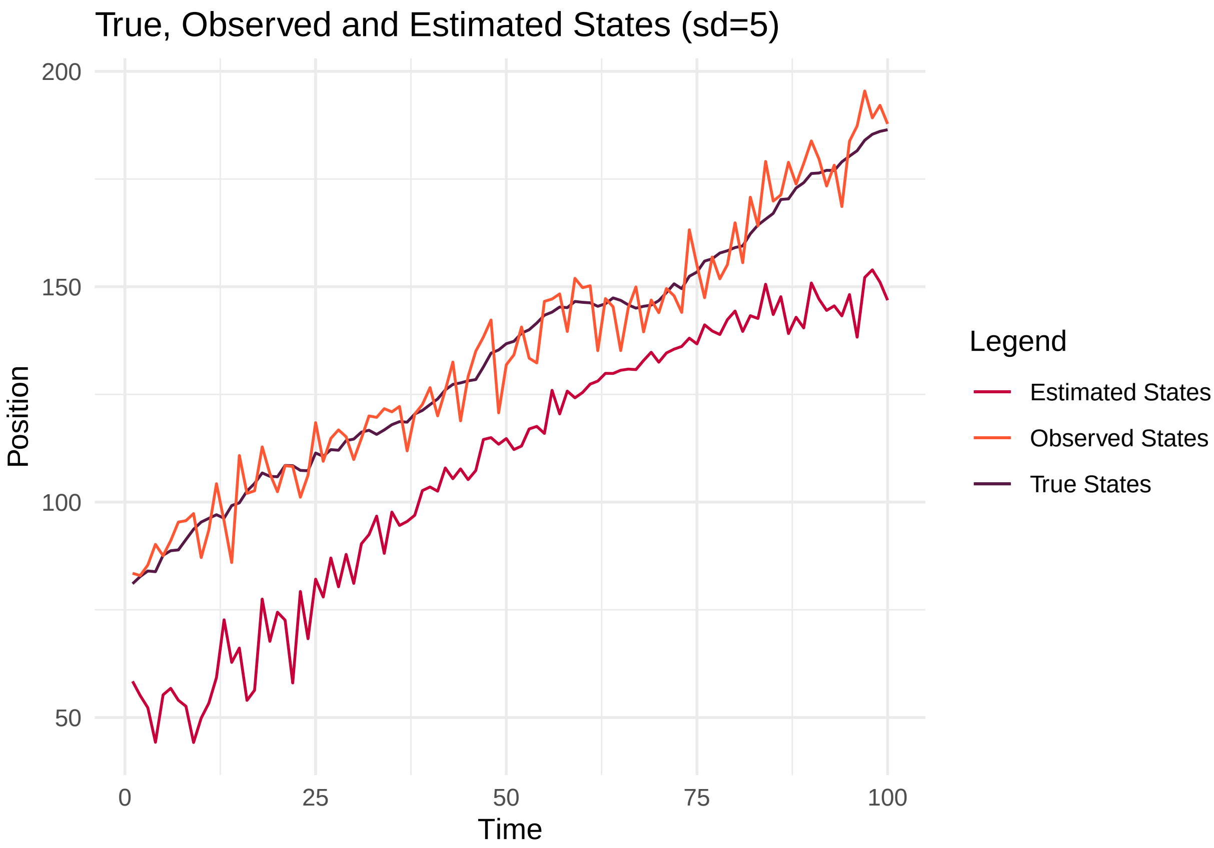
Task: Finally, show and explain what happens when the weights in the particle filter are always equal to 1, i.e. there is no correction.

Answer: As we can see in the plot below, we now have a lag error. This can have two reasons:

- Either the dynamic model does not fit the reality.
- The reliability of the model is flawed. Either the real variance is higher or lower.

Forcing all draws to have the same probability, does not count in for the correct observed uncertainty and thus makes our model narrower (case two from the bullet points). The reason that we sample with probabilities is to draw from the correct posterior, but as we change the odds, our MCMC sampling does not give us the desired results. The same behaviour can be seen using a Kalman Filter instead of a Particle Filter. Also the variance just decreases until a specific point and teh stops getting smaller.

```
sd_emission = 5

observations = sample_observations(iterations, sd_emission)

res = particle_filter(observations$obs,
                      T = iterations,
                      M = partice_sample_size,
                      sd_emission = sd_emission,
                      corr=FALSE)
```



4 Source Code

```
library(ggplot2)
knitr::opts_chunk$set(echo = TRUE)
set.seed(12345)

# Parameter setup
partice_sample_size = 100 # M
iterations = 100 # T

rtransition = function(z_t_1, sd=1) {
  return(rnorm(1, sample(c(0,1,2), size=1)+z_t_1, sd=sd))
}

remission = function(n, z_t, sd=1) {
  return(rnorm(1, sample(c(0,1,2), size=1)+z_t, sd=sd))
}

demission = function(x_t, z_t, sd=1) {
  return(sum(dnorm(x_t, c(0, -1, 1)+z_t, sd=sd))/3)
}

rinit = function(n) {
  return(runif(n, 0, 100))
}
```

```r
sample_observations = function(n, sd_emission=1) {
  # First observation
  states = vector(length = n) # Z
  observations = vector(length = n) # X

  states[1] = rinit(1)
  observations[1] = remission(1, states[1], sd_emission)

  for (i in 2:n) {
    states[i] = rtransition(states[i-1])
    observations[i] = remission(1, states[i], sd_emission)
  }

  return(data.frame(states = states, obs = observations))
}

particle_filter = function(observations, T=100, M=100, sd_emission=1, corr=TRUE) {

  X = matrix(nrow = T, ncol = M) # Posterior believe
  X_bar = matrix(nrow = T, ncol = M) # Prior believe
  W = matrix(nrow = T, ncol = M)
  Z = vector(length = T)

  for (t in 1:T) {

    if (t == 1) {
      # Initialization
      X_temp = rinit(M)
      # Prediction
      X_bar[t,] = sapply(X_temp, rtransition)
    }
    else {
      # Prediction
      X_bar[t,] = sapply(X[t-1,], rtransition)
    }

    # Importance Weight
    W[t, ] = sapply(X_bar[t,], demission, z_t = observations[t], sd=sd_emission)
    # Normalize
    W[t, ] = W[t, ]/sum(W[t, ])
    # Correction
    if (corr) {
      prob = W[t,]
    }
    else {
      prob = rep(1, length(W[t,]))
    }
    X[t,] = sample(X_bar[t,], M, prob = prob, replace = TRUE)
    # Taken from Bishop
    Z[t] = as.numeric(W[t, ] %*% X[t, ])

  }
```

```
  return(list(X=X, X_bar=X_bar, W=W, Z=Z))
}



sd_emission = 1

observations = sample_observations(iterations, sd_emission)

res = particle_filter(observations$obs,
                      T = iterations,
                      M = partice_sample_size,
                      sd_emission = sd_emission)


df = data.frame(time = 1:iterations,
                true_states=observations$states,
                observed_states=observations$obs,
                estimated_states=res$Z)

ggplot(df) +
  geom_line(aes(x = time, y = true_states, colour = "True States")) +
  geom_line(aes(x = time, y = observed_states, colour = "Observed States")) +
  geom_line(aes(x = time, y = estimated_states, colour = "Estimated States")) +
  labs(title = "True, Observed and Estimated States (sd=1)",
       y = "Position",
       x = "Time", color = "Legend") +
  scale_color_manual(values = c("#C70039", "#FF5733", "#581845")) +
  theme_minimal()


sd_emission = 5

observations = sample_observations(iterations, sd_emission)

res = particle_filter(observations$obs,
                      T = iterations,
                      M = partice_sample_size,
                      sd_emission = sd_emission)


df = data.frame(time = 1:iterations,
                true_states=observations$states,
                observed_states=observations$obs,
                estimated_states=res$Z)

ggplot(df) +
  geom_line(aes(x = time, y = true_states, colour = "True States")) +
  geom_line(aes(x = time, y = observed_states, colour = "Observed States")) +
  geom_line(aes(x = time, y = estimated_states, colour = "Estimated States")) +
  labs(title = "True, Observed and Estimated States (sd=5)",
       y = "Position",
       x = "Time", color = "Legend") +
```

```r
  scale_color_manual(values = c("#C70039", "#FF5733", "#581845")) +
  theme_minimal()


sd_emission = 50

observations = sample_observations(iterations, sd_emission)

res = particle_filter(observations$obs,
                      T = iterations,
                      M = partice_sample_size,
                      sd_emission = sd_emission)


df = data.frame(time = 1:iterations,
                true_states=observations$states,
                observed_states=observations$obs,
                estimated_states=res$Z)

ggplot(df) +
  geom_line(aes(x = time, y = true_states, colour = "True States")) +
  geom_line(aes(x = time, y = observed_states, colour = "Observed States")) +
  geom_line(aes(x = time, y = estimated_states, colour = "Estimated States")) +
  labs(title = "True, Observed and Estimated States (sd=50)",
       y = "Position",
       x = "Time", color = "Legend") +
  scale_color_manual(values = c("#C70039", "#FF5733", "#581845")) +
  theme_minimal()


sd_emission = 5

observations = sample_observations(iterations, sd_emission)

res = particle_filter(observations$obs,
                      T = iterations,
                      M = partice_sample_size,
                      sd_emission = sd_emission,
                      corr=FALSE)


df = data.frame(time = 1:iterations,
                true_states=observations$states,
                observed_states=observations$obs,
                estimated_states=res$Z)

ggplot(df) +
  geom_line(aes(x = time, y = true_states, colour = "True States")) +
  geom_line(aes(x = time, y = observed_states, colour = "Observed States")) +
  geom_line(aes(x = time, y = estimated_states, colour = "Estimated States")) +
  labs(title = "True, Observed and Estimated States (sd=5)",
       y = "Position",
       x = "Time", color = "Legend") +
```

```
  scale_color_manual(values = c("#C70039", "#FF5733", "#581845")) +
  theme_minimal()
```