TrackingParticles

Luis Damiano

May 10, 2019

1 Learning outcomes

Statistics non-linear state-space models, learning more about Particle Filter in the hands-on way.

Computing GSL (vector, matrix, rng, ran, linalg), numerical errors handling (see filter.c:118), macros for allowing the code to be compiled standalone, linking C to R, Makevars, creating an R Package, writing a vignette with Sweave, good coding practices (documentation, commenting, version control).

2 Introduction

We introduce the bearing-only tracking problem for a moving vehicle. For this R package, which comes with a real-life dataset, we impement in C a Particle Filter to find the solution of the non-linear state-space model.

In Summer 2014, a tractor harvested an experimental agricultural site codenamed Interim, which is located at the Neal Smith National Wildlife Refuge. The vehicle was equipped with a yield monitor, a common device for precision agriculture that records several quantities of interests generated by sensors in real time. We assume the existence of two passive sensors situated at the coordinates 41°33′22.9”N 93°14′58.1”W and 41°33′27.6”N 93°14′51.1”W tracking solely this target. Every second, each sensor would record the horizontal angle in radians in the direction of the moving target and itself. The dataset contains a total of 11027 pairs of observations.

It is of interest to estimate the position of the moving target at each time step. A noiseless approximation could be produced by fixing the velocity equal to zero and estimating the position from the crossing of the measurements. By finding the solution to the following linear system,

$$\begin{pmatrix} \cos a_1 & \cos a_2 \\ \sin a_1 & \sin a_2 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} s_x^2 - s_x^1 \\ s_y^2 - s_y^1 \end{pmatrix}, \tag{1}$$

the position could be recovered as

$$\begin{pmatrix} \hat{x}_k \\ \hat{y}_k \end{pmatrix} = d1 \begin{pmatrix} \cos a_1 \\ \sin a_1 \end{pmatrix}. \tag{2}$$

As we live in a world full of uncertainty (or is it just me?), we employ instead a non-linear state-space model as described in Särkkä (2013). The measurement model for the i-th sensor with $i \in \{1, 2\}$ is given by

$$a_k^i = \tan^{-1}\left(\frac{y_k - s_y^i}{x_k - s_x^i}\right) + r_k, \tag{3}$$

where $\left(s_x^i, s_y^i\right)$ is the position of the sensor and $r_k \sim N(0, \sigma^2)$ is a Gaussian measurement noise with standard deviation $\sigma > 0$ measured in radians.

The state of the vehicle at time step k could be represented by the latent vector $\mathbf{x}_k = (x_k, y_k, \dot{x}_k, \dot{y}_k)^T$, where the elements are the the position and velocity respectively. The state dynamics could be then modeled with the discretized Wiener velocity model as follows:

$$\begin{pmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{pmatrix} + \mathbf{q}_{k-1}. \tag{4}$$

The variable $\mathbf{q}_{k-1}$ is a zero-mean Gaussian process noise with covariance

$$\mathbf{Q} = \begin{pmatrix} q_1^{\mathrm{c}}\Delta t^3/3 & 0 & q_1^{\mathrm{c}}\Delta t^2/2 & 0 \\ 0 & q_2^{\mathrm{c}}\Delta t^3/3 & 0 & q_2^{\mathrm{c}}\Delta t^2/2 \\ q_1^{\mathrm{c}}\Delta t^2/2 & 0 & q_1^{\mathrm{c}}\Delta t & 0 \\ 0 & q_2^{\mathrm{c}}\Delta t^2/2 & 0 & q_2^{\mathrm{c}}\Delta t \end{pmatrix}, \tag{5}$$

where the constant $\Delta t \geq 0$ represents the sampling period measured as times per seconds (e.g. $\Delta t = 0.1$ means ten times per second), and the constants $q_1^c > 0$ and $q_2^c > 0$ are the diffusion coefficients.

3 Filtering

Linear state-space models with Gaussian noise have the popular closed-form solution known as Kalman Filter. In this particular case, we propose a Particle Filter (with a caveat) to estimate the posterior mean of the latent state at each time step given a fixed value for the model parameter vector.

The Sequential Importance Resampling (SIR algorithm), also known as the Particle Filter (PF), is the following.

Step 0. Draw N samples $\mathbf{x}_0^{(i)}$ of the latent state vector from the prior distribution:

$$\mathbf{x}_0^{(i)} \sim p\left(\mathbf{x}_0\right), \quad i = 1, \ldots, N, \tag{6}$$

and set all weights to the constant $w_0^{(i)} = N^{-1} \; \forall \; i$.

For each time step $k = 1, \ldots, T$, do the following.

Step 1. Draw N samples $\mathbf{x}_k^{(i)}$ from the importance distribution:

$$\mathbf{x}_k^{(i)} \sim \pi\left(\mathbf{x}_k | \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k}\right), \quad i = 1, \ldots, N. \tag{7}$$

Step 2. Compute the new weights:

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p\left(\mathbf{y}_k | \mathbf{x}_k^{(i)}\right) p\left(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1}^{(i)}\right)}{\pi\left(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k}\right)}. \tag{8}$$

Step 3. Normalize the weights to sum to unity:

$$\bar{w}_k^{(i)} = \frac{w_k^{(i)}}{\sum_{i=1}^N w_k^{(i)}}. \tag{9}$$

Step 4. Compute the effective number of particles:

$$n_{\text{eff}} \approx \frac{1}{\sum_{i=1}^N \left(\bar{w}_k^{(i)}\right)^2}. \tag{10}$$

Step 5. If the effective number of particles is too low for a given decision rule, resample with repetition the particles $\mathbf{x}_k^{(i)}$ with probability $\bar{w}_k^{(i)}$ and set all weights to the constant $w_k^{(i)} = N^{-1} \; \forall \; i$.

Caveat: the resampling step is not currently implemented – expect particle degeneracy.

4 Instructions

We present a minimal example illustrating how to (1) set up all the known parameters, (2) run the particle filter, and (3) plot the results. The object returned by the `particle_filter` function is a named list with four elements:

noiseless is a T x 2 matrix with the noiseless approximation of the vehicle position (assumes no noise and velocity equal to zero).

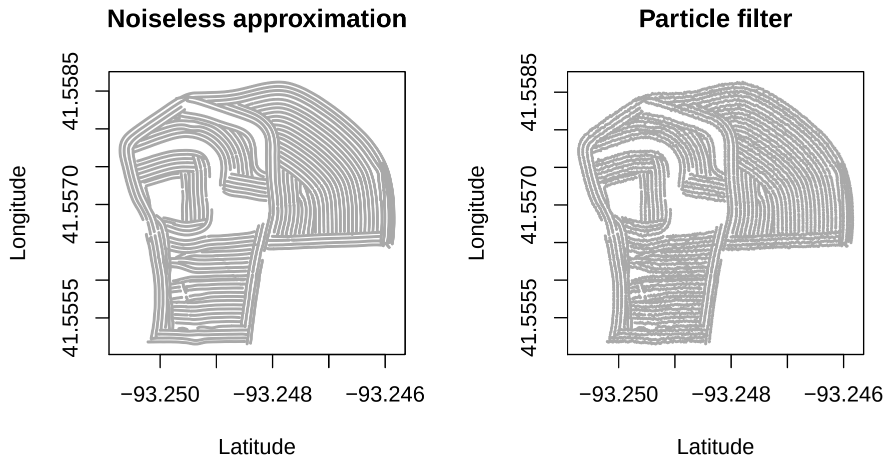
stateMean is a T x 4 matrix with the posterior mean of the latent state at each time step.

weights is a T x nParticles matrix with the normalized weights.

ess is a T-sized vector with the effective sample size at each time step.

For more information, we encourage the reader to read the package manual: from R, simply run `help(package=TrackingParticles)`, or call `?particle_filter`, `?vehicle`, and `?plot.filtered`.

```
library(TrackingParticles)

head(vehicle)

##          a1        a2
## 1 -1.660934 -2.375897
## 2 -1.668618 -2.378755
## 3 -1.674765 -2.380928
## 4 -1.679765 -2.382801
## 5 -1.683469 -2.384178
## 6 -1.683469 -2.384178

# Set up model constants and known values ---------------------------------
nParticles <- 100

# Time step in seconds
dt <- 1

# Location of sensors 1 and 2
s1 <- c(x = -93.2494663765932, y = 41.5563518606521)
s2 <- c(x = -93.2475338232000, y = 41.5576632356000)

# Measurement model parameters
sr <- 0.01

# State model parameters
q1 <- 0.0005
q2 <- 0.0005

# State model priors
statepriorMu   <- c(-93.24952047, 41.55575337)
statepriorDiag <- c(5.0E-09, 3.5E-08, 5.0E-04, 5.0E-04)

# Importance distribution parameters
# If you ever wonder how I found 0.0025, let's just call it heuristics :D
importanceDiag <- 0.0025 * c(5.00E-10, 1.75E-08, 5.00E-05, 5.00E-05)

# Run ---------------------------------------------------------------------
res <- particle_filter(
  vehicle,
  dt,
  s1, s2,
  sr,
  q1, q2,
  statepriorMu, statepriorDiag,
  importanceDiag,
  nParticles
)

print(str(res))

## List of 4
##  $ noiseless: num [1:11027, 1:2] -93.2 -93.2 -93.2 -93.2 -93.2 ...
##  $ stateMean: num [1:11027, 1:4] -93.2 -93.2 -93.2 -93.2 -93.2 ...
##  $ weights  : num [1:11027, 1:100] 2.09e-07 3.43e-10 2.72e-11 9.70e-09 9.76e-10 ...
##  $ ess      : num [1:11027] 1.1 1.31 2.24 1.86 1.09 ...
##  - attr(*, "class")= chr "filtered"
## NULL

# Package comes with a basic visualization routine ------------------------
plot(res, pch  = 16, col = "darkgray", cex = 0.3)
```



```r
# Finally, the functions are fully documented.
help(package = TrackingParticles)
```

5 References

Simo Särkkä. 2013. "Bayesian Filtering and Smoothing". *Cambridge University Press*. Read online.