Sensors networks

Non-linear filtering

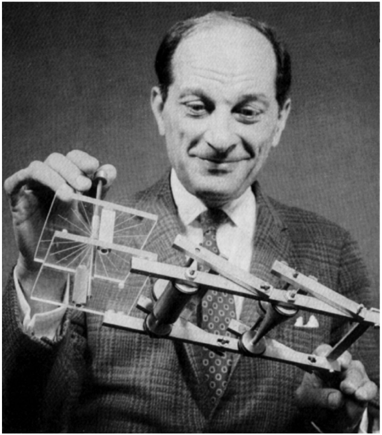
Manuel A. Vázquez
Joaquín Míguez
Jose Miguel Leiva

February 4, 2024

Index

Index

Linearity



"Using a term like nonlinear science is like referring to the bulk of zoology as the study of non-elephant animals"

— Stanislaw Ulam

Index

Non-linear dynamic model

We consider the same state equation as before

- $$\mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{v}_t,$$

...but now the connection between the state and the observations is given by the (vector) function $\mathbf{h} : \mathbb{R}^M \rightarrow \mathbb{R}^N$ (plus additive Gaussian noise like before)

- $$\mathbf{y}_t = \mathbf{h}\left(\mathbf{x}_t\right) + \mathbf{w}_t,$$

with $\mathbf{h}$ being a vector of scalar functions of a vector

$$\mathbf{h}\left(\mathbf{x}_t\right) = \begin{bmatrix} \mathrm{h}_1\left(\mathbf{x}_t\right) \\ \mathrm{h}_2\left(\mathbf{x}_t\right) \\ \vdots \\ \mathrm{h}_N\left(\mathbf{x}_t\right) \end{bmatrix}$$

We **cannot** apply the Kalman filter!!

Linearized dynamic model

Goal

To apply the KF over the non-linear model to estimate $\mathbf{x}_t$ given $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_t$

We can build a linear approximation to the observation equation[1] using a *first-order* Taylor series,

$$\mathbf{h}(\mathbf{x}_t) \approx \mathbf{h}(\mathbf{x}^0) + \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}_t}\right]_{\mathbf{x}_t = \mathbf{x}^0} (\mathbf{x}_t - \mathbf{x}^0),$$

where

$$\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}_t}\right] = \begin{bmatrix} \frac{\partial \mathrm{h}_1}{\partial x_{1,t}} & \frac{\partial \mathrm{h}_1}{\partial x_{2,t}} & \cdots & \frac{\partial \mathrm{h}_1}{\partial x_{M,t}} \\ \frac{\partial \mathrm{h}_2}{\partial x_{1,t}} & \frac{\partial \mathrm{h}_2}{\partial x_{2,t}} & \cdots & \frac{\partial \mathrm{h}_2}{\partial x_{M,t}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathrm{h}_N}{\partial x_{1,t}} & \frac{\partial \mathrm{h}_N}{\partial x_{2,t}} & \cdots & \frac{\partial \mathrm{h}_N}{\partial x_{M,t}} \end{bmatrix}$$

is the Jacobian matrix (of partial derivatives) of $\mathbf{h}$.

[1] We could do the same thing to deal with a non-linear state equation!!

Deriving the extended Kalman filter

EKF defines the *corrected* observations,

$$\tilde{\mathbf{y}}_t = \mathbf{y}_t - \mathbf{h}\left(\mathbf{x}^0\right) + \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}_t}\right]_{\mathbf{x}_t = \mathbf{x}^0} \mathbf{x}^0,$$

which yield an approximate dynamic model which is both linear and Gaussian

$$\begin{aligned} \mathbf{x}_t &= \mathbf{F}\mathbf{x}_{t-1} + \mathbf{v}_t \\ \tilde{\mathbf{y}}_t &= \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}_t}\right]_{\mathbf{x}_t = \mathbf{x}^0} \mathbf{x}_t + \mathbf{w}_t \end{aligned}$$

Success!!

It is straightforward to apply the KF on the previous model.

Extended Kalman Filter

- Prediction

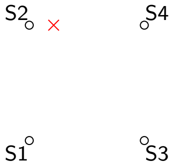
$$\begin{aligned} \hat{\mathbf{x}}_{t|t-1} &= \mathbf{F}\hat{\mathbf{x}}_{t-1|t-1} \\ \mathbf{P}_{t|t-1} &= \mathbf{Q} + \mathbf{F}\mathbf{P}_{t-1|t-1}\mathbf{F}^\top \end{aligned}$$

- Update

$$\begin{aligned} \mathbf{K}_t &= \mathbf{P}_{t|t-1}\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}_t}\right]^\top_{\mathbf{x}_t=\mathbf{x}^0}\left(\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}_t}\right]_{\mathbf{x}_t=\mathbf{x}^0}\mathbf{P}_{t|t-1}\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}_t}\right]^\top_{\mathbf{x}_t=\mathbf{x}^0} + \mathbf{R}\right)^{-1} \\ \hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{h}(\hat{\mathbf{x}}_{t|t-1})) \\ \mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} - \mathbf{K}_t\left(\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}_t}\right]_{\mathbf{x}_t=\mathbf{x}^0}\mathbf{P}_{t|t-1}\left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}_t}\right]^\top_{\mathbf{x}_t=\mathbf{x}^0}\right)\mathbf{K}_n^\top, \end{aligned}$$

  where $\mathbf{Q}$ is the covariance matrix of $\mathbf{v}_t$, and $\mathbf{R}$ that of $\mathbf{w}_t$.
- The linearization point $\mathbf{x}^0$ must be close enough to $\mathbf{x}_t$ for the algorithm to work properly. Usually, we take $\mathbf{x}^0 = \hat{\mathbf{x}}_{t|t-1}$.

Example of non-linear function for tracking



Goal

Localization and tracking of an object moving with a known constant velocity $\mathbf{c}$.

Same state equation as before,

- $$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{c}T + \mathbf{v}_t,$$

...a more *realistic* observation equation based on the Received Signal Strength Indicator (RSSI),

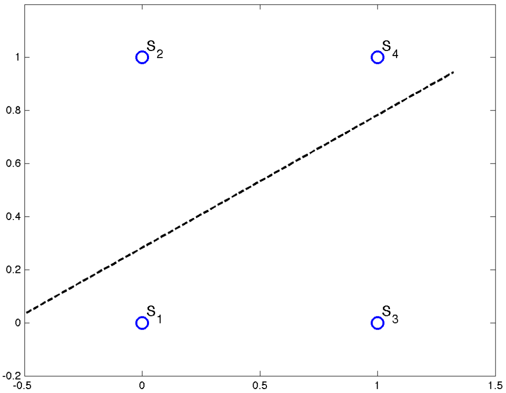
- $$y_{t,i} = \underbrace{k_1 - k_2 \log \|\mathbf{x}_t - \mathbf{s}_i\|}_{\mathrm{RSSI}_i} + w_{t,i}, \quad i = 1, \cdots, N$$

  with k_1 and k_2 being some known constants and $\mathbf{s}_i$ the position of the corresponding sensor.

  (previously, $\mathbf{y}_t = \mathbf{x}_t + \mathbf{w}_t$)

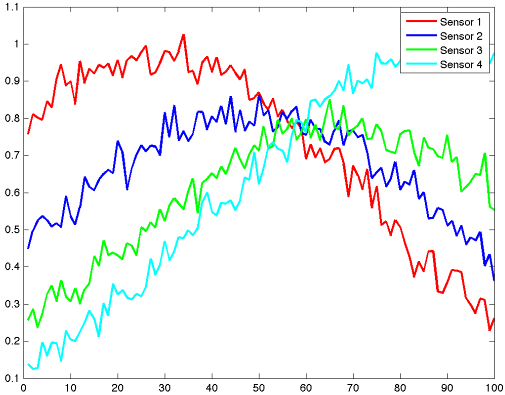
Example

- True trajectory



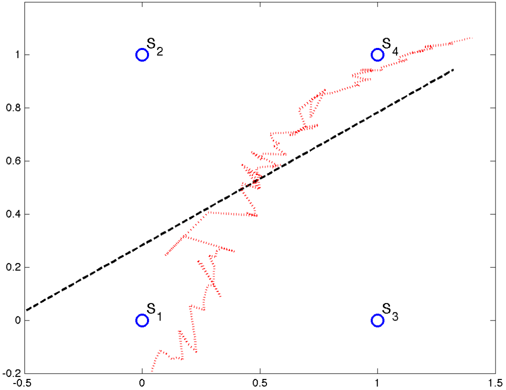
Example

- Sensors readings



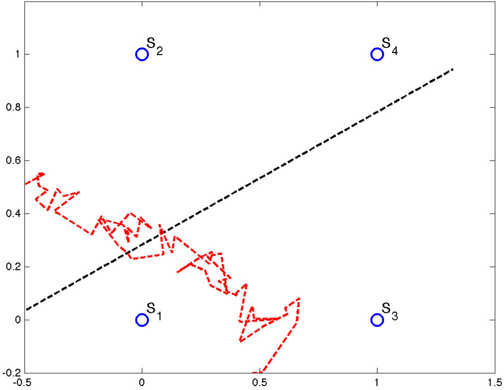
Example

- Result when filtering using only Sensor 2



Example

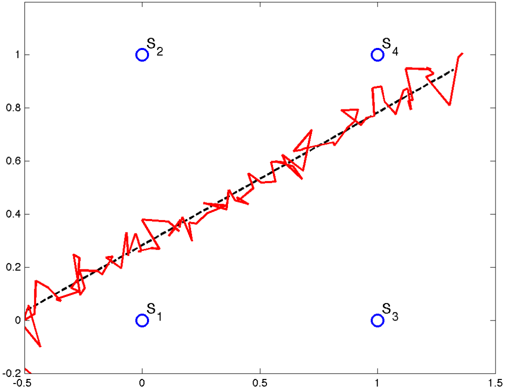
- Result when filtering using Sensors 1 and 2



- Sensors cannot disambiguate the direction.

Example

- Result when filtering using the four sensors



Index

Unscented Kalman filter

- Another non-linear extension of the Kalman filter (alternative to EKF)...
- The model:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{f}\left(\mathbf{x}_{t-1}, \mathbf{v}_t\right), \quad \mathbf{v}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_n\right) \\ \mathbf{y}_t &= \mathbf{H}_t \mathbf{x}_t + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}_n\right) \end{aligned}$$

  The observation equation is linear...but the state equation is **not** (**f** is any arbitrary vector function)
- It relies on the...

unscented transformation

a method for computing the moments of a *Gaussian* random variable that undergoes a nonlinear transformation.

...which in turn makes use of a...

Sigma point representation

Let us consider $p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{x}_t|\hat{\mathbf{x}}_t, \mathbf{P}_t)$. We can represent this distribution using a collection of (deterministic) **sigma points**

$$\begin{aligned} \mathbf{X}_t(0) &= \hat{\mathbf{x}}_t, & \mathrm{W}_t(0) &= \kappa/(M+\kappa) \\ \mathbf{X}_t(i) &= \hat{\mathbf{x}}_t + \left(\sqrt{(M+\kappa)P_t}\right)_i, & \mathrm{W}_t(i) &= 1/\left(2(M+\kappa)\right) \\ \mathbf{X}_t(i+M) &= \hat{\mathbf{x}}_t - \left(\sqrt{(M+\kappa)P_t}\right)_i, & \mathrm{W}_t(i+M) &= 1/\left(2(M+\kappa)\right) \end{aligned}$$

for $i = 1, \cdots, M$, where $\kappa \in \mathbb{R}$ and $\left(\sqrt{(M+\kappa)P_t}\right)_i$ is the i-th column of the matrix square root of $(M+\kappa)\mathbf{P}_t$.

Theorem: Sigma points

This set of weighted samples has the same sample mean and covariance as the original distribution.

Steps in the unscented Kalman filter

Once the sigma points are computed, the **prediction step** at time $t+1$ can be carried out as follows:

1. Propagate each sigma point through the non-linearity $\mathbf{f}$

$$\mathbf{X}_{t+1|t}(i) = \mathbf{f}(\mathbf{X}_t(i), 0).$$

2. Compute the predicted mean

$$\hat{\mathbf{x}}_{t+1}^{-} = \sum_{i=0}^{2M} \mathrm{W}_t(i)\mathbf{X}_{t+1|t}(i).$$

3. Compute the predictive covariance

$$\mathbf{P}_{t+1}^{-} = \sum_{i=0}^{2M} \mathrm{W}_t(i) \left(\mathbf{X}_{t+1|t}(i) - \hat{\mathbf{x}}_{t+1}^{-}\right) \left(\mathbf{X}_{t+1|t}(i) - \hat{\mathbf{x}}_{t+1}^{-}\right)^{\top}$$

The *update step* is carried out as in the standard KF.

Remarks

- The mean vector and covariance matrix computed by propagating the sigma points through the nonlinearity are still **estimates**, but more accurate than those produced by the EKF. They are correct up to the 2nd order of a Taylor expansion. ✓
- Approximations are still Gaussian, i.e., the method is not suitable when multimodal posterior distributions are expected. ✗
- The UKF can be used without computing derivatives. A linearization of the model is implicit, though (i.e., the UKF can be re-written as a linearization method). ✓
- Different choices of sigma points are possible. If a Gauss-Hermite quadrature rule is used, a larger number of points is needed but the approximations are more accurate as well.
- UKF algorithms look simple to implement. However performance may actually vary depending, e.g., on the number of points.

Index

State space models

- Formal statement of the estimation problem...
- ...in a Bayesian framework.
- Non-linear state space model

$$\left\{ \begin{array}{c} \mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{v}_t) \\ \mathbf{y}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{w}_t) \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{c} \mathbf{x}_0 \sim p(\mathbf{x}_0) \\ \mathbf{x}_t \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}) \\ \mathbf{y}_t \sim p(\mathbf{y}_t | \mathbf{x}_t) \end{array} \right\}$$

where

- $\mathbf{f}, \mathbf{h} \equiv$ state and observation functions;
- $\mathbf{v}_t, \mathbf{w}_t \equiv$ state and observation noise;
- $p(\mathbf{x}_0) \equiv$ prior pdf of the state;
- $p(\mathbf{x}_t|\mathbf{x}_{t-1}) \equiv$ transition pdf of the state;
- $p(\mathbf{y}_t|\mathbf{x}_t) \equiv$ conditional pdf of the observation (likelihood of the state).

Stochastic filtering

Goal

Tracking the posterior distribution, $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, which allows computing the expectation of any function of interest, $\mathbf{g}$, as

$$\mathbb{E}\left[\mathbf{g}(\mathbf{x}_t)\right] = \int \mathbf{g}(\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t}) d\mathbf{x}_t$$

Using Bayes theorem, one can easily show

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}$$

Stochastic filtering

There is uncertainty in the observations and/or the noise governing the evolution of the system...that's why we talk about **stochastic filtering**[a].

[a] Kalman filter also falls within this category!!

Index

Monte Carlo integration

Let X be a r.v. with pdf $p(x)$ and consider the problem of approximating

$$\mathbb{E}\left[h(X)\right] = \int h(x)p(x)dx$$

for some integrable function h.

One possible approach

If we can **draw** N **i.i.d. samples** $x^{(1)}, ..., x^{(N)}$ from $p(x)$ and the variance of the r.v. $Y = h(X)$ is finite, then

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} h(X^{(n)}) = \mathbb{E}\left[h(X)\right]$$

almost surely (a.s.).

Sampling

Unfortunately, in many problems it is impossible to draw samples from $p(x)$...

Example

$$\mathbf{y}_t = \mathbf{H}^H \mathbf{x}_t + \mathbf{w}_t$$

We want to estimate $\mathbf{x}_t$ from $\mathbf{y}_t$, i.e., we aim at approximating $p(\mathbf{x}_t \mid \mathbf{y}_t)$...but we cannot sample directly from the latter (how??)

...but maybe $p(x)$ can be evaluated *up to a proportionality constant*[2]:

$$p(\mathbf{x}_t \mid \mathbf{y}_t) = \frac{p(\mathbf{y}_t \mid \mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{y}_t)} \propto p(\mathbf{y}_t \mid \mathbf{x}_t)p(\mathbf{x}_t)$$

[2] Say $p(x) = Kf(x)$ where function $f(x)$ is known, but constant K is not.

Index

Importance sampling

Assume the pdf of interest, $p(x)$, (the **target** pdf) can be evaluated up to a proportionality constant and

- *choose* a pdf, $q(x)$, known as **proposal function** such that

$$p(x) > 0 \Rightarrow q(x) > 0$$

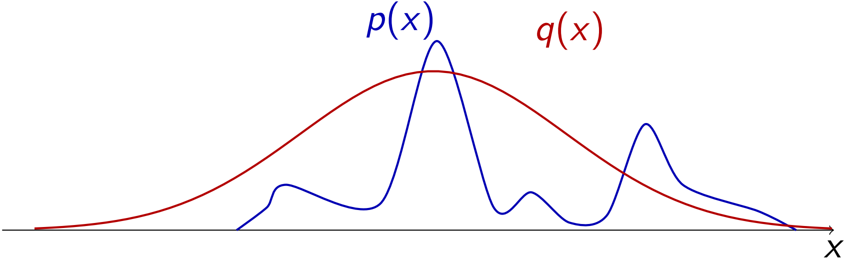
- define the **weight function** as

$$w(x) = c\frac{p(x)}{q(x)}$$

where c is an arbitrary (possibly unknown) constant

then we can compute the expectation of any arbitrary function $h(x)$ with respect to $p(x)$...but using samples from $q(x)$!!

Importance sampling: $q(x)$



Constraint

The support of $q(x)$ must encompass that of $p(x)$,

$$p(x) > 0 \Rightarrow q(x) > 0$$

How to choose it

For the sake of efficiency, the proposal pdf should be as close as possible to the target pdf.

Importance sampling: procedure

...to approximate $\mathbb{E}\left[h(X)\right]$ with respect to $p(x)$ using samples from $q(x)$

1. Draw $\mathbf{x}^{(i)} \sim q(x)$ for $i = 1, \cdots, N$
2. Compute

$$w(\mathbf{x}^{(i)}) = c\frac{p(\mathbf{x}^{(i)})}{q(\mathbf{x}^{(i)})} \triangleq w^{*(i)} \text{ (\textit{unnormalized} weight)}$$

   for $i = 1, \cdots, N$
3. Normalize the weights as

$$w^{(i)} = \frac{w^{*(i)}}{\sum_{j=1}^{N} w^{*(j)}}$$

4. Approximate $\mathbb{E}\left[h(X)\right]$ as

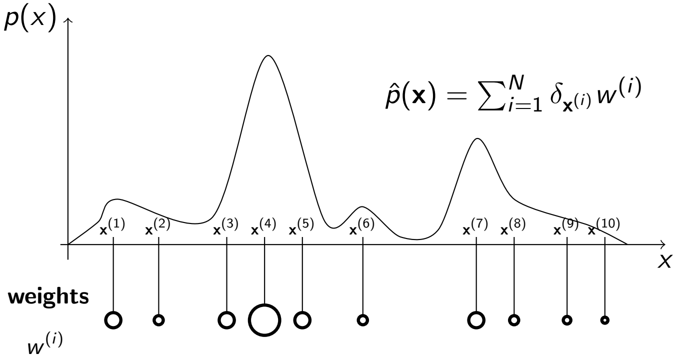
$$\mathbb{E}\left[h(X)\right] \approx \sum_{i=1}^{N} w^{(i)} h(\mathbf{x}^{(i)}) \tag{1}$$

Importance sampling: interpretation

Using IS, we end up with a collection of pairs (`sample`,`weight`):

$$\left\{ \left(\mathbf{x}^{(1)}, w^{(1)}\right), \left(\mathbf{x}^{(2)}, w^{(2)}\right), \left(\mathbf{x}^{(3)}, w^{(3)}\right), \cdots \right\}$$

The weight can be *interpreted* as the probability of the corresponding sample



Importance sampling in dynamic systems

Recursive IS

Can we apply IS to **recursively** estimate the state in a dynamic system?

Let us consider a dynamic model in state-space form specified by

$$\mathbf{x}_0 \sim p(\mathbf{x}_0), \quad \mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad \mathbf{y}_t \sim p(\mathbf{y}_t|\mathbf{x}_t)$$

We already know how to approximate **any** distribution of interest, and hence we could approximate

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}), p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:t+1}), p(\mathbf{x}_{t+2} \mid \mathbf{y}_{1:t+2}), \cdots$$

one after the other, but they are related...

Goal

Build (using importance sampling) an approximation of $p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:t+1})$ using one from $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$.

Index

Importance sampling in dynamic systems

Assume we have an approximation of $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ given by

$$\hat{p}^N(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) = \sum_{i=1}^{N} \delta_{\mathbf{x}_{t-1}^{(i)}} w_{t-1}^{(i)}$$

$$\begin{aligned}
p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) &= \frac{p(\mathbf{y}_t \mid \mathbf{x}_t, \mathbf{y}_{1:t-1}) p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})}{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})} \\
&\propto p(\mathbf{y}_t \mid \mathbf{x}_t, \mathbf{y}_{1:t-1}) p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) \\
&= p(\mathbf{y}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} \\
&\approx p(\mathbf{y}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) \hat{p}^N(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} \\
&= p(\mathbf{y}_t \mid \mathbf{x}_t) \sum_{i=1}^{N} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{1:t-1}) w_{t-1}^{(i)}
\end{aligned}$$

Particle filtering

- **Initialization**
  - samples are drawn from the prior,
  $$\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0),\ i = 1, \cdots, N,$$
  - all the weights are set to the same value
  $$w_i^{(0)} = 1/N, i = 1, \cdots, N$$
- **Recursion** at time t
  - draw samples, $\mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)}, \cdots$, from the *selected* proposal,
  $$\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t \mid \mathbf{y}_{1:t})$$
  - compute the weights
  $$w_t^{(i)} \propto \frac{p(\mathbf{x}_t^{(i)} \mid \mathbf{y}_{1:t})}{q(\mathbf{x}_t^{(i)} \mid \mathbf{y}_{1:t})} = \frac{p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) \sum_{i=1}^N p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{1:t-1}) w_{t-1}^{(i)}}{q(\mathbf{x}_t^{(i)} \mid \mathbf{y}_{1:t})}$$

This scheme is called **particle filtering** or *Sequential Importance Sampling* (SIS). Once samples are available, Equation (1) can be used to approximate any integral with respect to $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$.

Bootstrap filter

If we choose as proposal function

$$q(\mathbf{x}_t \mid \mathbf{y}_{1:t}) = \sum_{i=1}^{N} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{1:t-1}) w_{t-1}^{(i)}$$

computing the weights is easy

$$w_t^{(i)} \propto \frac{p(\mathbf{x}_t^{(i)} \mid \mathbf{y}_{1:t})}{q(\mathbf{x}_t^{(i)} \mid \mathbf{y}_{1:t})} = \frac{p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) \cancel{\sum_{i=1}^{N} p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{1:t-1}) w_{t-1}^{(i)}}}{\cancel{\sum_{i=1}^{N} p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{1:t-1}) w_{t-1}^{(i)}}}$$
$$= p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$$

The resulting algorithm is the **bootstrap filter**, considered the first particle filter

Bootstrap filter: the proposal function

Drawing samples from the proposal

$$q(\mathbf{x}_t \mid \mathbf{y}_{1:t}) = \sum_{i=1}^{N} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{1:t-1}) w_{t-1}^{(i)}$$

can be seen as a two step procedure:

- **resampling** the previous approximation,

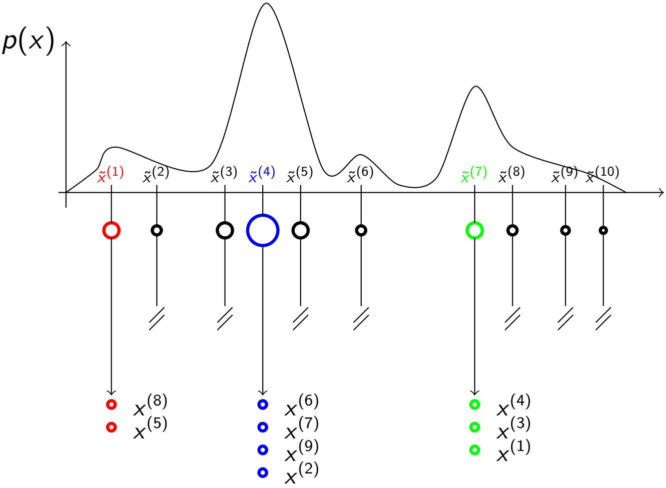
$$\left\{ \left(\mathbf{x}_{t-1}^{(1)}, w_{t-1}^{(1)} \right), \left(\mathbf{x}_{t-1}^{(2)}, w_{t-1}^{(2)} \right), \left(\mathbf{x}_{t-1}^{(3)}, w_{t-1}^{(3)} \right), \cdots \right\}$$

  to get $\mathbf{x}_{t-1}^{(j_1)}, \mathbf{x}_{t-1}^{(j_2)}, \cdots, \mathbf{x}_{t-1}^{(j_t)}$ with $j_1, j_2, \cdots, j_t \in \{1, \cdots, N\}$

- **propagating** each resampled particle using the transition pdf, $p(\mathbf{x}_t | \mathbf{x}_{t-1})$, as

$$x_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(j_i)}), i = 1, \cdots, N$$

Resampling



Bootstrap filter: implementation

- **Initialization**
  - sample $\mathbf{x}_0^{(i)}, i = 1, \cdots, N$ from the prior $p(\mathbf{x}_0)$
- **Recursion** given $\hat{p}^N(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) = \sum_{i=1}^N w^{(i)} \delta_{\tilde{\mathbf{x}}_{t-1}^{(i)}}$,

$\hat{p}^N(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) = \frac{1}{N} \sum_{i=1}^N \delta_{\mathbf{x}_{t-1}^{(i)}}$,

1. propagation (sampling)

$$\tilde{\mathbf{x}}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i)}), i = 1, \cdots, N$$

2. weight computation...

$$w^{*(i)} = p(\mathbf{y}_t \mid \tilde{\mathbf{x}}_t^{(i)}), i = 1, \cdots, N$$

...and normalization

$$w^{(i)} = \frac{w^{*(i)}}{\sum_{j=1}^N w^{*(j)}}, i = 1, \cdots, N$$

3. resampling: let $\mathbf{x}_t^{(i)} = \tilde{\mathbf{x}}_t^{(j)}$ with probability $w^{(j)}, i = 1, \cdots, N, j \in \{1, \cdots, N\}$.

Bootstrap filter: overview

1. **Initialization**

$$\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0) \text{ for } i = 1, \cdots, N$$

2. **Recursive step:** starting from samples at time instant $t-1$

2.1. **Samples propagation**

$$\tilde{\mathbf{x}}_t^{(i)} \sim p\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}\right)$$

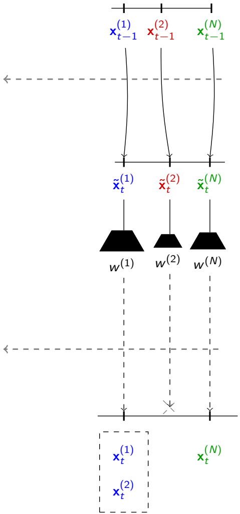
2.2. **Weights computation and normalization**

$$w^{(i)} \propto p\left(\mathbf{y}_t \mid \tilde{\mathbf{x}}_t^{(i)}\right), i = 1, \cdots, N$$

2.3. **Resampling**

$$\mathbf{x}_t^{(i)} = \tilde{\mathbf{x}}_t^{(j)}, i = 1, \cdots, N$$
with probability $w^{(j)}, j \in \{1, \cdots, N\}$

samples at time t



Bootstrap filter: epilogue

In the above implementation, at the end of every iteration we have samples

$$\mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)}, \cdots \mathbf{x}_t^{(N)}$$

that make up an approximation of

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}),$$

but the initial goal was to approximate the expectation of some (known) function of interest, $\mathbf{g}$, with respect to $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$, i.e.,

$$\mathbb{E}\left[\mathbf{g}(\mathbf{x}_t)\right] = \int \mathbf{g}(\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t}) d\mathbf{x}_t.$$

We simply use the samples to compute a Monte Carlo approximation,

$$\mathbb{E}\left[\mathbf{g}(\mathbf{x}_t)\right] \approx \frac{1}{N} \sum_{n=1}^{N} \mathbf{g}(\mathbf{x}_t^{(n)})$$