Monte Carlo Filtering Using Kernel Embedding of Distributions

Motonobu Kanagawa
The Institute of Statistical Mathematics, Tokyo

Gatsby Unit External Seminar: 19 Nov. 2014
(Joint work with Yu Nishiyama, Arthur Gretton, and Kenji Fukumizu)
Introduction and Problem Setup
Filtering with a state-space model

We deal with time-series data consisting of

- **state:**
  \[ x_1, x_2, \ldots, x_t, \ldots \in \mathcal{X} \]

- **observation:**
  \[ y_1, y_2, \ldots, y_t, \ldots \in \mathcal{Y} \]

State-space model: these data are generated by

- **transition model** \( p(x_t|x_{t-1}) \):
  \[ x_t \sim p(x_t|x_{t-1}) \]

- **observation model** \( p(y_t|x_t) \):
  \[ y_t \sim p(y_t|x_t) \]
Filtering with a state-space model

**Filtering:** For each time $t$, we observe $y_t$. Then estimate the posterior distribution on state $x_t$, given a history $y_{1:t} := y_1, \ldots, y_t$:

$$p(x_t | y_{1:t}).$$

Make use of Bayes’ rule:

$$p(x_t | y_{1:t}) \propto \underbrace{p(y_t | x_t)p(x_t | y_{1:t-1})}_{\text{likelihood prior}}$$

$$= p(y_t | x_t) \int p(x_t | x_{t-1}) \underbrace{p(x_{t-1} | y_{1:t-1})}_{\text{posterior at } t-1} dx_{t-1}.$$ 

Posterior estimation is to be done recursively.
Basics of filtering algorithms

Suppose \( p(x_{t-1}|y_{1:t-1}) \) was already estimated.

At time \( t \), decompose posterior estimation into two steps:

1. **Prediction**: Estimate the prior

\[
p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1}) \ p(x_{t-1}|y_{1:t-1}) \, dx_{t-1}.
\]

2. **Correction**: Given \( y_t \), estimate the posterior with Bayes’ rule

\[
p(x_t|y_{1:t}) \propto p(y_t|x_t) \ p(x_t|y_{1:t-1}) \ p(x_t|x_{t-1})
\]

\( p(x_t|x_{t-1}) \) and \( p(y_t|x_t) \) are assumed to be known.
Existing filtering methods

Kalman filters:
- $p(x_t|x_{t-1})$ and $p(y_t|x_t)$ are assumed to be linear-Gaussian.
- Nonlinear approximation: extended/unscented kalman filters.

Particle filters (Doucet et al., 2001):
- $p(x_t|x_{t-1})$ and $p(y_t|x_t)$ can be nonlinear-nonGaussian.
- Posterior is estimated as a weighted empirical distribution:
  \[
  \hat{p}(x_t|y_{1:t}) = \sum_{i=1}^{n} w_{t,i} \delta x_{t,i},
  \]
- $w_{t,1}, \ldots, w_{t,n} \geq 0$ are importance weights.
- $X_{t,1}, \ldots, X_{t,n} \in \mathcal{X}$ are called particles.
Particle filters (PF) (Doucet et al., 2001)

Suppose $p(x_{t-1}|y_{1:t-1})$ was already estimated as

$$
\hat{p}(x_{t-1}|y_{1:t-1}) = \sum_{i=1}^{n} w_{t-1,i} \delta x_{t-1,i}
$$

At time $t$, PF estimates $p(x_t|y_{1:t})$ by the following steps:

1. **Prediction step**: estimate $p(x_t|y_{1:t-1})$ by sampling:

   $$
   X_{t,i} \sim p(x_t|x_{t-1} = X_{t-1,i}), \quad (i = 1, \ldots, n),
   $$

   $$
   \hat{p}(x_t|y_{1:t-1}) := \sum_{i=1}^{n} w_{t-1,i} \delta x_{t,i}.
   $$
2. **Correction step**: given $y_t$, estimate $p(x_t|y_{1:t})$ by importance weighting:

$$w_{t,i} \propto p(y_t|x_t = X_{t,i}) w_{t-1,i}, \quad (i = 1, \ldots, n)$$

$$\hat{p}(x_t|y_{1:t}) = \sum_{i=1}^{n} w_{t,i} \delta x_{t,i}.$$ 

3. **Resampling step**: resample particles from $\hat{p}(x_t|y_{1:t})$ to collapse small weight particles.
Limitation of particle filters

PF requires evaluation of the observation model $p(y_t|x_t)$:

$$w_{t,i} \propto p(y_t|x_t = X_{t,i}) w_{t-1,i}. \underbrace{\text{observation model}}$$

But this may not be possible if

- $p(y_t|x_t)$ is **unknown**: e.g. robot localization (Vlassis et al., 2002), brain computer interface (Wang et al., 2011).

- $p(y_t|x_t)$ is **intractable**: e.g. econometrics (Jasra et al., 2012).
Example 1: vision-based mobile robot localization

**Problem:** estimate a mobile robot’s position $x_t$ for each time $t$, only given its vision images $y_1, \ldots, y_t$.

**Figure:** COLD database (Pronobis and Caputo, 2009)
Example 1: vision-based mobile robot localization

State-space formulation:

- **state** $x_t$: position (2/3-dim location + angle).
- **observation** $y_t$: vision image (high-dimensional).
- Problem reduces to filtering: estimation of $p(x_t|y_{1:t})$.

Observation model $p(y_t|x_t)$: conditional distribution on vision-images given a position:

- unknown: parametric models cannot be easily defined.
- But, training samples $\{(X_i, Y_i)\}_{i=1}^n$ are available (collected in training phase).

Transition model $p(x_t|x_{t-1})$: robot motion model; many available models (Thrun et al., 2005).
Example 2: brain-computer interface

**Problem:** decoding subject’s finger flexion $x_t$ from ECoG signals $y_1, \ldots, y_t$.

*Figure:* Experimental setup (Wang et al., 2011)
Example 2: brain-computer interface

State-space modeling is possible for BCI tasks (Pistohl et al., 2008; Wang et al., 2011):

- **state** $x_t$: finger positions (5 dim)
- **observation** $y_t$: ECoG signals (64 dim)
- Decoding can be formulated as filtering: $p(x_t|y_{1:t})$

Observation model $p(y_t|x_t)$: conditional distribution on ECoG signals given finger positions:

- **unknown**: parametric models cannot be easily defined.
- But, training samples $\{(X_i, Y_i)\}_{i=1}^n$ are available (collected in training phase).

Transition model $p(x_t|x_{t-1})$: can be modeled using prior knowledge (Wang et al., 2011)
Problem setting

Filtering under the following assumptions:

- transition model \( p(x_t|x_{t-1}) \): known (same as PF).
- observation model \( p(y_t|x_t) \): unknown. But training samples are given:
  \[(X_1, Y_1), \ldots, (X_n, Y_n) \subset \mathcal{X} \times \mathcal{Y}.\]

We develop a filter based on **kernel embedding of distributions**.
Preliminaries: Kernel Embedding of Distributions
Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel.

e.g. for $\mathcal{X} = \mathbb{R}^d$

- Gaussian kernel: $k(x, x') = \exp(-\|x - x'|^2/\gamma^2)$.
- Polynomial kernel: $k(x, x') = (\langle x, x' \rangle + c)^m$.

$\mathcal{X}$ can be a set of structured data, such as image, text and graph, once an appropriate kernel is defined (Schölkopf and Smola, 2002).
Reproducing Kernel Hilbert space (RKHS)

Any positive definite kernel $k$ defines an RKHS $\mathcal{H}$, such that

- For all $x \in \mathcal{X}$,
  \[
  k(\cdot, x) \in \mathcal{H}.
  \]

- For all $f \in \mathcal{H}$ and $x \in \mathcal{X}$ (reproducing property),
  \[
  f(x) = \langle f, k(\cdot, x) \rangle_{\mathcal{H}}.
  \]

Function $k(\cdot, x) \in \mathcal{H}$ is called the feature of $x$.

- Kernel is the inner-product between the features:
  \[
  k(x, x') = \langle k(\cdot, x), k(\cdot, x') \rangle_{\mathcal{H}}, \quad x, x' \in \mathcal{X}.
  \]
Represent any probability distribution $P$ on $\mathcal{X}$ as the mean of features:

$$m_P := \int k(\cdot, x) dP(x) \in \mathcal{H},$$

which will be referred to as the kernel mean of $P$.

- If $k$ is characteristic, e.g. the Gaussian kernel on $\mathbb{R}^d$, $m_P$ uniquely identifies $P$ (Fukumizu et al., 2008):

  $$m_P = m_Q \Rightarrow P = Q$$

- Estimation of $P$ can be cast as estimation of $m_P$. 
Empirical estimate

Given i.i.d. sample \( X_1, \ldots, X_n \) from \( P \), one can estimate \( m_P \) by

\[
\hat{m}_P := \frac{1}{n} \sum_{i=1}^{n} k(\cdot, X_i).
\]

Convergence rate: \( \| \hat{m}_P - m_P \|_{\mathcal{H}} = O_P(n^{-1/2}) \) (Smola et al., 2007).

Other popular estimators:

- Conditional mean embedding (Song et al., 2009).
- Kernel Bayes’ rule (Fukumizu et al., 2011).

These provide estimates in the form of a weighted sum

\[
\hat{m}_P = \sum_{i=1}^{n} w_i k(\cdot, X_i)
\]

with some \( w_1, \ldots, w_n \in \mathbb{R} \) and \( X_1, \ldots, X_n \in \mathcal{X} \).
Proposed Method
Filtering by kernel embedding

Filtering can be cast as estimation of \textit{posterior kernel mean}:

\[ m_{x_t|y_{1:t}} := \int k_{\mathcal{X}}(\cdot, x_t) p(x_t|y_{1:t}) dx_t. \]

This is to be done under the following assumptions:

- transition model \( p(x_t|x_{t-1}) \): known (same as PF).
- observation model \( p(y_t|x_t) \): \textit{unknown}. But training samples are given:
  \[ (X_1, Y_1), \ldots, (X_n, Y_n) \subset \mathcal{X} \times \mathcal{Y}. \]

Define kernels \( k_{\mathcal{X}} \) and \( k_{\mathcal{Y}} \) on \( \mathcal{X} \) and \( \mathcal{Y} \), respectively.
(\( \mathcal{X} \) and \( \mathcal{Y} \) can be any spaces, once kernels are defined.)
Proposed method

Suppose the posterior kernel mean at time $t - 1$

\[
m_{x_{t-1}|y_{1:t-1}} := \int k(x', x_{t-1}) p(x_{t-1}|y_{1:t-1}) dx_{t-1}.
\]

was already estimate.

At time $t$, do the following steps:

1. **Prediction**: estimate the prior kernel mean:

\[
m_{x_t|y_{1:t-1}} := \int k(x', x_t) p(x_t|y_{1:t-1}) dx_t
\]

2. **Correction**: given $y_t$, estimate the posterior kernel mean:

\[
m_{x_t|y_{1:t}} := \int k(x', x_t) p(x_t|y_{1:t}) dx_t.
\]
1. Prediction step

Suppose $m_{x_{t-1}|y_{1:t-1}}$ was estimated as

$$\hat{m}_{x_{t-1}|y_{1:t-1}} := \sum_{i=1}^{n} w_{t-1,i} k(x, X_{t-1,i}).$$

Estimate the *prior kernel mean* $m_{x_t|y_{1:t-1}}$ by sampling:

$$X_{t,i} \sim p(x_t|x_{t-1} = X_{t-1,i}), \quad (i = 1, ..., n),$$

under the *transition model*.

$$\hat{m}_{x_t|y_{1:t-1}} := \sum_{i=1}^{n} w_{t-1,i} k(x, X_{t,i}).$$
1. Prediction step: theoretical analysis

Under some assumptions, the error of the prediction step is

\[ \| m_{x_t|y_{t-1}} - \hat{m}_{x_t|y_{1:t-1}} \|_{\mathcal{H}_x}^2 \]

\[ = O(\| m_{x_{t-1}|y_{1:t-1}} - \hat{m}_{x_{t-1}|y_{1:t-1}} \|_{\mathcal{H}_x}^2 + \sum_{i=1}^{n} w_{t-1,i}^2) \]

(a): error at \( t - 1 \).

(b): inverse of effective sample size: close to 0 when the variance of the weights are small.

e.g. if \( w_{t-1,i} = 1/n \), (b) = 1/n.
2. Correction step

Given $y_t$, estimate the posterior kernel mean $m_{x_t|y_1:t}$ by the kernel Bayes’ rule (Fukumizu et al., 2011):

$$\hat{m}_{x_t|y_1:t} := \sum_{i=1}^{n} w_{t,i} k_{\mathcal{X}}(\cdot, X_i),$$

where $w_{t,i}$ are calculated from $y_t$, $\hat{m}_{x_t|y_1:t-1}$ and $\{(X_i, Y_i)\}_{i=1}^{n}$:

$$m := (\hat{m}_{x_t|y_1:t-1}(X_j)) \in \mathbb{R}^n, \quad k_Y := (k_Y(y_t, Y_j)) \in \mathbb{R}^n$$
$$G_X = (k_{\mathcal{X}}(X_i, X_j)) \in \mathbb{R}^{n \times n}, \quad G_Y = (k_Y(Y_i, Y_j)) \in \mathbb{R}^{n \times n}$$
$$\Lambda := \text{diag}((G_X + n \varepsilon I_n)^{-1}m) \in \mathbb{R}^{n \times n}$$
$$w_t := \Lambda G_Y((\Lambda G_Y)^2 + \delta I_n)^{-1}\Lambda k_Y \in \mathbb{R}^n,$$

This is a consistent estimator of $m_{x_t|y_1:t}$ (Fukumizu et al., 2013).
State estimation

The estimate \( \hat{m}_{x_t|y_{1:t}} = \sum_{i=1}^{n} w_{t,i} k_x(\cdot, X_i) \) can be used for estimating the expectation of a smooth function (Kanagawa and Fukumizu, 2014):

\[
\int f(x_t) p(x_t|y_{1:t}) dx_t.
\]

This is done by

\[
\sum_{i=1}^{n} w_{t,i} f(X_i).
\]

e.g. posterior mean \( \int x_t p(x_t|y_{1:t}) dx_t \) can be estimated by

\[
\sum_{i=1}^{n} w_{t,i} X_i.
\]
Experiments
We compare with the following methods:

**kNN-PF (Vlassis et al., 2002):** the observation model is learned with k-NN approach, and then combined with a particle filter.

**GP-PF (Ferris et al., 2006):** the observation model is learned with Gaussian process regression, and then combined with a particle filter.

**KBR filter (Fukumizu et al., 2011):** kernel embedding-based filter that also learns the transition model from data as well as the observation model. Used as baseline.

State estimation is done by estimation of posterior mean.
Synthetic experiment 1

transition model:

\[ x_1 = v_1, \quad v_1 \sim \mathcal{N}(0, 1/(1 - 0.9^2)). \]
\[ x_t = 0.9x_{t-1} + 0.5u_t + 0.5v_t, \quad v_t \sim \mathcal{N}(0, 1). \]

observation model:

\[ y_t = x_t + w_t, \quad w_t \sim \mathcal{N}(0, 1). \]

- Linear Gaussian model.
- Control \( u_t \) are generated randomly generated from Gaussian \( \mathcal{N}(0, 1) \).
- Training samples \( \{(X_i, Y_i)\} \) are generated by running the model.
Synthetic experiment 1

- GP-PF performed the best, since the noise is Gaussian.

Figure: RMSE of state estimation, varying the training data size. (KMC) proposed method. (KBR) KBR filter. (NN) kNN-PF. (GP) GP-PF.
Synthetic experiment 2

transition model:

\[ x_1 = \nu_1, \quad \nu_1 \sim \mathcal{N}(0, 1/(1 - 0.9^2)).\]
\[ x_t = 0.9x_{t-1} + 0.5u_t + 0.5\nu_t, \quad \nu_t \sim \mathcal{N}(0, 1).\]

observation model:

\[ y_t = 0.5 \exp(x_t/2)w_t, \quad w_t \sim \mathcal{N}(0, 1).\]

- Transition model is the same as Experiment 1.
- Observation model is nonlinear-transformation + multiplicative noise.
- Control \( u_t \) are generated randomly generated from Gaussian \( \mathcal{N}(0, 1) \).
Synthetic experiment 2

- Proposed method performed the best, due to strong nonlinearity of the observation model.

Figure: RMSE of state estimation, varying the training data size. (KMC) proposed method. (KBR) KBR filter. (NN) kNN-PF. (GP) GP-PF.
Vision-based mobile robot localization

**Problem:** estimate a mobile robot’s position $x_t$ for each time $t$, only given its vision images $y_1, \ldots, y_t$.

**Figure:** COLD database (Pronobis and Caputo, 2009)
Vision-based mobile robot localization

Experiment using the COLD database
(Pronobis and Caputo, 2009).

▶ state $x_t$ (position): Gaussian kernel $k_X$.
▶ observation $y_t$ (image): Spatial pyramid kernel $k_Y$
  (Lazebnik et al., 2006).
▶ Training samples $\{(X_i, Y_i)\}_{i=1}^n$: position-image pairs.
▶ Transition model $p(x_t|x_{t-1}, u_t)$: odometry motion model
  (Thrun et al., 2005).

Position is estimated as a sample point with maximum weight:

$$\arg\max_{X_i} \mathbf{w}_{t,i}$$
Result

- Proposed method performed best: even superior to kNN-PF, which was originally proposed to this task.

Figure: RMSE of state estimation, varying the training data size. (KMC) Proposed method, (KBR) KBR filter, (NN) kNN PF, (NAI) Baseline
We developed a filter for the setting where

- transition model $p(x_t|x_{t-1})$ is known.
- observation model $p(y_t|x_t)$ is unknown, but training samples

$$(X_1, Y_1), \ldots, (X_n, Y_n)$$

are given.

Our method can be applied if

- kernels are defined on $\mathcal{X}$ and $\mathcal{Y}$. 


*In Advances in NIPS 20*, pages 489–496.


Prediction of arm movement trajectories from ecog-recordings in humans.

COLD: COsy Localization Database.

*Learning with Kernels.*
MIT Press.

A Hilbert space embedding for distributions.

Hilbert space embeddings of conditional distributions with applications to dynamical systems.

Probabilistic Robotics.
MIT Press.

Auxiliary particle filter robot localization from high-dimensional sensor observations.

Prior knowledge improves decoding of finger flexion from electrocorticographic signals.
Frontiers in Neuroscience, 5:127.