粒子フィルタ

中野 慎也

「データ科学: 理論から実用へ」

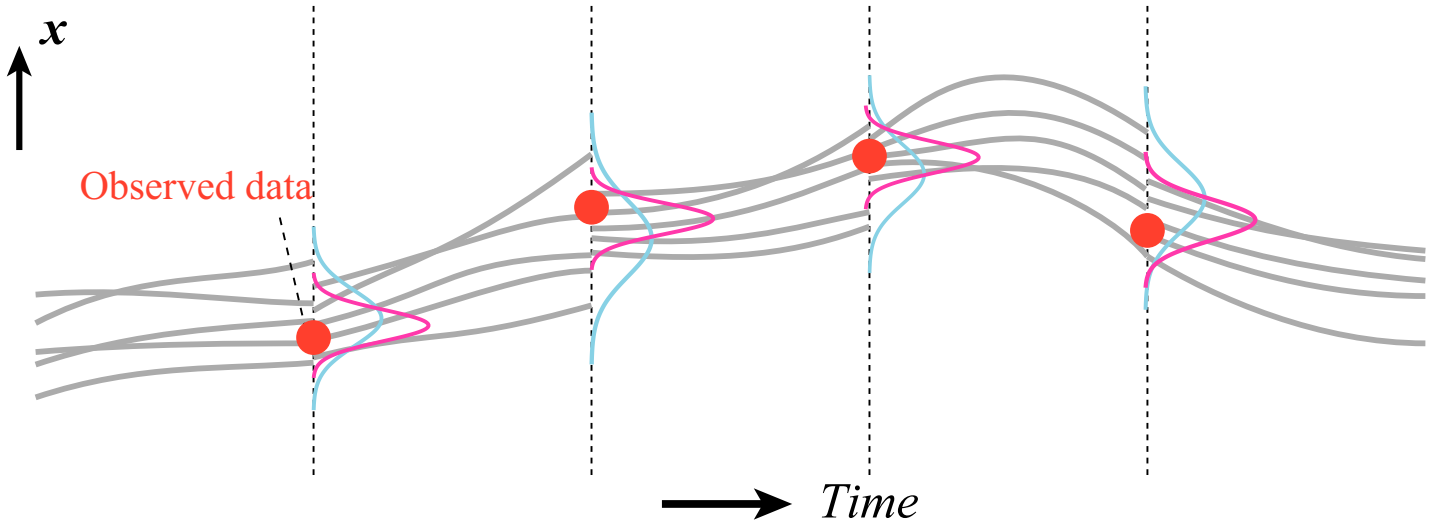
15 September 2023

逐次データ同化 (復習)

逐次データ同化では，

1. 前の時間ステップ $(*)$ から今の状態 x_k を予測する分布 $p(x_k|y_1,\ldots,y_{k-1})$ の計算，
2. $p(x_k|y_1,\ldots,y_{k-1})$ を事前分布とし，観測 y_k を用いて得た事後分布 $p(x_k|y_1,\ldots,y_{k-1},y_k)$ の計算，

を繰り返すことで，各時刻の x_k を順々に推定していく．



逐次データ同化 (復習)

時刻 t_{k-1} までの観測 $y_1, \ldots, y_{k-1}$ を使った x_k の予測 $p(x_k \mid y_{1:k-1})$:

$$p(x_k \mid y_{1:k-1}) = \int p(x_k \mid x_{k-1}) p(x_{k-1} \mid y_{1:k-1}) dx_{k-1}.$$

Given

時刻 t_k の観測 y_k の情報を追加して x_k の予測を修正 (フィルタリング) :

$$p(x_k \mid y_{1:k}) = \frac{p(y_k \mid x_k) p(x_k \mid y_{1:k-1})}{\int p(y_k \mid x_k) p(x_k \mid y_{1:k-1}) dx_k}.$$

Given

- 予測とフィルタリングの計算を繰り返すことで，各時間ステップの推定を行う.

逐次データ同化手法

- カルマンフィルタ (線型性，ガウス性を仮定)
- アンサンブルカルマンフィルタ (予測は非線型を考慮．フィルタリングで線型・ガウス性を仮定．)
- アンサンブル変換カルマンフィルタ (予測は非線型を考慮．フィルタリングで線型・ガウス性を仮定．計算効率が高い傾向．)
- 粒子フィルタ (予測，フィルタリングとも非線型，非ガウスの問題を扱える．)

粒子フィルタ

- アンサンブルカルマンフィルタ (EnKF) では，状態 $\boldsymbol{x}_k$ の確率分布がガウス分布であること，観測モデルが線型・ガウスの形で書けることを仮定している．
- しかし，システムが非線型の場合，$p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ をガウス分布で与えたとしても，$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})$ は一般にガウス分布にはならない．
- 非線型性の強いシステムでは，EnKF を適用しても，よい推定結果が得られない場合がある．
- 粒子フィルタは，状態 $\boldsymbol{x}_k$ の確率分布の非ガウス的な特徴を考慮できる手法である．

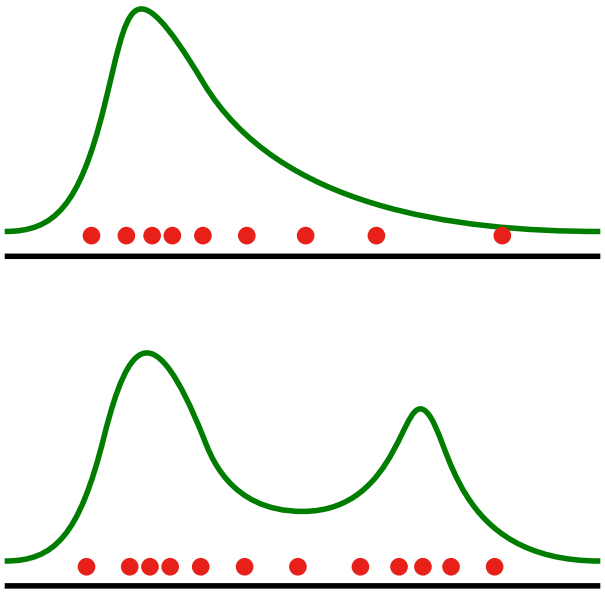
粒子フィルタの概要

- 粒子フィルタでは，予測分布 $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})$ の計算を，アンサンブルカルマンフィルタと同様のモンテカルロ法に基づく手続きによって行う．
- 一方，フィルタ分布 $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ の計算手続きは大きく異なる．
  1. 粒子フィルタでは，予測分布 $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})$ を近似するアンサンブルを得た後，その各メンバー (粒子) について観測への当てはまりのよさをまず評価する．
  2. そして，観測への当てはまりの悪いメンバーは破棄し，代わりに当てはまりのよいメンバーの数を増やすことにより，フィルタ分布 $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ を表現するアンサンブルを新たに構成する．

モンテカルロ近似

- 粒子フィルタでは，アンサンブルカルマンフィルタと同じく，粒子の集合 (アンサンブル)$\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$ で確率密度関数 $p(\boldsymbol{x})$ を近似する:

$$p(\boldsymbol{x}) \approx \frac{1}{N} \sum_{i=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}^{(i)}). \tag{1}$$



モンテカルロ近似による予測

時刻 t_{k-1} のフィルタ分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ が

$$p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1}) \approx \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-1|k-1}^{(i)})$$

のように近似されているとする．

アンサンブルカルマンフィルタのところで確認したように，$\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ と独立に $p(\boldsymbol{v}_k) = \mathcal{N}(\boldsymbol{0}, \mathbf{Q})$ からのランダムサンプル $\{\boldsymbol{v}_k^{(i)}\}$ を抽出し，

$$\boldsymbol{x}_{k|k-1}^{(i)} = \boldsymbol{f}_k(\boldsymbol{x}_{k-1|k-1}^{(i)}) + \boldsymbol{v}_k^{(i)} \tag{2}$$

のように $\{\boldsymbol{x}_{k|k-1}^{(1)}, \ldots, \boldsymbol{x}_{k|k-1}^{(N)}\}$ を生成すれば，これが $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})$ から抽出した $\boldsymbol{x}_k$ のランダムサンプルになる．

粒子フィルタの導出

粒子フィルタでフィルタ分布 $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ を計算するアルゴリズムは，ベイズの定理

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k)\, p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})}{\int p(\boldsymbol{y}_k|\boldsymbol{x}_k)\, p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})\, d\boldsymbol{x}_k}$$

に，モンテカルロ近似

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) \approx \frac{1}{N}\sum_{i=1}^{N} \delta(\boldsymbol{x}_k - \boldsymbol{x}_{k|k-1}^{(i)})$$

を代入することで得られる．

粒子フィルタの導出

ベイズの定理

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k)\, p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})}{\int p(\boldsymbol{y}_k|\boldsymbol{x}_k)\, p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})\, d\boldsymbol{x}_k}$$

の式に，モンテカルロ近似の式を代入すると，ベイズの定理の式の右辺の分子は，

$$p(\boldsymbol{y}_k|\boldsymbol{x}_k)\, p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) \approx \frac{1}{N}\sum_{i=1}^{N} p\left(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)}\right) \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_{k|k-1}^{(i)}\right) \tag{3}$$

となり，右辺の分母は以下のようになる:

$$\begin{aligned}\int p(\boldsymbol{y}_k|\boldsymbol{x}_k)\, p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})\, d\boldsymbol{x}_k &\approx \int \frac{1}{N}\sum_{i=1}^{N} \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_{k|k-1}^{(i)}\right)\, p(\boldsymbol{y}_k|\boldsymbol{x}_k)\, d\boldsymbol{x}_k \\ &= \frac{1}{N}\sum_{i=1}^{N} p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)}).\end{aligned} \tag{4}$$

粒子フィルタの導出

したがって，

$$
\begin{aligned}
p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) &\approx \frac{1}{\sum_j p\left(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(j)}\right)} \sum_{i=1}^{N} p\left(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)}\right) \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_{k|k-1}^{(i)}\right) \\
&= \sum_{i=1}^{N} \beta_k^{(i)} \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_{k|k-1}^{(i)}\right).
\end{aligned} \tag{5}
$$

のように，予測分布を表す粒子 $\{\boldsymbol{x}_{k|k-1}^{(i)}\}$ に重みをつけた形で書くことができる．

但し，重みは以下のように定義した:

$$
\beta_k^{(i)} = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)})}{\sum_j p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(j)})}. \tag{6}
$$

粒子フィルタの導出

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) = \sum_{i=1}^{N} \beta_k^{(i)} \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_{k|k-1}^{(i)}\right).$$

の式のままでも，フィルタ分布の近似になっているが，次の予測ステップに進むために，さらなる近似を適用する．

そのために，以下を満たす整数列 $\{m_k^{(i)}\}_{i=1}^N$ を導入する:

$$m_k^{(i)} \approx N\beta_k^{(i)} \quad \left(\sum m_k^{(i)} = N;\ m_k^{(i)} \geq 0\right). \tag{7}$$

すると，$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ はさらに以下のように近似できる．

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) &\approx \sum_{i=1}^{N} \beta_k^{(i)} \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_{k|k-1}^{(i)}\right) \\ &\approx \sum_{i=1}^{N} \frac{m_k^{(i)}}{N} \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_{k|k-1}^{(i)}\right) \end{aligned} \tag{8}$$

リサンプリング

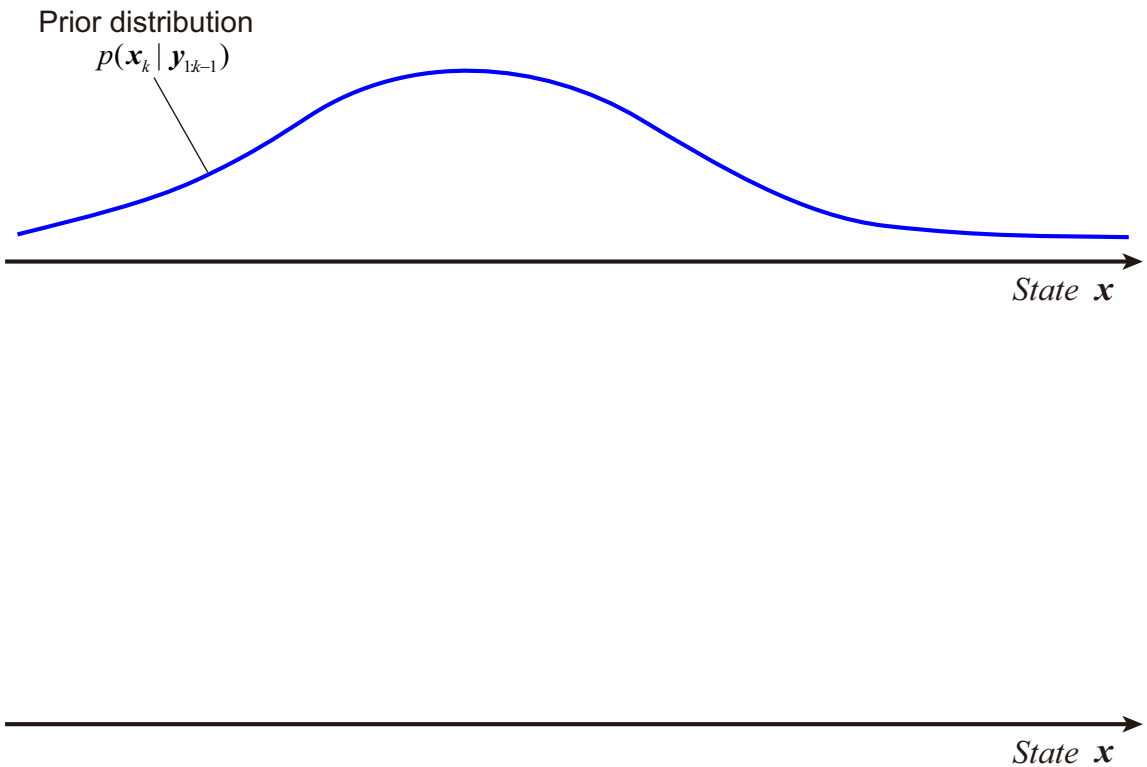
予測分布を表現する粒子 $\{\boldsymbol{x}_{k|k-1}^{(i)}\}$ のうち i 番目の粒子 $\boldsymbol{x}_{k|k-1}^{(i)}$ の複製が $m_k^{(i)}$ 個ずつ含まれるような新たな粒子の集合 $\{\boldsymbol{x}_{k|k}^{(i)}\}_{i=1}^N$ を考えると，

$$
\begin{aligned}
p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) &\approx \sum_{i=1}^{N} \frac{m_k^{(i)}}{N} \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_{k|k-1}^{(i)}\right) \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_{k|k}^{(i)}\right)
\end{aligned}
\tag{9}
$$

のように変形できる．このような手続きをリサンプリングと呼ぶ．これでアンサンブルカルマンフィルタと同じ，モンテカルロ法に基づく予測分布の計算に進むことができる．

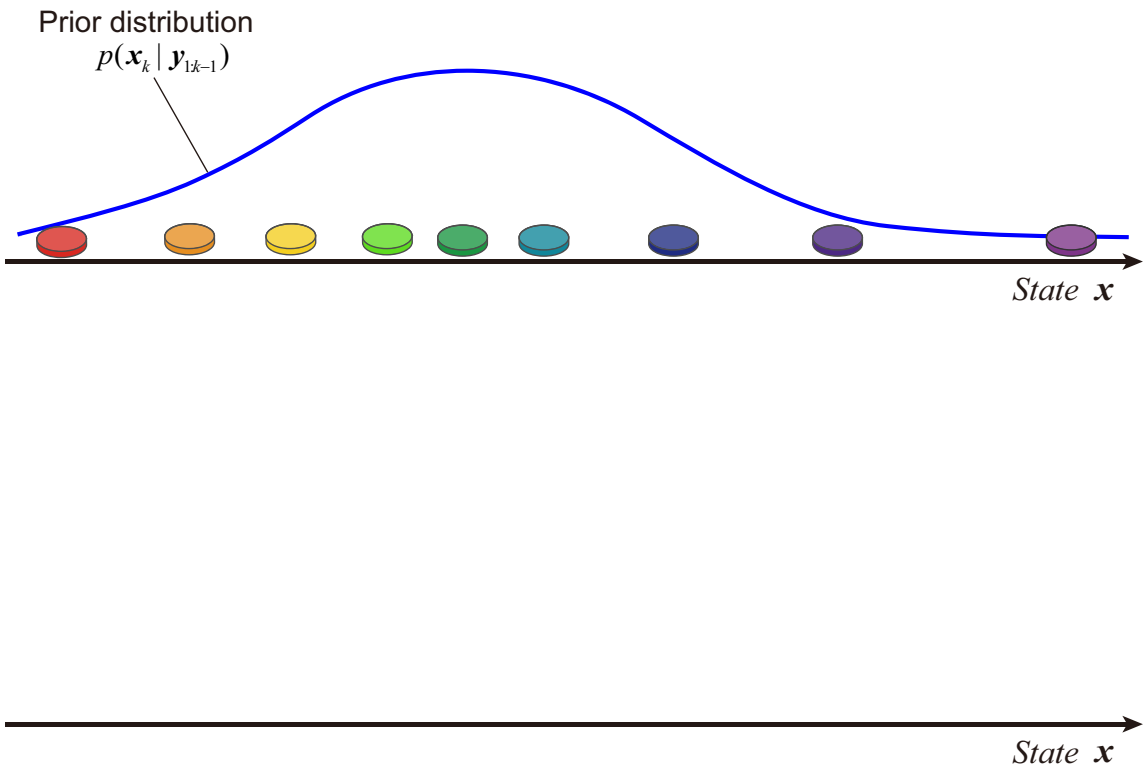
リサンプリング

観測への当てはまりが悪い粒子 (尤度が小さい粒子) を破棄し，その代わりに観測への当てはまりのよい粒子の数を増やす．



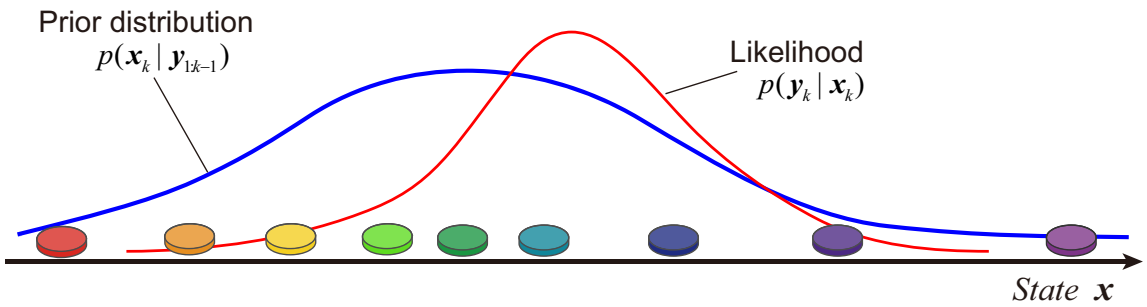
リサンプリング

観測への当てはまりが悪い粒子 (尤度が小さい粒子) を破棄し，その代わりに観測への当てはまりのよい粒子の数を増やす．



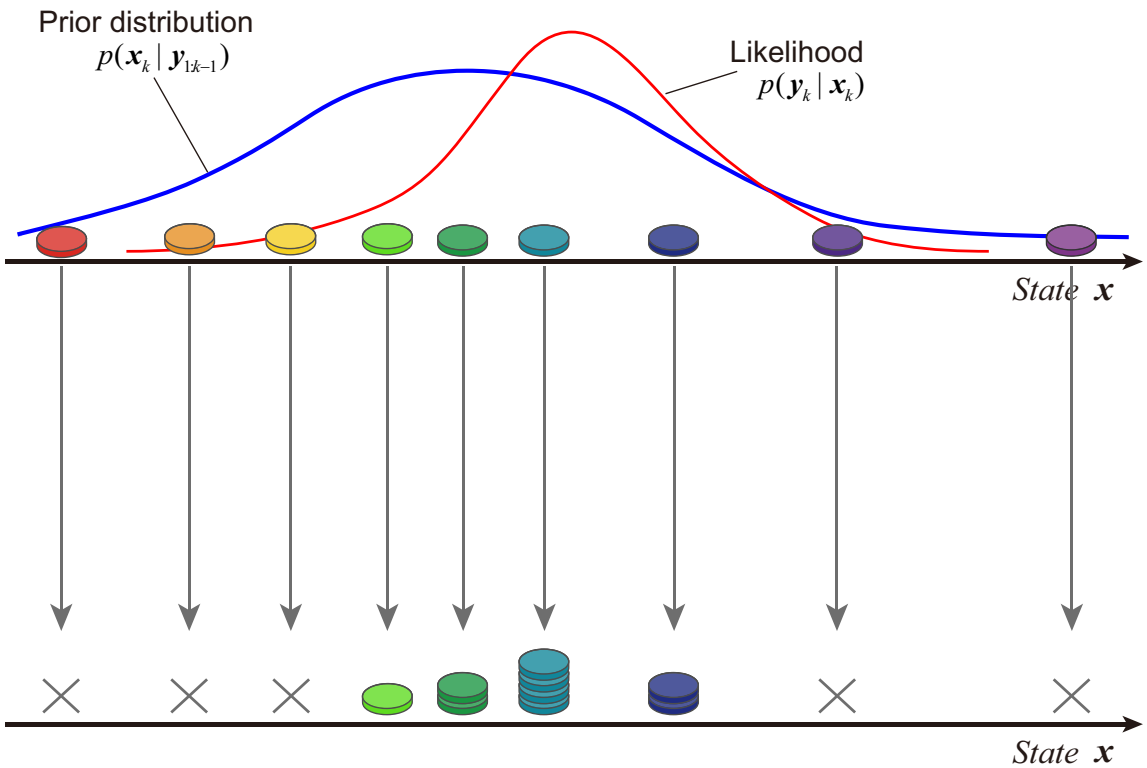
リサンプリング

観測への当てはまりが悪い粒子 (尤度が小さい粒子) を破棄し，その代わりに観測への当てはまりのよい粒子の数を増やす．



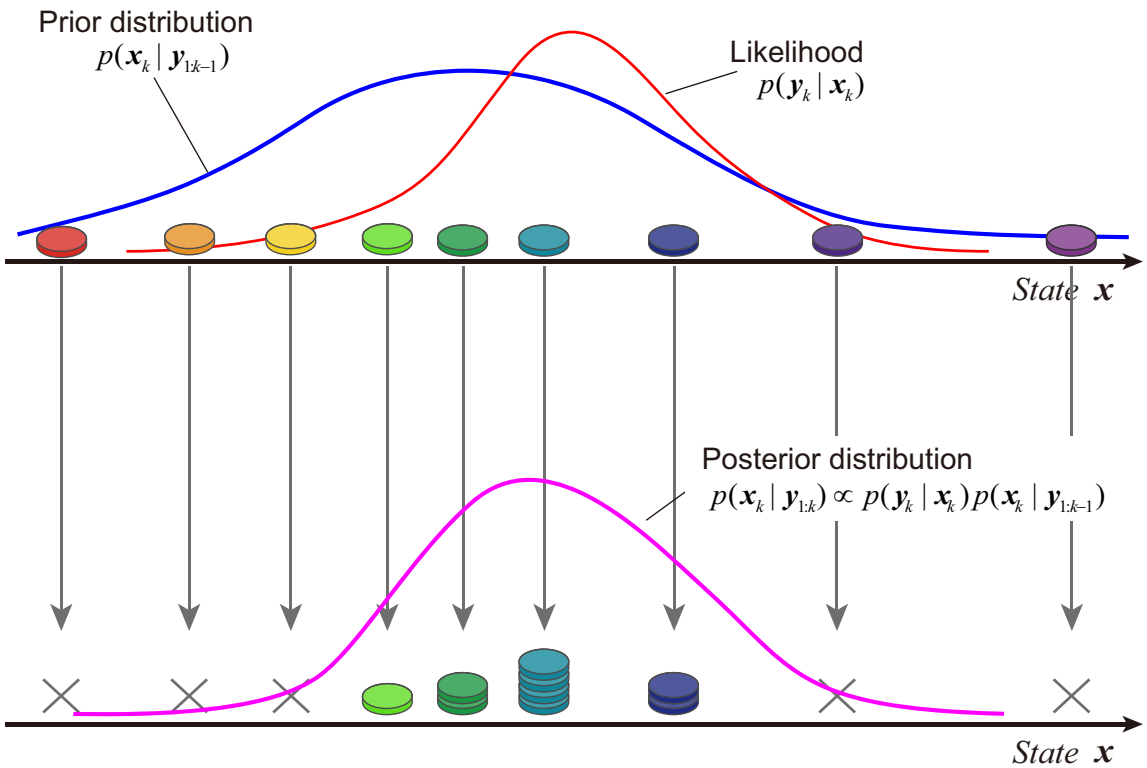
リサンプリング

観測への当てはまりが悪い粒子 (尤度が小さい粒子) を破棄し，その代わりに観測への当てはまりのよい粒子の数を増やす．



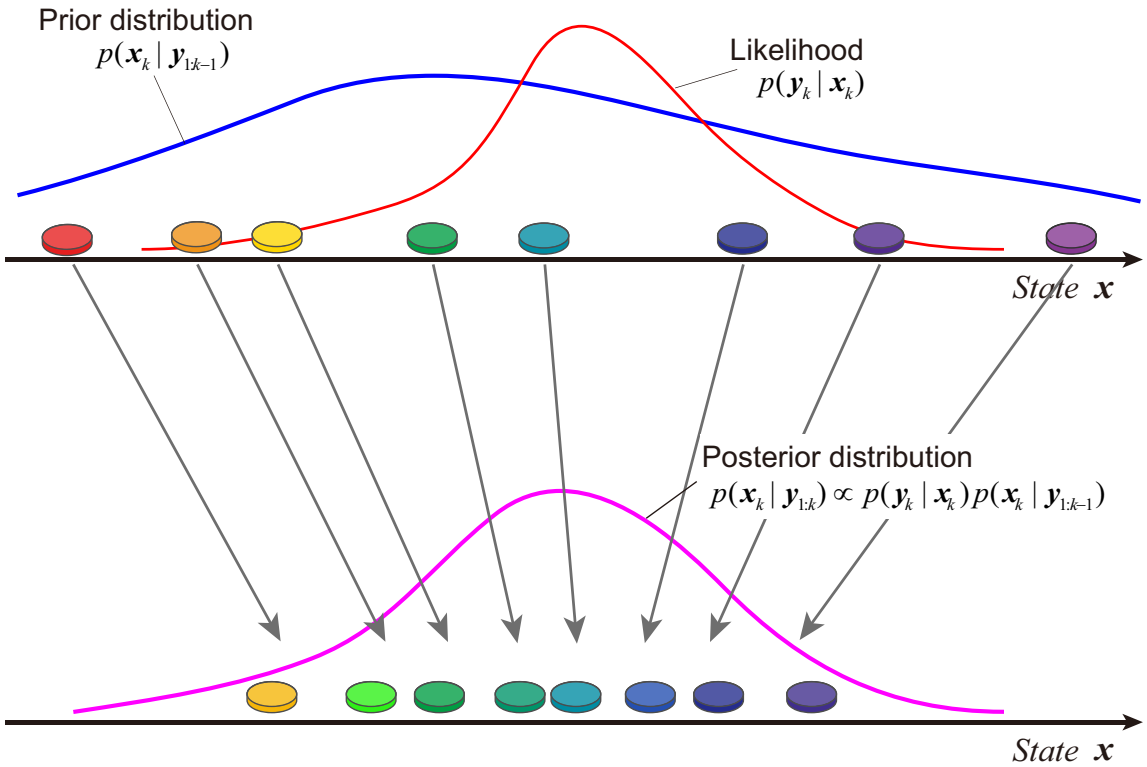
リサンプリング

観測への当てはまりが悪い粒子 (尤度が小さい粒子) を破棄し，その代わりに観測への当てはまりのよい粒子の数を増やす．



アンサンブルカルマンフィルタ

一方，アンサンブルカルマンフィルタでは，メンバーを観測に合う方向に動かすという操作を行う．



Remarks

- $p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)})$ はアンサンブルメンバー (粒子)$\boldsymbol{x}_{k|k-1}^{(i)}$ の観測データ $\boldsymbol{y}_k$ への当てはまりのよさを表しており，尤度と呼ばれる．
- 尤度 $p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)})$ は，$p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ がガウス分布になると仮定した場合には，そのまま

$$p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)}) = \frac{1}{\sqrt{(2\pi)^n|\mathbf{R}_k|}} \exp\left(-\frac{1}{2}(\boldsymbol{y}_k - \mathbf{H}_k\boldsymbol{x}_{k|k-1}^{(i)})^T\mathbf{R}_k^{-1}(\boldsymbol{y}_k - \mathbf{H}_k\boldsymbol{x}_{k|k-1}^{(i)})\right)$$

となる．

- 粒子フィルタでは $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ をガウス分布と仮定する必要はなく，様々な確率分布を用いることができる．

粒子フィルタのアルゴリズムのまとめ

- 予測:
  - 各アンサンブルメンバー (粒子) にシステムモデルを適用.

$$\boldsymbol{x}_{k|k-1}^{(i)} = \boldsymbol{f}_k(\boldsymbol{x}_{k-1|k-1}^{(i)}) + \boldsymbol{v}_k^{(i)}.$$

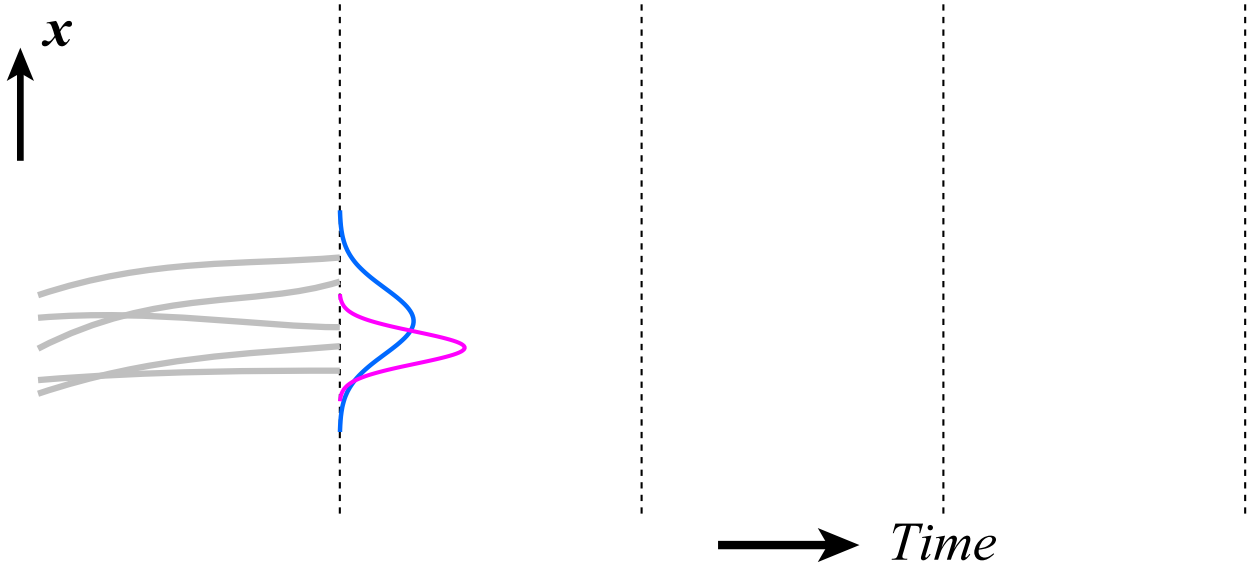
- フィルタリング:
  - 各メンバーについて，観測への当てはまり $l_k^{(i)} = p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)})$ を計算する.
  - $L_k = \sum_{i=1}^N l_k^{(i)}$ を求める.
  - 各メンバーの重み $\beta_k^{(i)} = l_k^{(i)}/L_k$ を求める.
  - アンサンブル $\{\boldsymbol{x}_{k|k-1}^{(i)}\}_{i=1}^N$ から，各メンバー $\boldsymbol{x}_{k|k-1}^{(i)}$ が $\beta_k^{(i)}$ の割合で抽出されるようにして復元抽出を N 回繰り返し，$\{\boldsymbol{x}_{k|k}^{(i)}\}_{i=1}^N$ を生成する.

なお，このアルゴリズムのことを特にブートストラップフィルタ (あるいはモンテカルロフィルタ) と呼び，より一般的なアルゴリズムを指して粒子フィルタと呼んでいる文献もある.

粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 x_k を推定する:

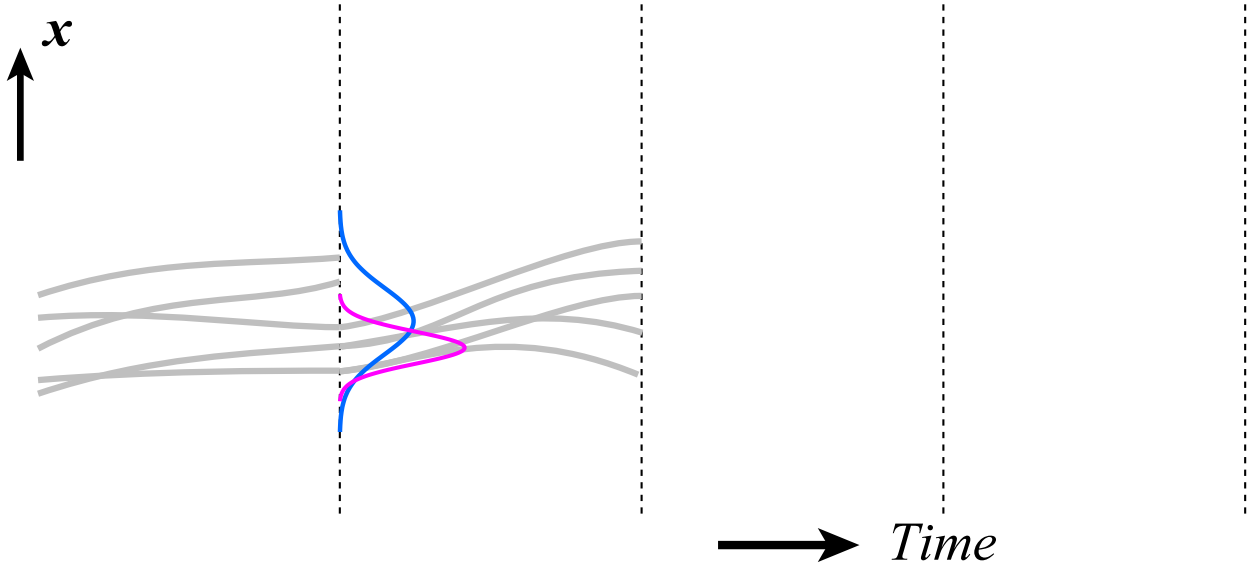
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 y_k を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 x_k を推定する:

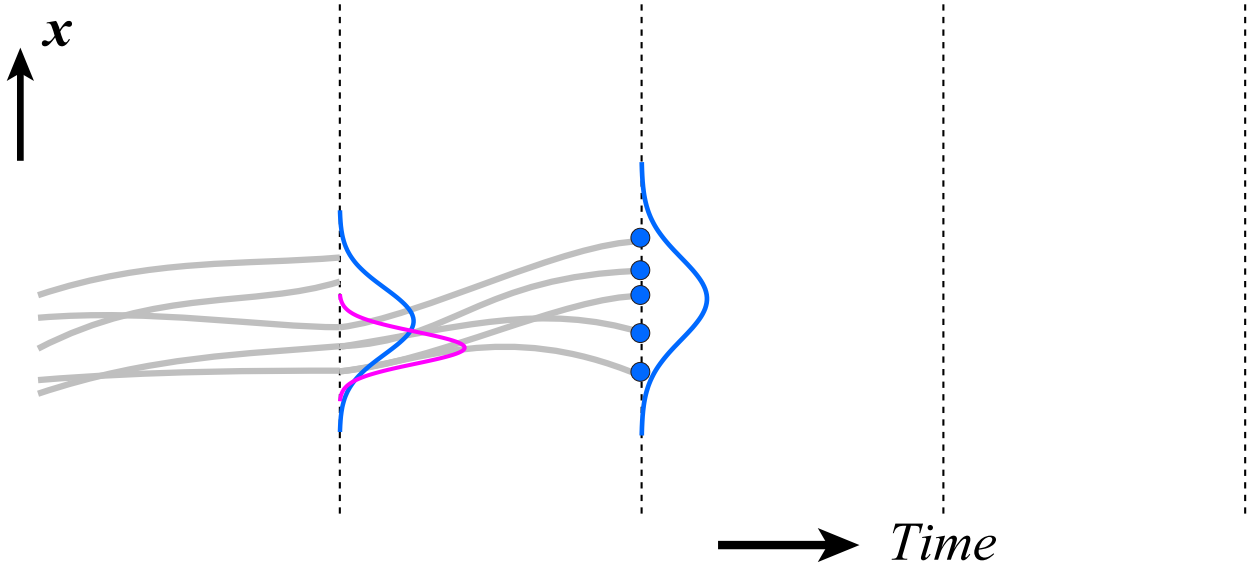
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 y_k を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 x_k を推定する:

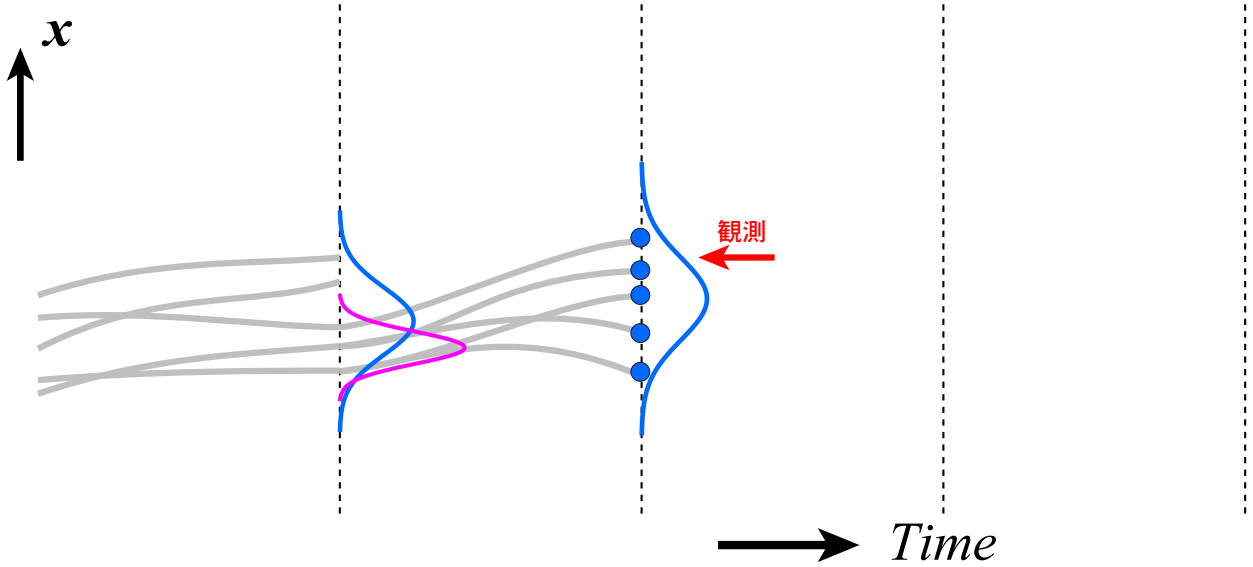
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 y_k を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 x_k を推定する:

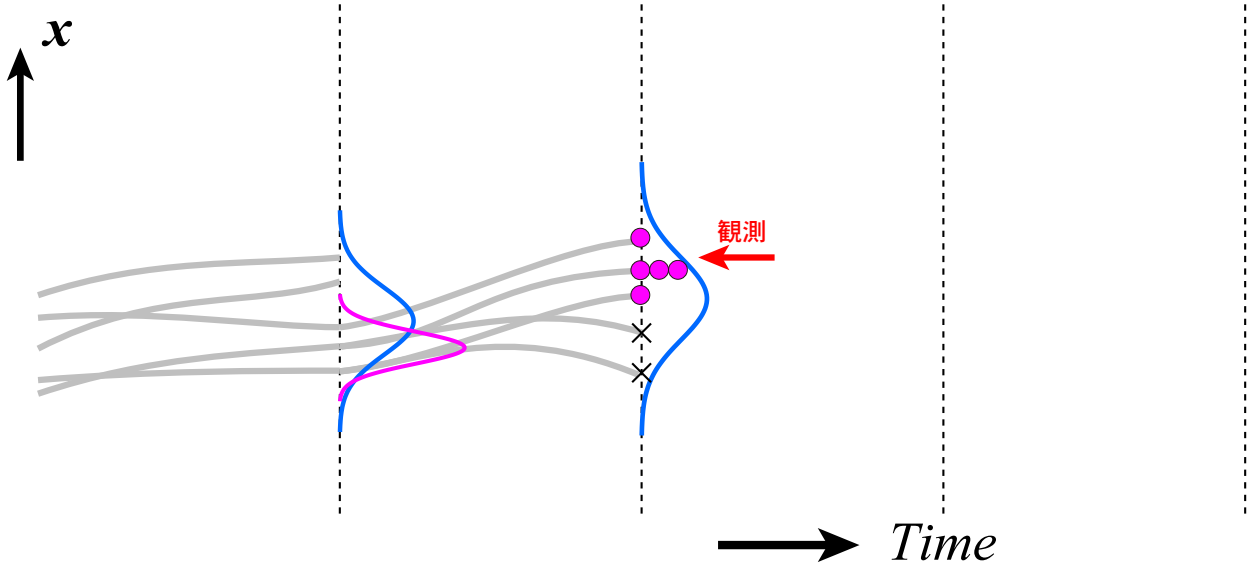
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 y_k を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 x_k を推定する:

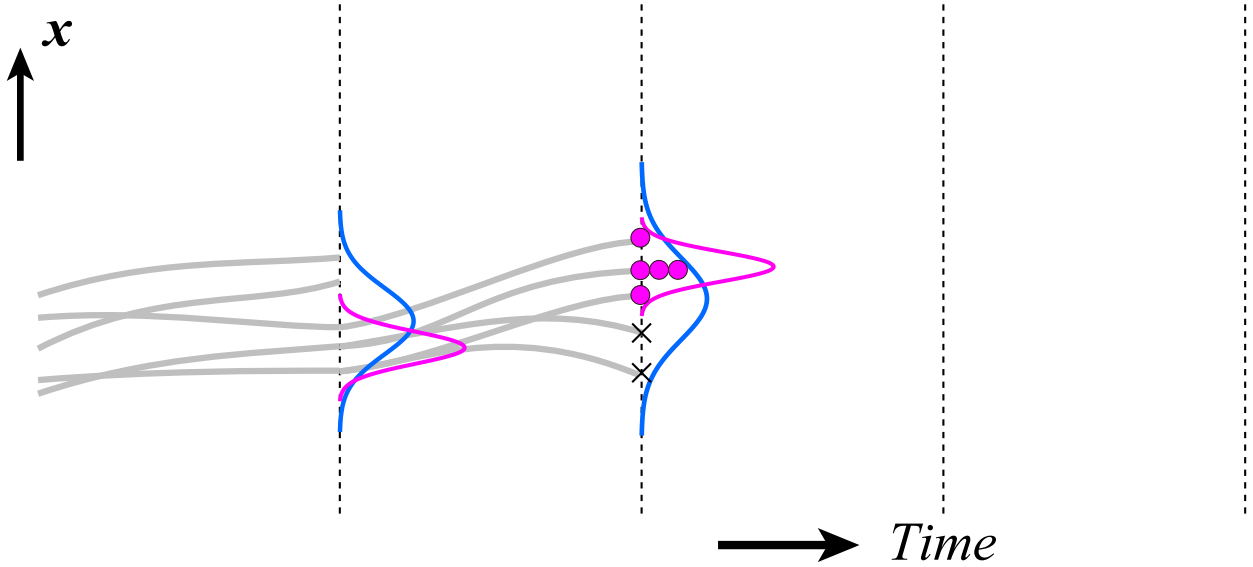
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 y_k を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 x_k を推定する:

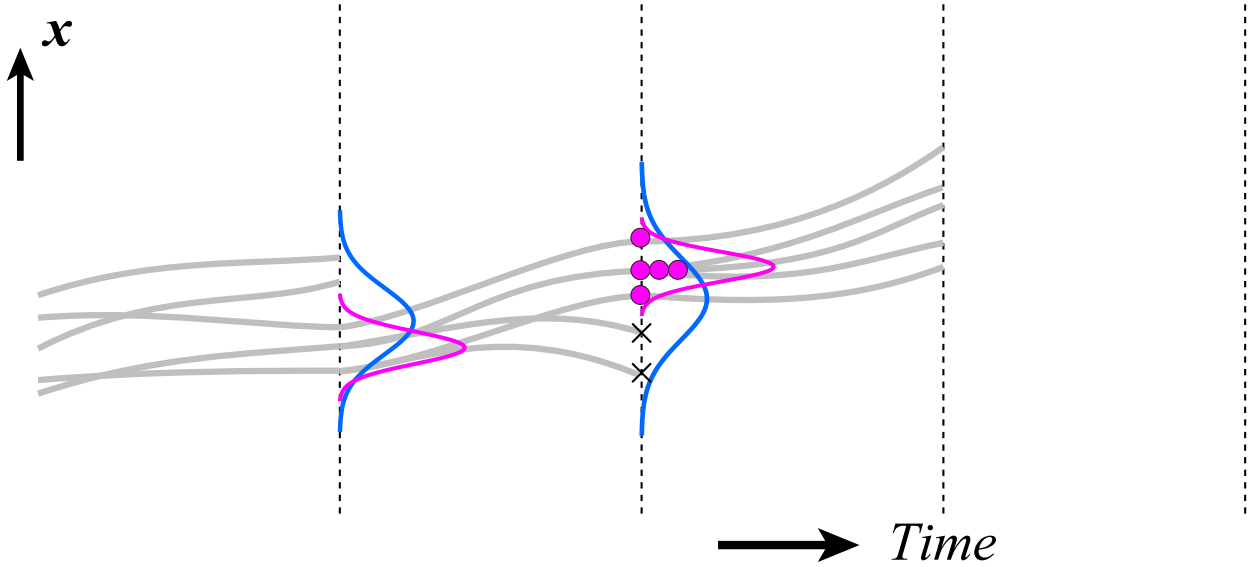
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 y_k を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 x_k を推定する:

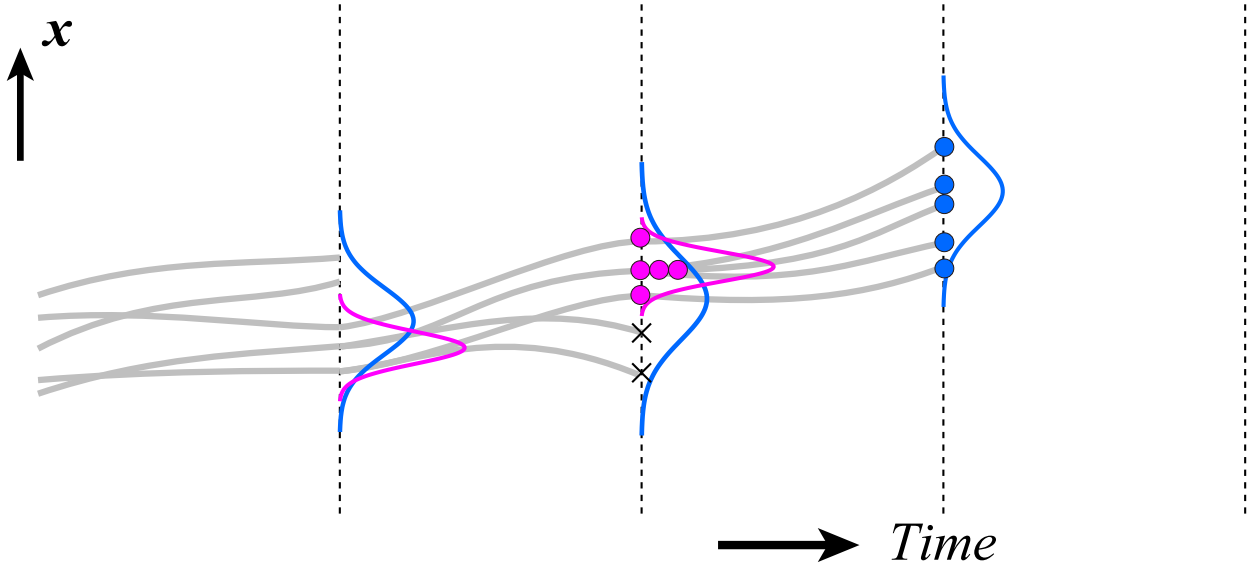
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 y_k を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 x_k を推定する:

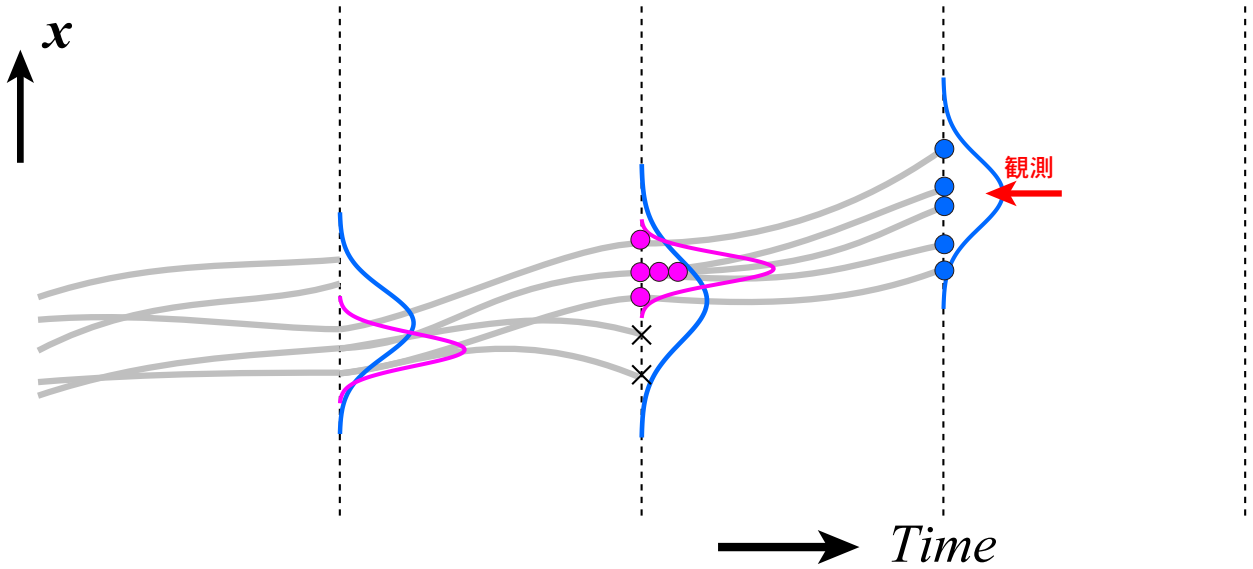
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 y_k を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 x_k を推定する:

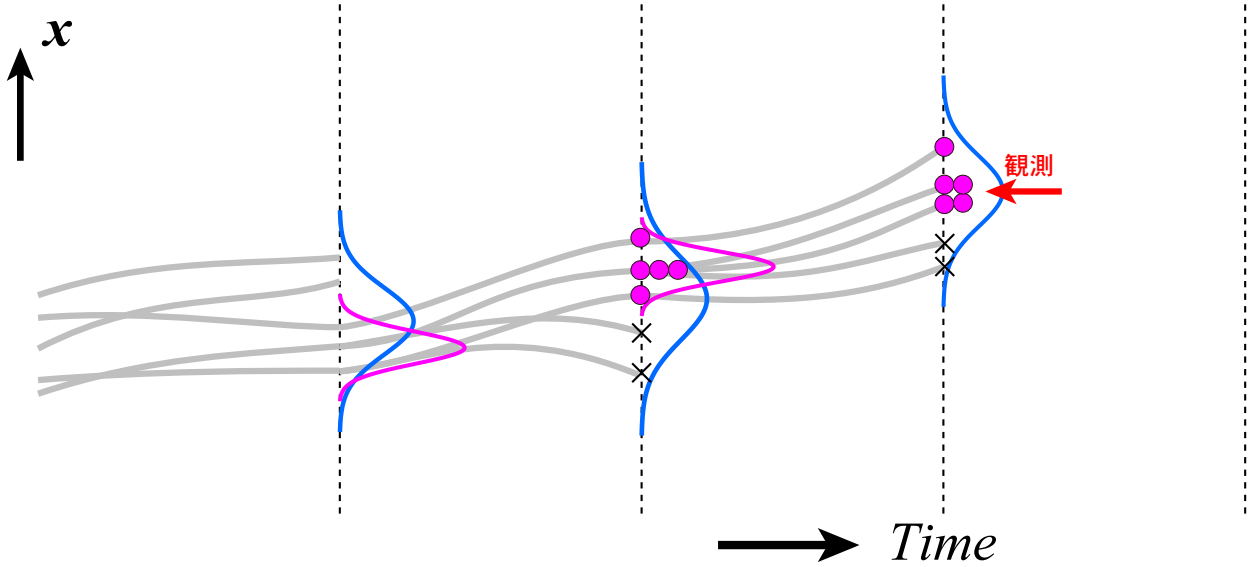
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 y_k を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 $\boldsymbol{x}_k$ を推定する:

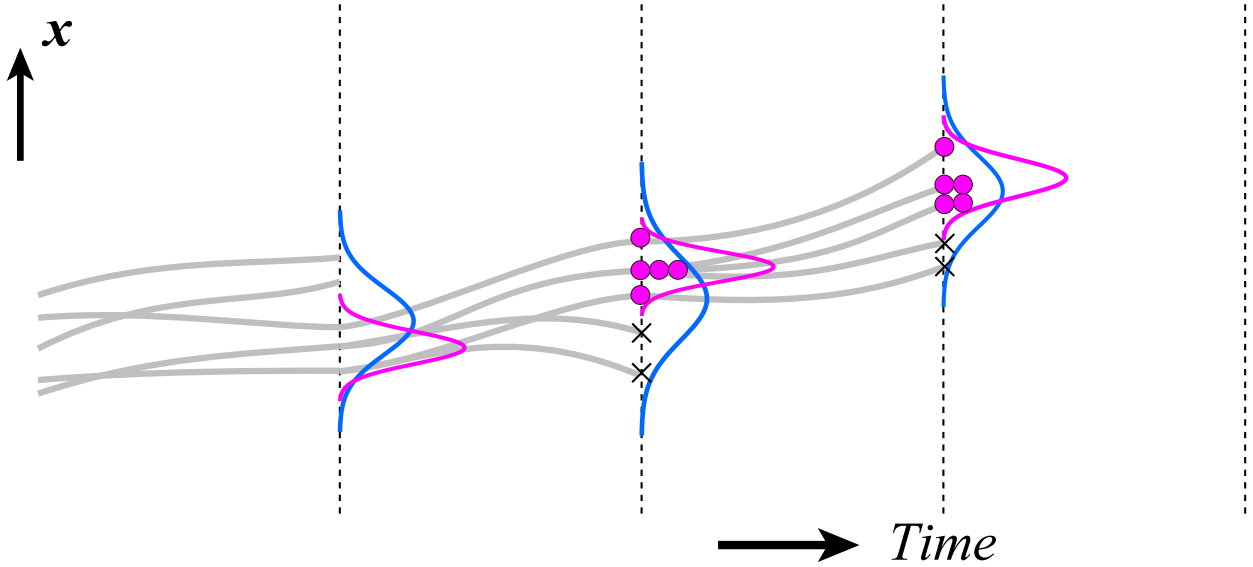
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 $\boldsymbol{y}_k$ を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 $\boldsymbol{x}_k$ を推定する:

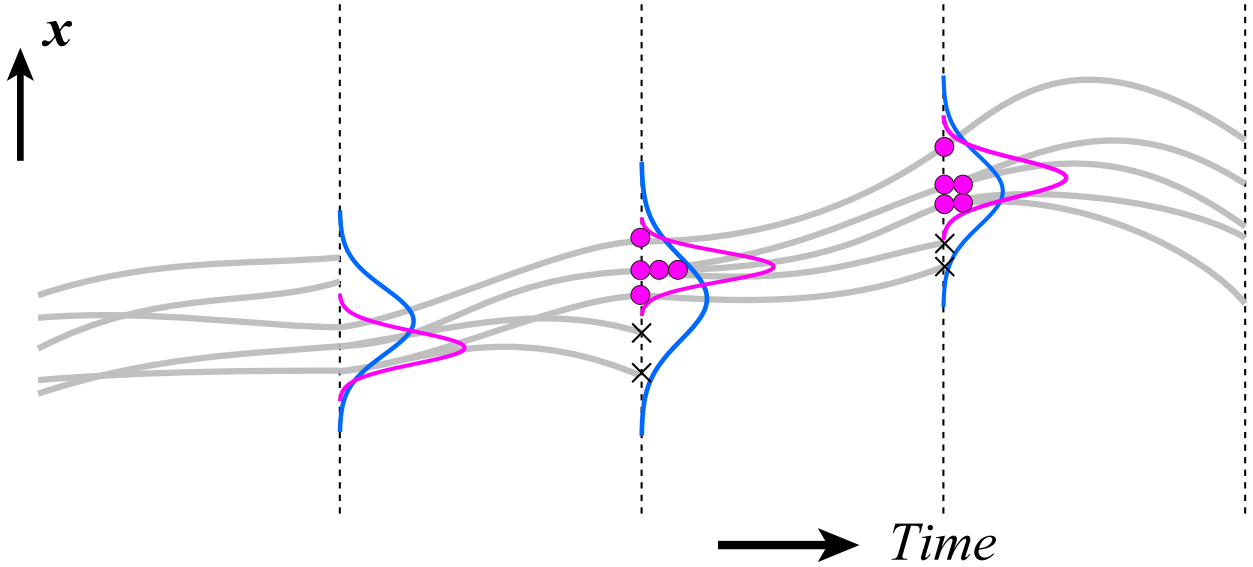
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 $\boldsymbol{y}_k$ を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 $\boldsymbol{x}_k$ を推定する:

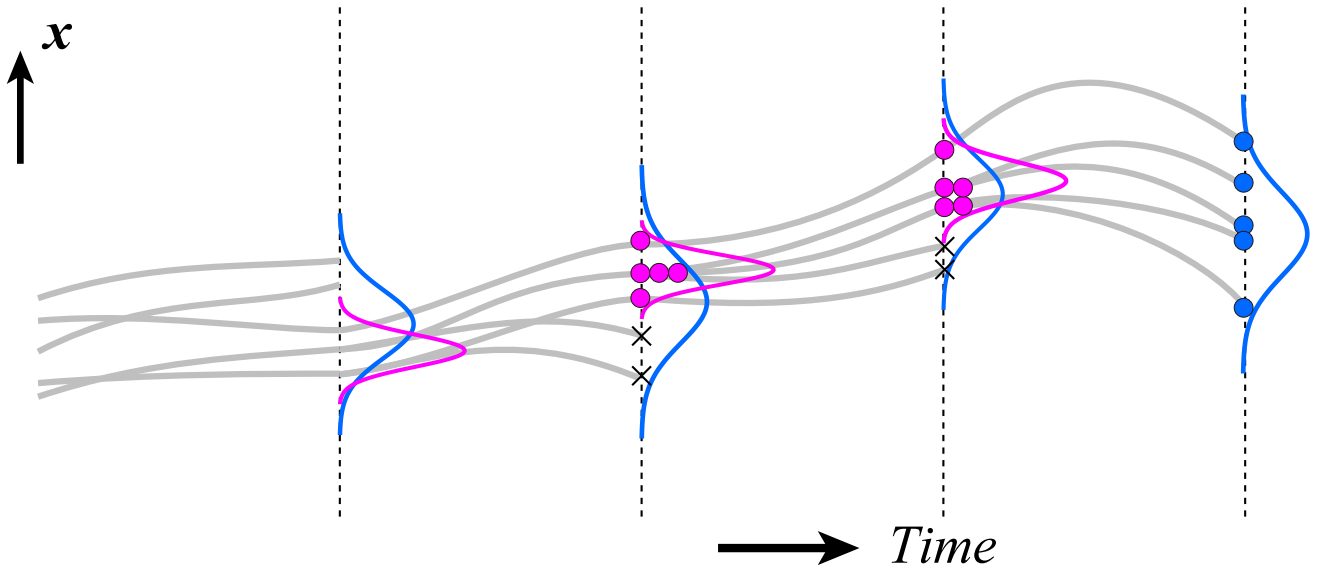
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 $\boldsymbol{y}_k$ を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 x_k を推定する:

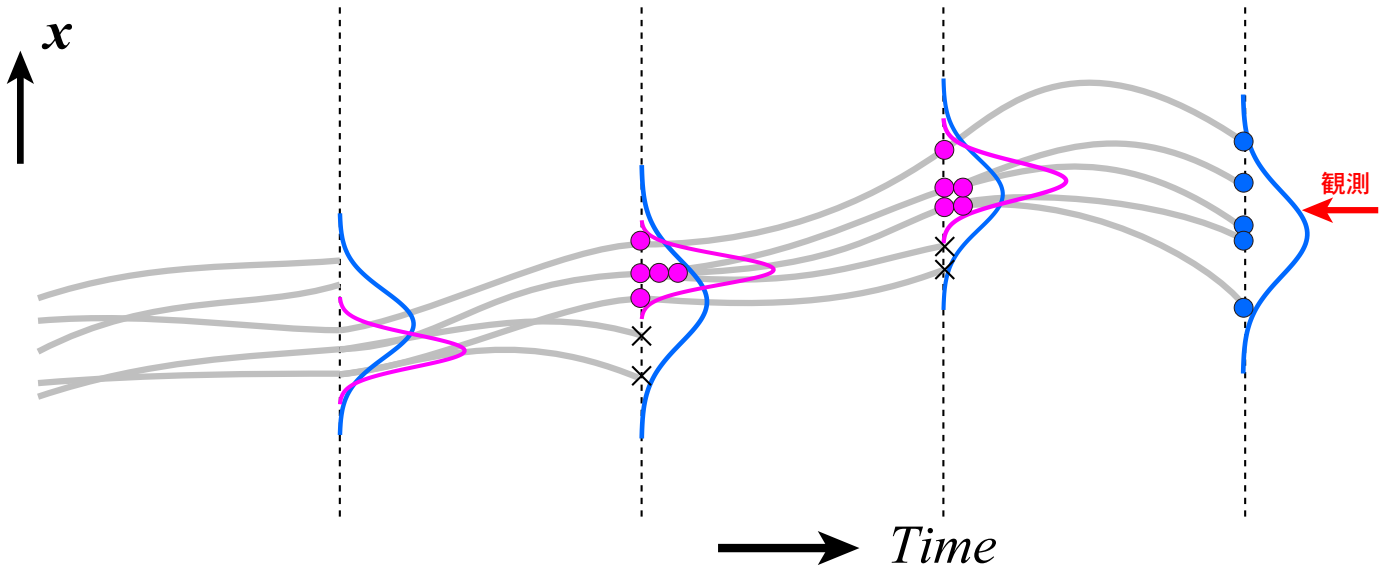
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 y_k を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 x_k を推定する:

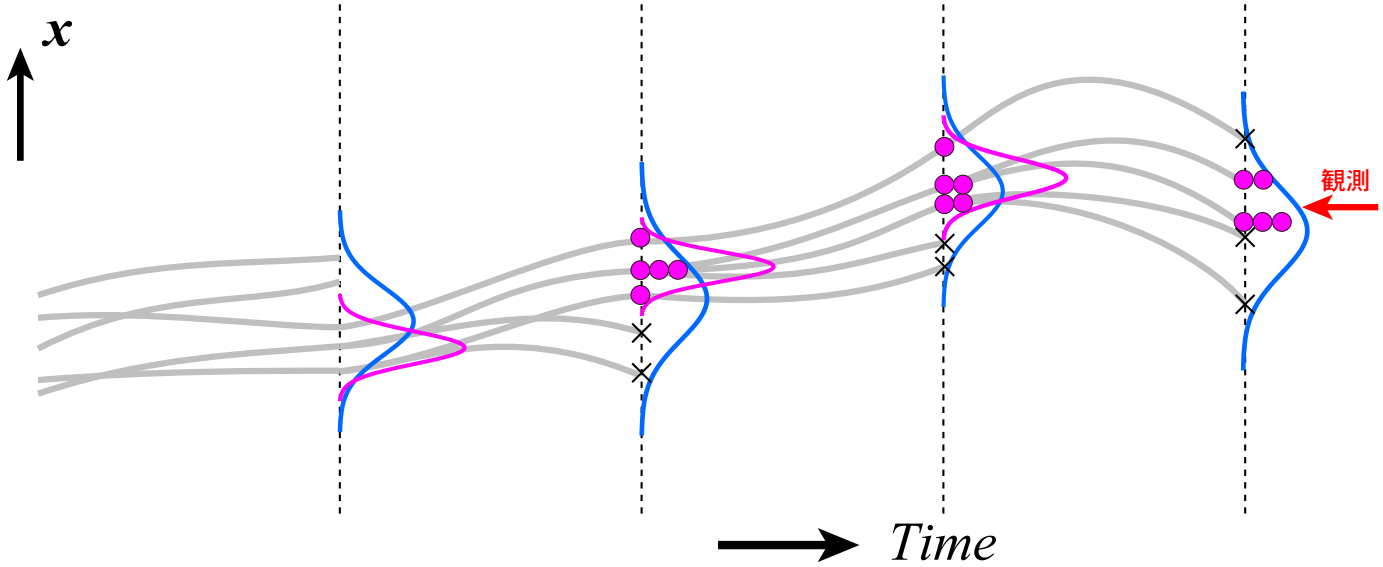
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 y_k を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 x_k を推定する:

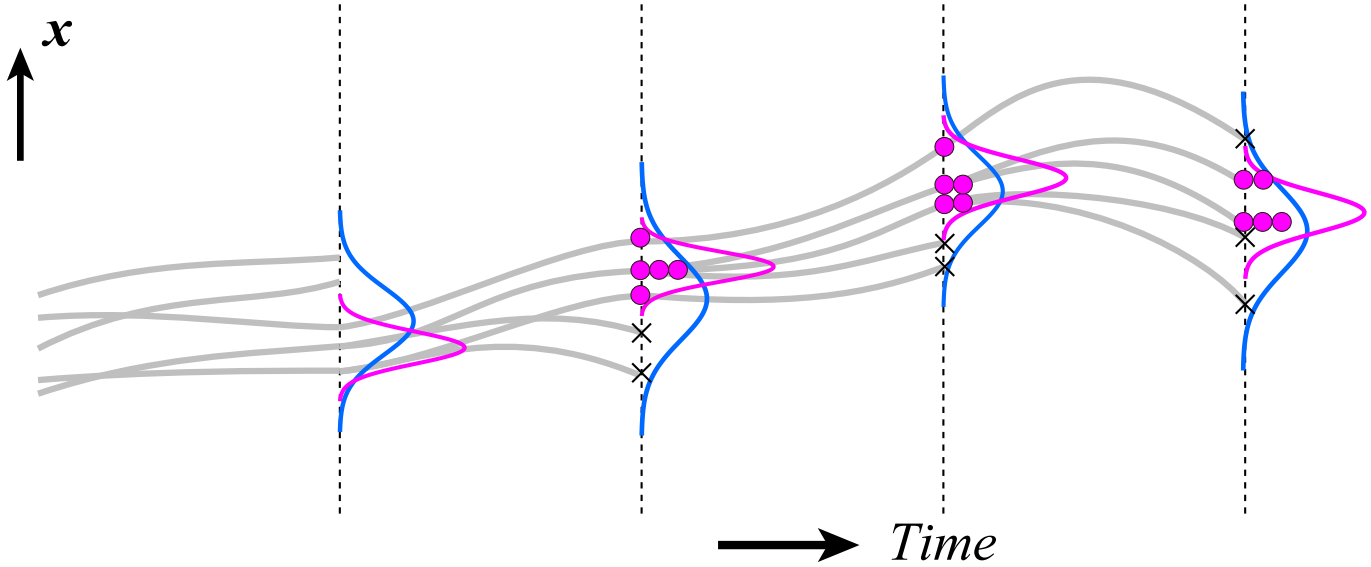
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 y_k を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 $\boldsymbol{x}_k$ を推定する:

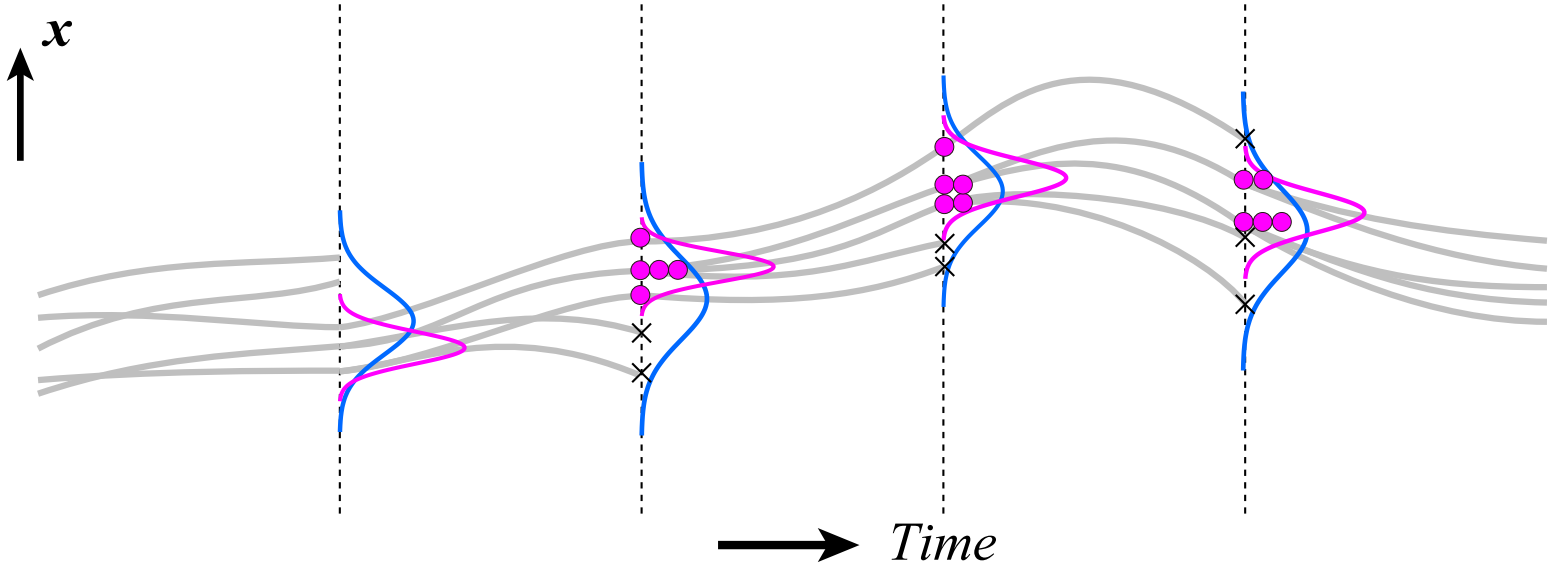
- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 $\boldsymbol{y}_k$ を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタのイメージ

粒子フィルタは，以下のような手順を繰り返すことで，各時刻の状態 $\boldsymbol{x}_k$ を推定する:

- 時刻 t_{k-1} の状態の確率分布 $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$ を表現する粒子 $\{\boldsymbol{x}_{k-1|k-1}^{(i)}\}$ が N 個得られているとする．
- 個々の粒子の時間発展を計算し，時刻 t_k の状態を予測．
- 観測 $\boldsymbol{y}_k$ を参照し，観測に合わない粒子は破棄．代わりに観測によく合う粒子を複製．



粒子フィルタの利点

- 実装がきわめて容易.
- 線型性，ガウス性などの仮定を一切置いていないため，あらゆるタイプの非線形・非ガウス型状態空間モデルに適用可能.
- カルマンフィルタと比較すると，
  - 非線型システム (状態遷移が非線型) の場合や，非ガウス的な過程などもそのまま扱うことができる
  - 観測モデルが非線型性，非ガウス性を含む場合にも自然に適用できる

などが利点となる.

粒子フィルタの欠点

- リサンプリングを繰り返すと，アンサンブルメンバーのほとんどが同一あるいは互いに極めて近い値を取ることになり，本来の広がりを持った確率分布をうまく表現できなくなってしまう (縮退とか退化と呼ばれる)．縮退が起こると，推定もうまく行かない．
- 1 回 (1 ステップ) で参照するデータが多い (観測ベクトル y_k が高次元) 場合や，観測の精度が高い (観測ノイズが小さい) 場合には特にこの問題が起きやすい．
- 縮退による性能の低下は，アンサンブルメンバー数を十分に増やすことである程度回避できる．しかし，メンバー数を増やせば多大な計算量が必要になってしまうため，粒子フィルタでは計算量が多くなりがちである．

粒子フィルタの欠点

- 粒子フィルタの縮退の問題を回避するための方法は，色々と提案されている．方向性としては，大きく分けて二つある．
  - 一つは，何らかの仮定や近似を導入し，その代わりに多様な粒子が生成されるようにする方法．一度，ガウス近似を行うガウス粒子フィルタ，複数の粒子を組み合わせる融合粒子フィルタなどがある．アンサンブル変換カルマンフィルタのところで述べる局所化を粒子フィルタで行う手法も，最近では提案されている．
  - もう一つは，リサンプリングの前になるべく観測データに近い粒子を生成しておくという方法．使える状況は限られるが，Optimal proposal distribution による方法などが知られる．

重みの計算時の留意点

粒子フィルタの重み

$$\beta_k^{(i)} = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)})}{\sum_j p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(j)})} \tag{6}$$

の計算には各粒子の尤度 $p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)})$ の値を使う．

しかし，データが高次元の問題では，そもそも対数尤度の絶対値が大きくなるため，計算機で exponential を計算しようとするとアンダーフローを起こしやすい．

重みの計算時の留意点

実際に粒子フィルタの重みの計算で必要なのは尤度そのものではなく尤度の比であることを考慮すれば，以下のようにしてアンダーフローの影響を回避できる．

1. 全粒子のうち，対数尤度 $\ell^{(i)}$ が最も大きい粒子を選び，その粒子を $\boldsymbol{x}^{(K)}$，その対数尤度を $\ell^{(K)}$ とする．
2. 各粒子ごとに，$\zeta^{(i)} = \exp\bigl(\ell^{(i)} - \ell^{(K)}\bigr)$ の値を計算する．
3. $Z = \sum \zeta^{(i)}$ を計算する．
4. $\beta^{(i)} = \zeta^{(i)}/Z$ で重みが得られる．

リサンプリングの方法

素直な方法:

粒子の集合 $\{\boldsymbol{x}_{k|k-1}^{(i)}\}$ からランダム復元抽出を N 回繰り返す．(各粒子が $\beta_k^{(i)}$ の確率で抽出されるものとして復元抽出を行う．)

. . . ただし，N 回のランダム復元抽出を行った場合，ある粒子 $\boldsymbol{x}_{k|k-1}^{(i)}$ が抽出される回数は二項分布に従う．

つまり，粒子 $\boldsymbol{x}_{k|k-1}^{(i)}$ が抽出される回数には $\sim \sqrt{N\beta_k^{(i)}(1-\beta_k^{(i)})}$ 程度のばらつきが生じる．

N が十分大きければ問題ないが，N があまり大きくない場合には，ランダム抽出を用いると本来の分布をうまく表現できない可能性がある．

比例配分 1

粒子による近似の目的は確率分布の形状をよりよく表現することなので，抽出される粒子数の比が尤度比になるべく近くなることが望ましいと考えられる．(粒子数の比が運次第で変わってしまうのは好ましくない．)

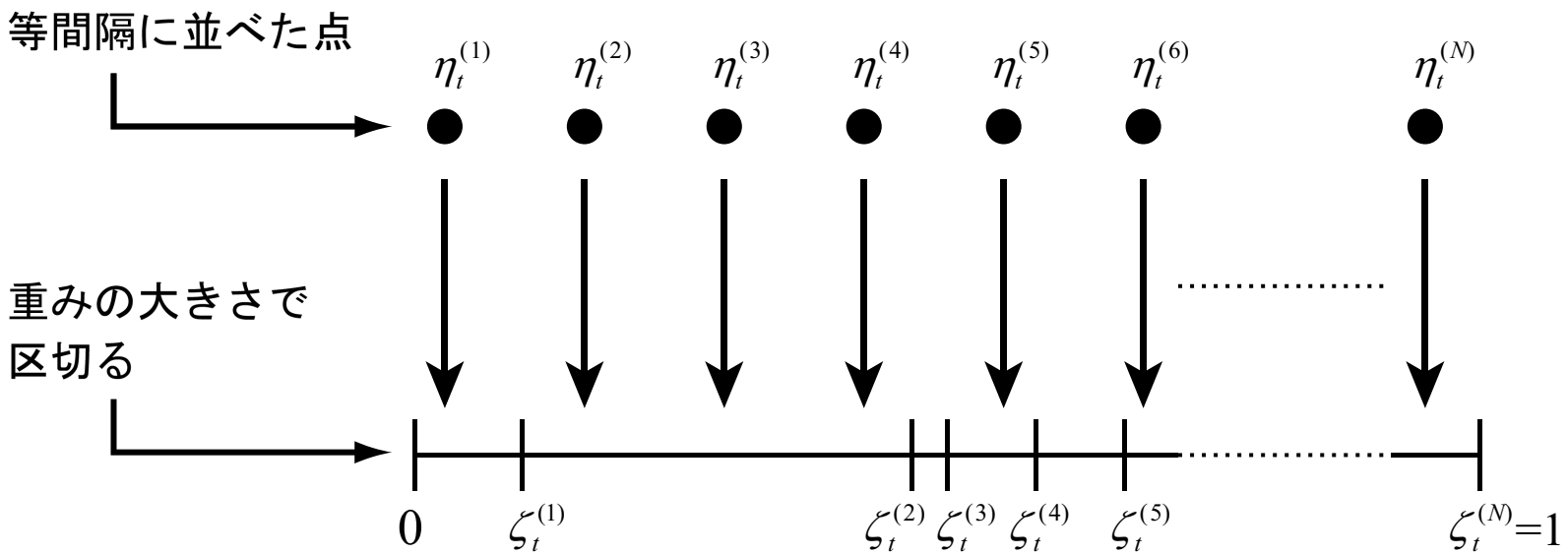
そこで，抽出される粒子数の比が尤度比になるべく近くなるようにするために，ランダム抽出を行わず，代わりに以下のような方法を取ることがある．

1. $N\beta_k^{(i)}$ を整数部分 $\bar{m}_k^{(i)}$ として，まず，各粒子 $\boldsymbol{x}_{k|k-1}^{(i)}$ をそれぞれ $\bar{m}_k^{(i)}$ 個ずつ抽出する．これで，$\sum_i \bar{m}_k^{(i)}$ 個が確定する．
2. 粒子数を N 個にするために，各粒子 $\boldsymbol{x}_{k|k-1}^{(i)}$ が $(N\beta_k^{(i)} - \bar{m}_k^{(i)})/(N - \sum_i \bar{m}_k^{(i)})$ の確率で抽出されるようなランダム抽出によって，不足分 $N - \sum_i \bar{m}_k^{(i)}$ 個の粒子を抽出する．

比例配分２

また，ほとんどまったく乱数を使わずに決める方法も一つの手段として考えられる．

1. まず，$\{\zeta_k^{(i)}\}_{i=0}^N$, $\{\eta_k^{(i)}\}_{i=1}^N$ をそれぞれ $\zeta^{(0)} = 0$, $\zeta^{(i)} = \sum_{j=1}^{i} \beta_k^{(j)}$, $\eta_k^{(i)} = (i - \varepsilon)/N$ $(0 < \varepsilon \leq 1)$ のように定義する．
2. $i = 1, \cdots, N$ について $\zeta_k^{(i-1)} \leq \eta_k^{(j)} < \zeta^{(i)}$ を満たす j を探し，$\boldsymbol{x}_{k|k}^{(j)} := \boldsymbol{x}_{k|k-1}^{(i)}$ とする．



有効粒子数

- 設定によっては縮退の問題が起こり，
  重み $\beta_k^{(i)}$ がある 1 個の粒子のみで 1 に近い値になり，残りの粒子の重みがほとんど 0 になってしまう場合がある．
- こうなると，分布の広がりが表現できず，推定がうまく行かない．
- こうした重みの偏りを調べる指標として，以下の有効粒子数と呼ばれるものがある:

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (\beta_k^{(i)})^2}. \tag{10}$$

- 有効粒子数 N_{eff} は，重みのばらつき具合によって 1 から N までの値を取る．N 個の粒子がすべて同じ重み $(= 1/N)$ なら $N_{\text{eff}} = N$ で，1 個の粒子のみが重み 1 で残りは 0 という極端に偏った場合は $N_{\text{eff}} = 1$ である．

有効粒子数

- 重みのバランスはエントロピー

$$S = -\sum_{i=1}^{N} \beta_k^{(i)} \log_e \beta_k^{(i)} \tag{11}$$

  によって調べることもできる．
- エントロピー S から

$$N_S = e^S \tag{12}$$

  のようにして，有効粒子数 (のようなもの) を定義することもできる．
- やはり，N 個の粒子がすべて同じ重み $(= 1/N)$ なら $N_S = N$，1 個の粒子のみが重み 1 で残りは 0 という場合は $N_S = 1$ となる．

有効粒子数

- 有効粒子数が 1 に近い値になった場合，確率分布がうまく表現できなくなっているので，何らかの対策をするのが望ましい．
- 有効粒子数がすぐに 1 に近くなってしまう場合によくある原因として，
  - 1 回 (ステップ) で参照するデータが多い (観測ベクトル $\boldsymbol{y}_k$ が高次元)
  - 観測の精度が高い (観測ノイズが小さい) と仮定した時

  などが挙げられる．
- これに対処するには，粒子の評価をする尤度関数 $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ の幅を広げて (観測ノイズの分散を大きく取って) 推定を行ってみるのが，一つの方法である．
- リサンプリングに伴う計算コストを抑えるため，重みのばらつきを有効粒子数で評価し，必要なときだけリサンプリングを行うという利用法もよく見られる．(ただ，本来は毎回リサンプリングを実行するのが望ましいと思われる．)

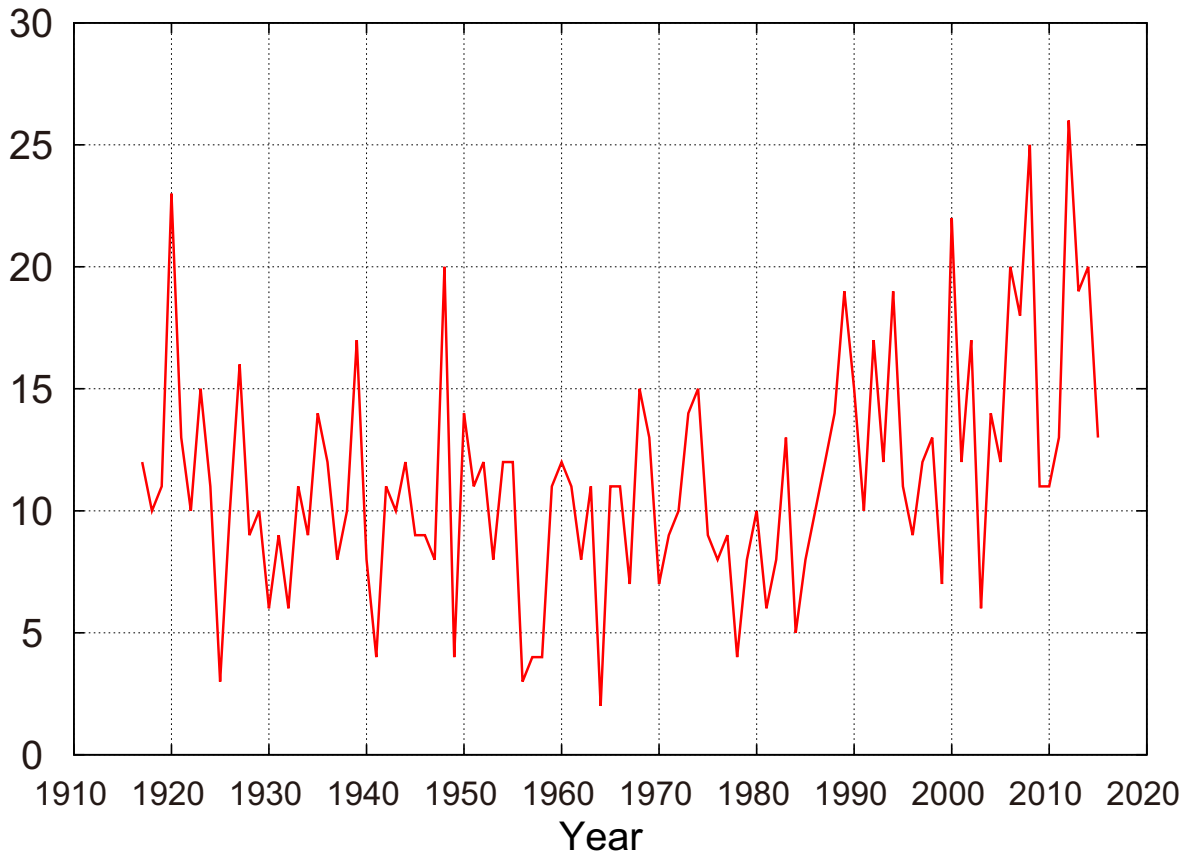
粒子フィルタ + SIS

1. $i = 1, \ldots, N$ について乱数 $\boldsymbol{x}_{0|0}^{(i)} \sim p(\boldsymbol{x}_0)$ を生成する．
2. $i = 1, \ldots, N$ について $\beta_0^{s,(i)} = 1/N$ と設定する．
3. $k = 1, \ldots, K$ について以下を実行する．
   a. 各 i $(i = 1, \ldots, N)$ について I, II を実行する．
      I. 乱数 $\boldsymbol{x}_{k|k-1}^{(i)} \sim p(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}^{(i)})$ を生成する．
      II. $l_k^{(i)} = p(\boldsymbol{y}_k | \boldsymbol{x}_{k|k-1}^{(i)})$ を計算する．
   b. $L_k = \sum_{i=1}^N \beta_{k-1}^{s,(i)} l_k^{(i)}$ を求める．
   c. 各 i $(i = 1, \ldots, N)$ について重み $\beta_k^{s,(i)} = (\beta_{k-1}^{s,(i)} l_k^{(i)})/L_k$ を求める．
   d. 有効粒子数 $N_{\text{eff}} = 1/\sum_i (\beta_k^{s,(i)})^2$ を計算する．
   e. $N_{\text{eff}} < N_{\text{th}}$ ならば以下を実行する
      - 粒子の集合 $\{\boldsymbol{x}_{k|k-1}^{(1)}, \ldots, \boldsymbol{x}_{k|k-1}^{(N)}\}$ から，各粒子 $\boldsymbol{x}_{k|k-1}^{(i)}$ が $\beta_k^{s,(i)}$ の割合で抽出されるようにして復元抽出を N 回繰り返し，$\{\boldsymbol{x}_{k|k}^{(1)}, \ldots, \boldsymbol{x}_{k|k}^{(N)}\}$ を生成する．
      - $i = 1, \ldots, N$ について $\beta_k^{s,(i)} = 1/N$ と設定する．

$*$ リサンプリングを行う閾値 N_{th} は，事前に適当に決めておく．

非線型・非ガウス状態空間モデルによる解析例

粒子フィルタの応用例として，東京の毎年の雷日数データ (気象庁) を解析する．



非線型・非ガウス状態空間モデルによる解析例

年ごとの雷が発生した日数は 0 以上の整数値しか取らない．ここでは，雷日数 y_k が x_k をパラメータとするポアソン分布

$$y_k \sim \text{Poission}(x_k) \tag{13}$$

にしたがうとものとする．$x_k > 0$ が常に成り立つ必要があるが，例えば，$\xi_k = \log x_k$ が

$$\xi_k = \xi_{k-1} + v_k \tag{14}$$

にしたがって変化するものとし，$x_k = e^{\xi_k}$ とすれば，x_k が負になることはない．

非線型・非ガウス状態空間モデルによる解析例

観測モデルを $y_k \sim \mathrm{Poission}(x_k)$ としたので，粒子フィルタの手続きの中で計算する粒子 $x^{(i)}_{k|k-1}$ の尤度は

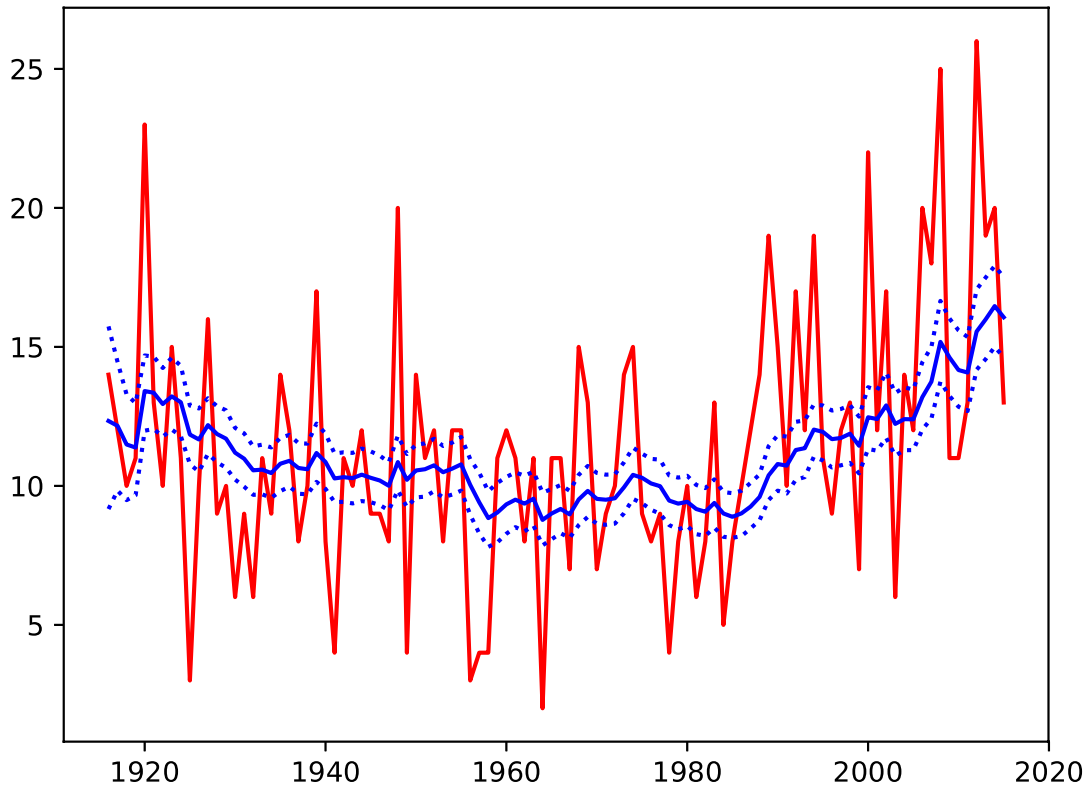
$$l^{(i)}_k = p(y_k|x^{(i)}_k) = \frac{e^{-x^{(i)}_k}(x^{(i)}_k)^{y_k}}{y_k!}. \tag{15}$$

対数を取ると

$$\log l^{(i)}_k = -x^{(i)}_k + y_k \log x^{(i)}_k - \log(y_k!). \tag{16}$$

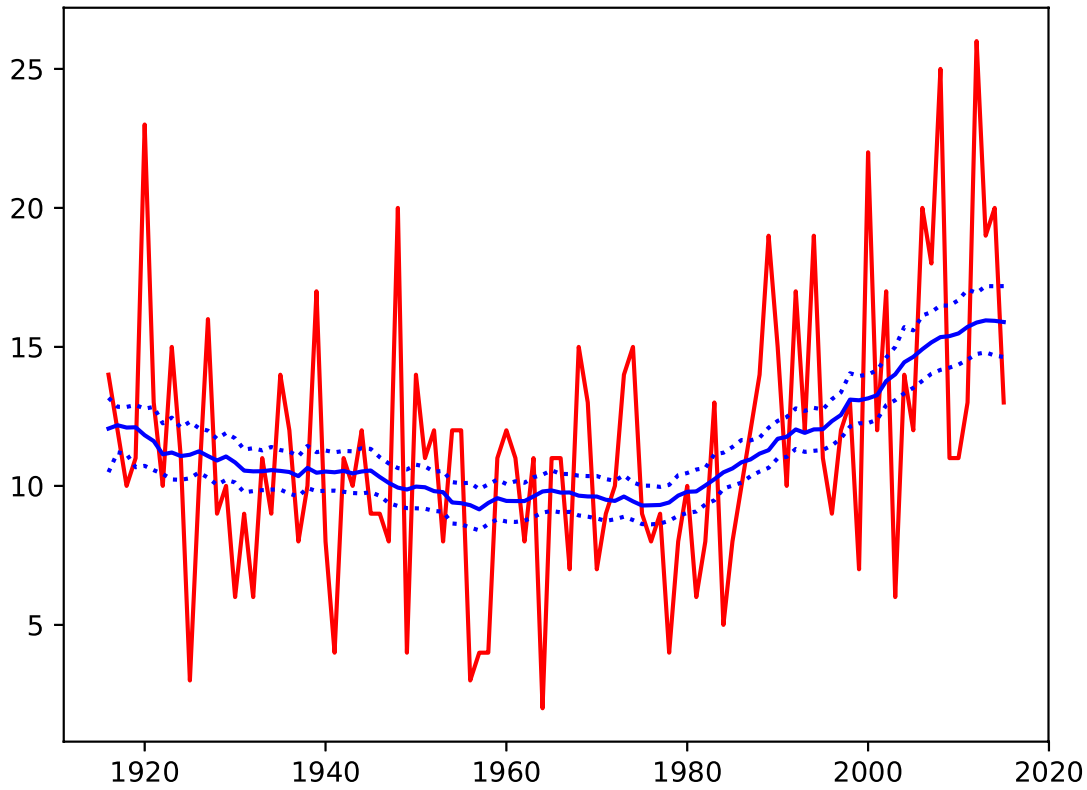
結果

フィルタ推定値 (点線は 15.9, 84.1%**点) (粒子数は** 1000)



結果

固定ラグ平滑化 (ラグ 10) による推定値 (粒子数は 1000)



課題 1

- 適当な気象時系列データ (例えば京都の各年の平均気温など) を一つ選び，以下の簡単な 1 変数状態空間モデル

$$x_k = x_{k-1} + v_k,$$

$$y_k = x_k + w_k$$

のもと，カルマンフィルタを適用し，各時点の x_k を推定せよ．$\mathbf{Q}_k$, $\mathbf{R}_k$ はデータの特徴に応じて適宜設定すること．

- 気象の時系列データは，例えば，気象庁の HP

https://www.data.jma.go.jp/obd/stats/etrn/index.php

にある．(地点などを選ぶとテーブルの形式で表示されるが，例えば，マウスで選んでコピー・アンド・ペーストすることで，Excel に取り込むことができる．

課題 2

- 先程，カルマンフィルタを適用したのと同じ気象時系列データで同じ 1 変数状態空間モデル

$$x_k = x_{k-1} + v_k,$$
$$y_k = x_k + w_k$$

のもと，アンサンブルカルマンフィルタを適用し，各時点の x_k を推定せよ．$\mathbf{Q}_k$, $\mathbf{R}_k$ はデータの特徴に応じて適宜設定すること．(N は数十程度でよいと思う．)

課題 3

- アンサンブルカルマンフィルタを実際に使ってみるためには，適当なモデルから，擬似的なデータを作り，アンサンブルカルマンフィルタで，同じモデルを使って推定できることを確認するとよい．
- ここでは，Lorenz 方程式 (Lorenz, 1963) を例として取り上げる:

$$\frac{dx}{dt} = -s(x-y)$$
$$\frac{dy}{dt} = rx - y - xz$$
$$\frac{dz}{dt} = xy - bz,$$

  但し，パラメータは，元の論文に従って $s = 10$, $r = 28$, $b = 8/3$ と設定する．
- 以下では，この Lorenz 方程式を

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{g}(\boldsymbol{x})$$

  と表す．

課題 3

手順:

1. まず，Lorenz 方程式

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{g}(\boldsymbol{x})$$

を 4 次の Runge-Kutta 法

$$\begin{aligned}
\boldsymbol{\kappa}_1 &= \boldsymbol{g}\left(\boldsymbol{x}_{k-1}\right) \\
\boldsymbol{\kappa}_2 &= \boldsymbol{g}\left(\boldsymbol{x}_{k-1} + \frac{\Delta t}{2}\boldsymbol{\kappa}_1\right) \\
\boldsymbol{\kappa}_3 &= \boldsymbol{g}\left(\boldsymbol{x}_{k-1} + \frac{\Delta t}{2}\boldsymbol{\kappa}_2\right) \\
\boldsymbol{\kappa}_4 &= \boldsymbol{g}\left(\boldsymbol{x}_{k-1} + \Delta t\boldsymbol{\kappa}_3\right) \\
\boldsymbol{x}_k &= \boldsymbol{x}_{k-1} + \frac{\Delta t}{6}(\boldsymbol{\kappa}_1 + 2\boldsymbol{\kappa}_2 + 2\boldsymbol{\kappa}_3 + \boldsymbol{\kappa}_4)
\end{aligned}$$

などで積分し，「真のシナリオ」を作る ($\Delta t = 0.01$ 程度)．

課題 3

2. 観測可能な時間間隔を設定し，真のシナリオにガウス乱数 (例えば $\mathcal{N}(\mathbf{0}, \mathbf{I})$ を加えて，擬似観測データを生成する．(例えば t が 0.1 増えるごとに観測が得られると仮定すればよい．)
3. 元の 4 次の Runge-Kutta 法による時間発展のモデルを f_k として使い，アンサンブルカルマンフィルタで $\boldsymbol{x}_k$ を推定する．$\mathbf{R} = \mathbf{I}$，$\mathbf{Q} = 0.01\mathbf{I}$ 程度に設定する．

まとめ

- 粒子フィルタは，モンテカルロ近似を用いたデータ同化手法の一つであり，非線型・非ガウスの問題も扱える汎用的な手法である．
- アンサンブルカルマンフィルタが粒子を観測に合う方向に動かすことでフィルタ分布を近似するアンサンブルを得るのに対して，粒子フィルタでは観測に合う粒子を多数複製するリサンプリングによってフィルタ分布を近似するアンサンブルを得る．
- しかし，粒子フィルタは，リサンプリングの結果，似たような粒子が増え，本来の広がりを持った確率分布を表現できなくなることがあり，それを回避するために，計算量が多くなりがちである．