Inference in Probabilistic Programming II

Xin Zhang

Peking University

Part of the content is from "An Introduction to Probabilistic Programming" by Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood
And
"An Introduction to Sequential Monte Carlo Methods" by Arnaud Doucet, Nando De Freitas, and Neil Gordon

Recap of Last Lecture

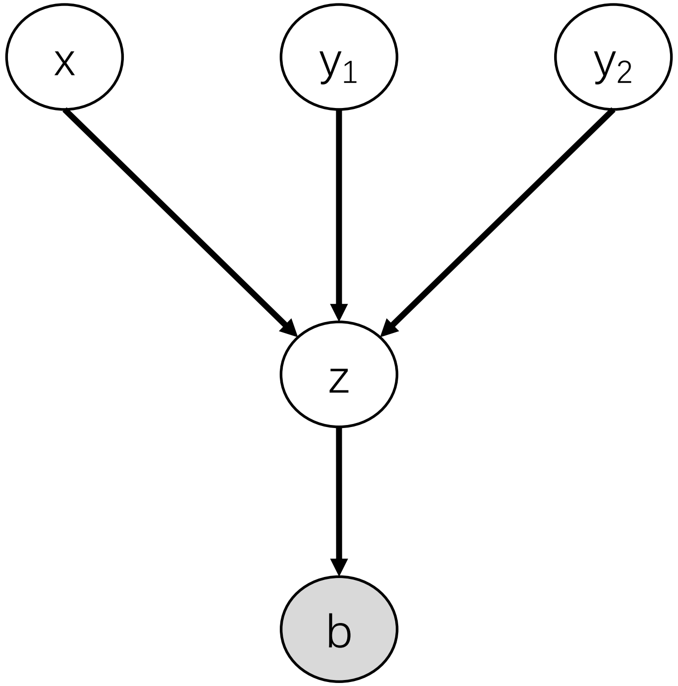
- Graph-based inference
  - Static
  - Cannot deal with programs with unbounded loops

Graph Translation: Example

```
x = bernoulli(0.2)
if(x){
        y1 = uniform(0, 2)
}
else
        y2 = gaussian(0, 5)

y3 = phi(x, y1, y2)
z = gaussian(y3, 1)

condition(z>10)
```



Inference on Translated Graphs

- Loopy belief propagation

- Sampling
  - Gibbs
  - Hamiltonian Monte Carlo

Gibbs Sampling

- Proposal distribution
  - Change one assignment at a time
  - $p(x \mid Y, X\backslash\{x\})$, where Y are observed variables

- When we cannot evaluate $p(x \mid Y, X\backslash\{x\})$, we can turn to Metropolis-Hasting while using $q(x \mid Y, X\backslash\{x\})$ as the proposal distribution

Hamiltonian Monte Carlo (HMC)

- An more scalable MCMC algorithm

$$H(\mathbf{z}, \mathbf{r}) = E(\mathbf{z}) + K(\mathbf{r})$$

Potential energy, **z** are the random variables to sample from

Kinetic energy, **r** are auxiliary variables, provides momentum

Intuition Behind HMC

- https://arogozhnikov.github.io/2016/12/19/markov_chain_monte_carlo.html

Put Things Together: HMC

- Augment distribution $p(\boldsymbol{z})$ with $p(\boldsymbol{z}, \boldsymbol{r})$
- Proposal distribution:
  - Update $\boldsymbol{z}, \boldsymbol{r}$ using Hamiltonian dynamics (in practice, a discretized approximation called leapfrog integration)
  - Judge whether to accept $\boldsymbol{z}, \boldsymbol{r}$ (see below)
  - Update $\boldsymbol{r}$ stochastically
- Acceptance probability (After applying Hamiltonian dynamics):

$$\min\left(1, \exp\{H(\mathbf{z}, \mathbf{r}) - H(\mathbf{z}^{\star}, \mathbf{r}^{\star})\}\right)$$

Account for approximation

The Leapfrog Approximation

$$\begin{aligned}
\widehat{r}_i(\tau + \epsilon/2) &= \widehat{r}_i(\tau) - \frac{\epsilon}{2}\frac{\partial E}{\partial z_i}(\widehat{\mathbf{z}}(\tau)) \\
\widehat{z}_i(\tau + \epsilon) &= \widehat{z}_i(\tau) + \epsilon\widehat{r}_i(\tau + \epsilon/2) \\
\widehat{r}_i(\tau + \epsilon) &= \widehat{r}_i(\tau + \epsilon/2) - \frac{\epsilon}{2}\frac{\partial E}{\partial z_i}(\widehat{\mathbf{z}}(\tau + \epsilon)).
\end{aligned}$$

To remove biases introduced by numerical errors,
the steps are sampled from ϵ and $-\epsilon$

Question 1: Is the statement right?

- For any given probabilistic program with loops, it cannot be converted into a graphical model

Question 2: Is the statement right?

- The graph obtained by translating a probabilistic program is always a tree

Question 3: Translate the program into a graph

```
x = guassian(0, 1)
y = uniform(0, x)
if (x>10){
        z = x
        condition(y >1.5)
}
else{
        condition(y<0.5)
        z = y
}
w = gaussian(z, 0)
```

Question 4: Is the statement right?

- Gibbs sampling can be applied to sample any distribution

Question 5: Is the statement right?

- In HMC, the gradient is the gradient of the density function of the target distribution

Question 6: Is the statement right?

- HMC cannot be applied to any probabilistic programs with branches

This Lecture

- Evaluation-based inference

- More sampling algorithms

Motivation

- The number of random variables is unknown at compile time
  - Introduce an upper bound on the number of variables
  - Implement inference methods that dynamically instantiate variables

Likelihood Weighting

- A form of importance sampling where the proposal is the prior

$$\mathbb{E}_{q(X)}\left[\frac{p(X|Y)}{q(X)}r(X)\right] = \frac{1}{p(Y)}\mathbb{E}_{q(X)}\left[\frac{p(Y,X)}{q(X)}r(X)\right]$$

$$\simeq \frac{1}{p(Y)}\frac{1}{L}\sum_{l=1}^{L} W^l r(X^l),$$

$$W^l = \frac{p(Y,X^l)}{q(X^l)} = \frac{p(Y|X^l)p(X^l)}{p(X^l)} = p(Y|X^l)$$

If we use $p(X^l)$ as the proposal distribution

Y are observed/conditioned variables

Likelihood Weighting

- But wait, every run of the program only evaluates a subset of all variables!

- It is OK: $r(X)$ is the return value projection of all variables X

Likelihood Weighting

- What happens if there are no factor statements but only condition statements in the program?
- How to implement it in a graph-based inference?

Likelihood Weighting: Evaluation-based Implementation

- Run the program to draw samples

- Update the weight W while running the program
  - Initially, $\log W = 0$

  - Whenever encounter an expression $condition(b)$, update $\log W \leftarrow logW + logp_b(true)$

Metropolis-Hasting

- Similar problem: each execution only evaluates a subset of variables

- Naïve method: use the prior distribution p(X) as the proposal distribution:

$$\alpha = \frac{P(X'|Y)q(X|X')}{P(X|Y)q(X'|X)} = \frac{P(X',Y)q(X|X')}{P(X,Y)q(X'|X)} = \frac{P(Y|X')}{P(Y|X)}$$

Metropolis-Hasting: Single-Site Proposals

- Most commonly used evaluation-based proposal

- Try to only change the value of a one variable at a time
  - Not always possible due to dependencies

Metropolis-Hasting: Single-Site Proposals

- Map $\sigma(X)$, such that $X(x)$ refers to the value of x (only variables in the current execution)

- Map $\sigma(logP)$, where $logP(v)$ evaluates the density for each variable
  - When sampling from a distribution d, we have
    $$\sigma\big(logP(x)\big) = LOG - PROB(d, X(x))$$
  - When encounter $condition(b)$, we have
    $$\sigma\big(logP(y)\big) = LOG - PROB(b, true)$$

Metropolis-Hasting: Single-Site Proposals

- Pick a variable $x_0 \in dom(X)$ at a random from the current sample

- Construct a proposal X', P' by re-running the program
  - For an expression d that sample from a variable x
    - If $x == x_0$, or $x \notin dom(X)$, then samples from the expression. Otherwise, reuse the value $X'(x) \leftarrow X(x)$
    - Calculate the probability $P'(x) \leftarrow PROB(d, X'(x))$
  - For expression $condition(b)$ with variable y:
    - Calculate the probability $P'(y) \leftarrow PROB(b, y) = 1_{[b==y]}$
  - For expression $observe(e, v)$ with variable y:
    - Calculate the probability $P'(y) \leftarrow PROB(e, v)$

Metropolis-Hasting: Single-Site Proposals

$$\alpha = \frac{p(Y', X')q(X|X')}{p(Y, X)q(X'|X)}$$

$$= \frac{p(Y', X')}{q(X'|X, x_0)} \frac{q(X|X', x_0)}{p(Y, X)} \frac{q(x_0|X')}{q(x_0|X)}.$$

Metropolis-Hasting: Single-Site Proposals

$$\frac{p(Y', X')}{q(X'|X, x_0)} \frac{q(X|X', x_0)}{p(Y, X)} \frac{q(x_0|X')}{q(x_0|X)}$$

$$\frac{q(x_0|X')}{q(x_0|X)} = \frac{|X|}{|X'|}.$$

$$p(Y', X') = p(Y'|X')p(X') = \prod_{y \in Y'} \mathcal{P}'(y) \prod_{x \in X'} \mathcal{P}'(x)$$

$$q(X'|X, x_0) = \prod_{x \in X'^{\text{sampled}}} \mathcal{P}'(x).$$

$$\frac{p(Y', X')}{q(X'|X, x_0)} = \prod_{y \in Y'} \mathcal{P}'(y) \prod_{x \in X'^{\text{reused}}} \mathcal{P}'(x)$$

We divide a sample into sampled part and reused part

$$\frac{p(Y, X)}{q(X|X, x_0)} = \prod_{y \in \mathcal{Y}} \mathcal{P}(y) \prod_{x \in X^{\text{reused}}} \mathcal{P}(x).$$

Metropolis-Hasting: Single-Site Proposals

$$\alpha = \frac{|\mathrm{dom}(\mathcal{X})|}{|\mathrm{dom}(\mathcal{X}')|} \frac{\prod_{y \in \mathcal{Y}} \mathcal{P}'(y) \prod_{x \in X'^{\mathrm{reused}}} \mathcal{P}'(x)}{\prod_{y \in \mathcal{Y}} \mathcal{P}(y) \prod_{x \in X^{\mathrm{reused}}} \mathcal{P}(x)}$$

Example

```
x = 0

while(bernoulli(0.5){
        x+= uniform(0,1)
}

condition(x >= 10)
```

Sequential Monte Carlo

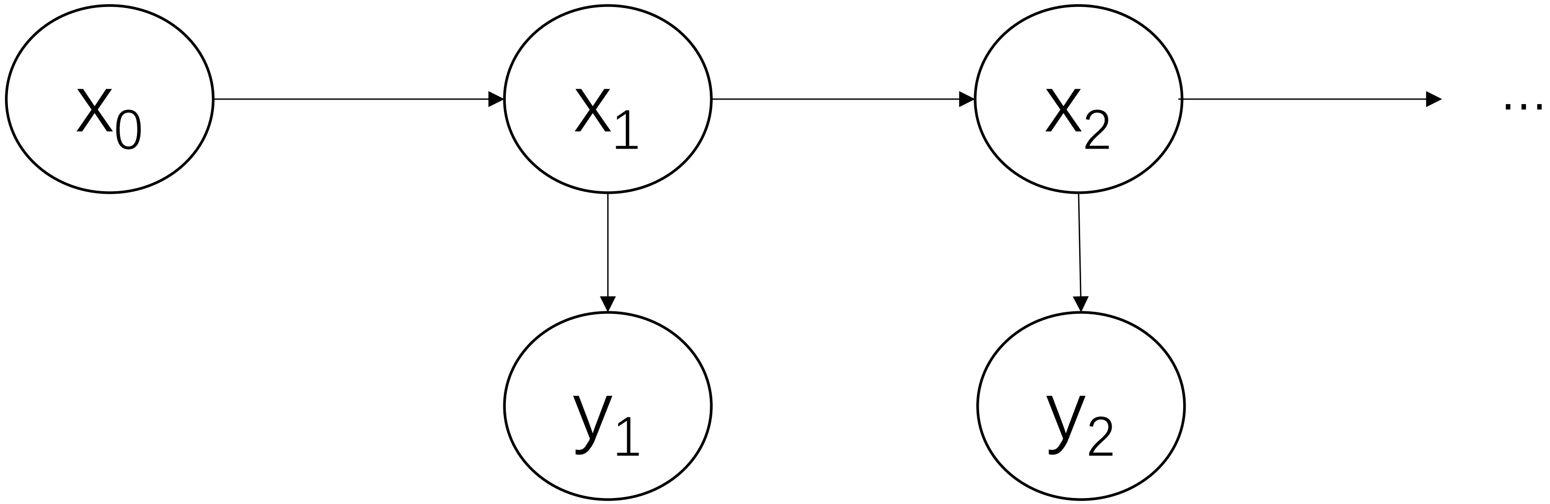
- Problem with likelihood weighting algorithm:
  - Essentially a "guess-and-check"
  - Doesn't work well with models where there are a lot of random variables

- Sequential Monte Carlo
  - In probabilistic programming, sample a high-dimensional distribution by sampling a sequence of lower dimensional distributions
  - Also called particle filters
  - Used in signal processing and probabilistic inference

Informal Example

- See the example by Andreas Svensson
  - https://www.bilibili.com/video/BV1XE41177D1?share_source=copy_web
  - https://www.youtube.com/watch?v=aUkBa1zMKv4

SMC: Problem Statement



Given

$p(x_0)$ and
$p(x_t|x_{t-1})$ and
$p(y_t|x_t)$ and
Observations $y_{1:t}$

Estimate

$p(x_{0:t}|y_{1:t})$ or
$p(x_t|y_{1:t})$ or
$I(f_t) = E_{p(x_{0:t}|y_{1:t})}[f_t(x_{0:t})] = \int f_t(x_{0:t})p(x_{0:t}|y_{1:t})dx_{0:t}$

SMC: Problem Analysis

Can you compute these expressions?

$$p\left(\mathbf{x}_{0:t} \middle| \mathbf{y}_{1:t}\right) = \frac{p\left(\mathbf{y}_{1:t} \middle| \mathbf{x}_{0:t}\right) p\left(\mathbf{x}_{0:t}\right)}{\int p\left(\mathbf{y}_{1:t} \middle| \mathbf{x}_{0:t}\right) p\left(\mathbf{x}_{0:t}\right) d\mathbf{x}_{0:t}}.$$

$$p\left(\mathbf{x}_{0:t+1} \middle| \mathbf{y}_{1:t+1}\right) = p\left(\mathbf{x}_{0:t} \middle| \mathbf{y}_{1:t}\right) \frac{p\left(\mathbf{y}_{t+1} \middle| \mathbf{x}_{t+1}\right) p\left(\mathbf{x}_{t+1} \middle| \mathbf{x}_{t}\right)}{p\left(\mathbf{y}_{t+1} \middle| \mathbf{y}_{1:t}\right)}.$$

$$\textit{Prediction: } p\left(\mathbf{x}_{t} \middle| \mathbf{y}_{1:t-1}\right) = \int p\left(\mathbf{x}_{t} \middle| \mathbf{x}_{t-1}\right) p\left(\mathbf{x}_{t-1} \middle| \mathbf{y}_{1:t-1}\right) d\mathbf{x}_{t-1};$$

$$\textit{Updating: } p\left(\mathbf{x}_{t} \middle| \mathbf{y}_{1:t}\right) = \frac{p\left(\mathbf{y}_{t} \middle| \mathbf{x}_{t}\right) p\left(\mathbf{x}_{t} \middle| \mathbf{y}_{1:t-1}\right)}{\int p\left(\mathbf{y}_{t} \middle| \mathbf{x}_{t}\right) p\left(\mathbf{x}_{t} \middle| \mathbf{y}_{1:t-1}\right) d\mathbf{x}_{t}}.$$

SMC: Problem Analysis

- Evaluation of complex high-dimensional integrals is hard

- People turn to approximate methods such as sampling

SMC: Approach

- Use samples to deal with integrations

- Effective method that leverages importance sampling

SMC: Naïve Importance Sampling

- Let the proposal distribution be $\pi(x_{0:t}|y_{1:t})$, then we have

$$I(f_t) = \frac{\int f_t(\mathbf{x}_{0:t})\, w(\mathbf{x}_{0:t})\, \pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})\, d\mathbf{x}_{0:t}}{\int w(\mathbf{x}_{0:t})\, \pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})\, d\mathbf{x}_{0:t}}, \quad w(\mathbf{x}_{0:t}) = \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}.$$

$$\widehat{I}_N(f_t) = \frac{\frac{1}{N}\sum_{i=1}^N f_t\left(\mathbf{x}_{0:t}^{(i)}\right) w\left(\mathbf{x}_{0:t}^{(i)}\right)}{\frac{1}{N}\sum_{j=1}^N w\left(\mathbf{x}_{0:t}^{(i)}\right)} = \sum_{i=1}^N f_t\left(\mathbf{x}_{0:t}^{(i)}\right) \widetilde{w}_t^{(i)}, \quad \widetilde{w}_t^{(i)} = \frac{w\left(\mathbf{x}_{0:t}^{(i)}\right)}{\sum_{j=1}^N w\left(\mathbf{x}_{0:t}^{(j)}\right)}.$$

A sample $x_{0:t}$ is called a particle

SMC: Naïve Importance Sampling

- Problem
  - Cannot be used for recursive estimation
  - One needs to get all $y_{1:t}$ before estimating $p(x_{0:t}|y_{1:t})$
  - Need to re-evaluate whenever there is a new y
  - Does not scale

SMC: Sequential Importance Sampling

- If we want to do recursive evaluation, the proposal distribution needs to satisfy

$$\pi\left(\mathbf{x}_{0:t}\middle|\,\mathbf{y}_{1:t}\right) = \pi\left(\mathbf{x}_{0:t-1}\middle|\,\mathbf{y}_{1:t-1}\right)\pi\left(\mathbf{x}_{t}\middle|\,\mathbf{x}_{0:t-1},\mathbf{y}_{1:t}\right).$$

- Which indicates

$$\pi\left(\mathbf{x}_{0:t}\middle|\,\mathbf{y}_{1:t}\right) = \pi\left(\mathbf{x}_{0}\right)\prod_{k=1}^{t}\pi\left(\mathbf{x}_{k}\middle|\,\mathbf{x}_{0:k-1},\mathbf{y}_{1:k}\right).$$

SMC: Sequential Importance Sampling

- Then we have

$$\widetilde{w}_t^{(i)} \propto \widetilde{w}_{t-1}^{(i)} \frac{p\left(\mathbf{y}_t \middle| \mathbf{x}_t^{(i)}\right) p\left(\mathbf{x}_t^{(i)} \middle| \mathbf{x}_{t-1}^{(i)}\right)}{\pi\left(\mathbf{x}_t^{(i)} \middle| \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t}\right)}.$$

- Important case

$$\pi\left(\mathbf{x}_{0:t} \middle| \mathbf{y}_{1:t}\right) = p\left(\mathbf{x}_{0:t}\right) = p\left(\mathbf{x}_0\right) \prod_{k=1}^{t} p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}\right).$$

How to Derive the Formula

$$\widetilde{w}_t^{(i)} \propto \widetilde{w}_{t-1}^{(i)} \frac{p\left(\mathbf{y}_t \middle| \mathbf{x}_t^{(i)}\right) p\left(\mathbf{x}_t^{(i)} \middle| \mathbf{x}_{t-1}^{(i)}\right)}{\pi\left(\mathbf{x}_t^{(i)} \middle| \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t}\right)}.$$

Given

$$\widetilde{w}_t^{(i)} = \frac{w\left(\mathbf{x}_{0:t}^{(i)}\right)}{\sum_{j=1}^{N} w\left(\mathbf{x}_{0:t}^{(j)}\right)}. \quad w\left(\mathbf{x}_{0:t}\right) = \frac{p\left(\mathbf{x}_{0:t} \middle| \mathbf{y}_{1:t}\right)}{\pi\left(\mathbf{x}_{0:t} \middle| \mathbf{y}_{1:t}\right)}. \quad \pi\left(\mathbf{x}_{0:t} \middle| \mathbf{y}_{1:t}\right) = \pi\left(\mathbf{x}_{0:t-1} \middle| \mathbf{y}_{1:t-1}\right) \pi\left(\mathbf{x}_t \middle| \mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}\right).$$

$$p\left(\mathbf{x}_{0:t+1} \middle| \mathbf{y}_{1:t+1}\right) = p\left(\mathbf{x}_{0:t} \middle| \mathbf{y}_{1:t}\right) \frac{p\left(\mathbf{y}_{t+1} \middle| \mathbf{x}_{t+1}\right) p\left(\mathbf{x}_{t+1} \middle| \mathbf{x}_t\right)}{p\left(\mathbf{y}_{t+1} \middle| \mathbf{y}_{1:t}\right)}.$$

We have

$$\omega(x_{0:t}) = \frac{p(x_{0:t}|y_{1:t})}{\pi(x_{0:t}|y_{1:t})} = \frac{p(x_{0:t-1}, |y_{1:t-1}) * p(y_t|x_t) * p(x_t|x_{t-1})/p(y_t|y_{1:t-1})}{\pi(x_{0:t-1}|y_{1:t-1})\pi(x_{\mathrm{t}}|y_{1:t-1}, y_{1:t})}$$

$$= \omega(x_{0:t-1}) * \frac{p(y_t|x_t) * p(x_t|x_{t-1})}{\pi(x_{\mathrm{t}}|y_{1:t-1}, y_{1:t})} * \frac{1}{p(y_{t+1}|y_{1:t})}$$

SMC: Sequential Importance Sampling

- Problem: as t increases, importance weights $\widetilde{\omega_t}^{(i)}$ becomes more and more skewed
  - Almost all weights will become 0 except 1

- Solution: the bootstrap filter

SMC: Bootstrap Filter

- Key idea: remove particles with low weights and keep particles with high weights

- Formally replace

$$\widehat{P}_N\left(d\mathbf{x}_{0:t}\middle|\,\mathbf{y}_{1:t}\right) = \sum\nolimits_{i=1}^{N} \widetilde{w}_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}\left(d\mathbf{x}_{0:t}\right)$$

δ is the Dirac measure

- with

$$P_N\left(d\mathbf{x}_{0:t}\middle|\,\mathbf{y}_{1:t}\right) = \frac{1}{N}\sum_{i=1}^{N} N_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}\left(d\mathbf{x}_{0:t}\right),$$

SMC: Bootstrap Filter

$$P_N\left(d\mathbf{x}_{0:t} \middle| \mathbf{y}_{1:t}\right) = \frac{1}{N}\sum_{i=1}^{N} N_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}\left(d\mathbf{x}_{0:t}\right),$$

- $\sum_{i=1}^{N} N_t^{(i)} = 0$, if $N_t^{(j)} = 0$, then the particle $x_{0:t}^{j}$ dies
- How to select $N_t^{(i)}$?
  - Many methods
  - The most popular method: sampling N times from $\widehat{P}_N\left(d\mathbf{x}_{0:t} \middle| \mathbf{y}_{1:t}\right)$

SMC: Bootstrap Filter

Assume the proposal distribution is $p(x_{1:t})$

1. Initialization. T = 0
   - For i = 1,…,N, sample $x_0^{(i)} \sim p(x_0)$ and set $t = 1$
2. Importance sampling step.
   - For sample $\tilde{x}_t^{(i)} \sim p(x_t|\tilde{x}_{t-1}^{(i)})$ and set $(\tilde{x}_{0:t-1}^{(i)}, \tilde{x}_t^{(i)})$.
   - For i = 1,…,N, evaluate the importance weights.
   - Normalize the importance weights
3. Selection step
   - Resample with replacement N particles from the current particles according to importance weights
   - Set $t \rightarrow t + 1$

More on Bootstrap Filter

- Compared to sequential importance sampling, it basically
  - Allows more variations under the prefixes with high weights
  - Throws away prefixes with low weights

- Advantages:
  - Easy to implement
  - Efficient
  - Modular
  - Can be parallelized
  - Can be used for complex models

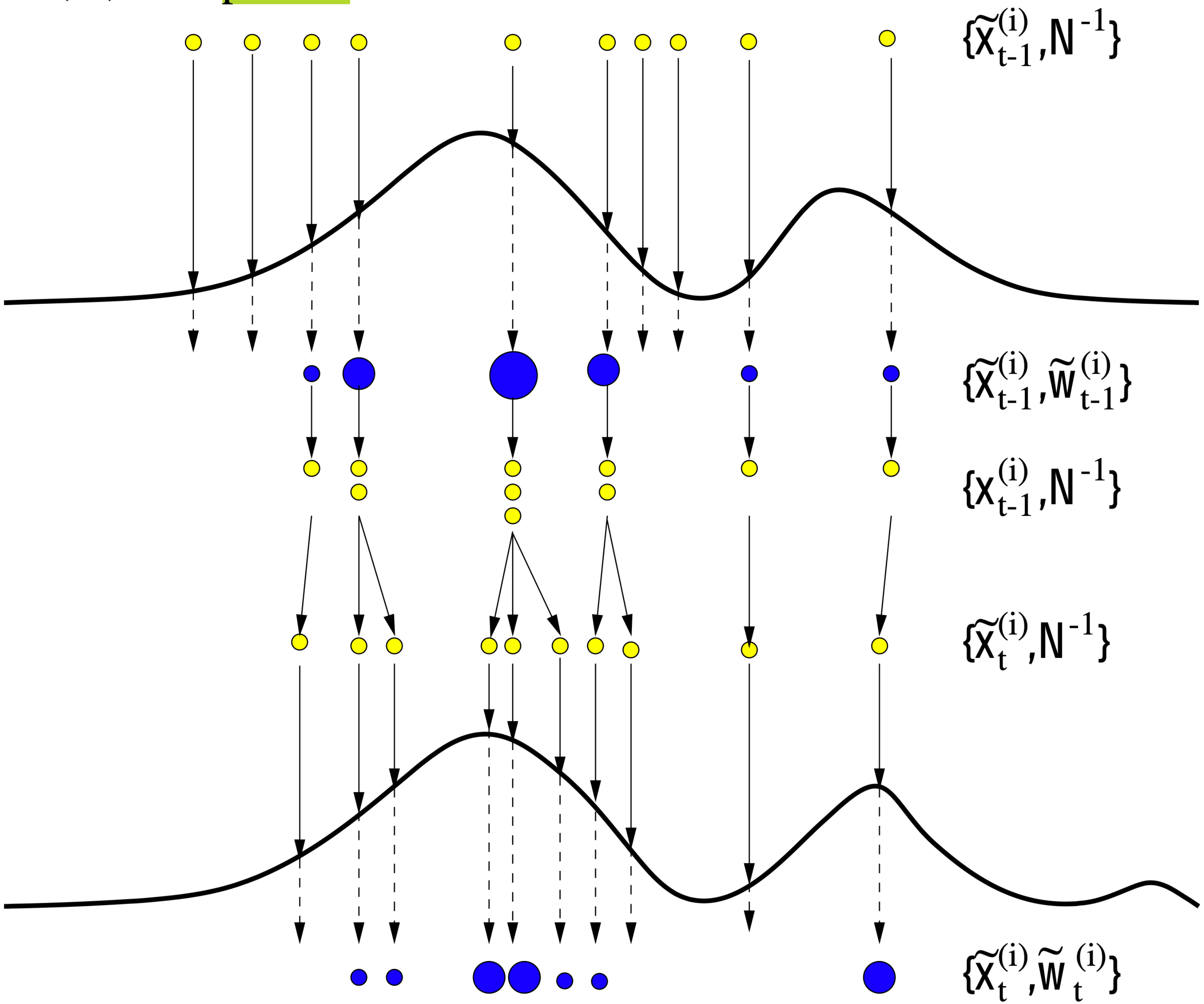
Bootstrap Filter: Example

$$
\begin{aligned}
x_t &= \frac{1}{2}x_{t-1} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2t) + v_t \\
y_t &= \frac{x_t^2}{20} + w_t,
\end{aligned}
$$

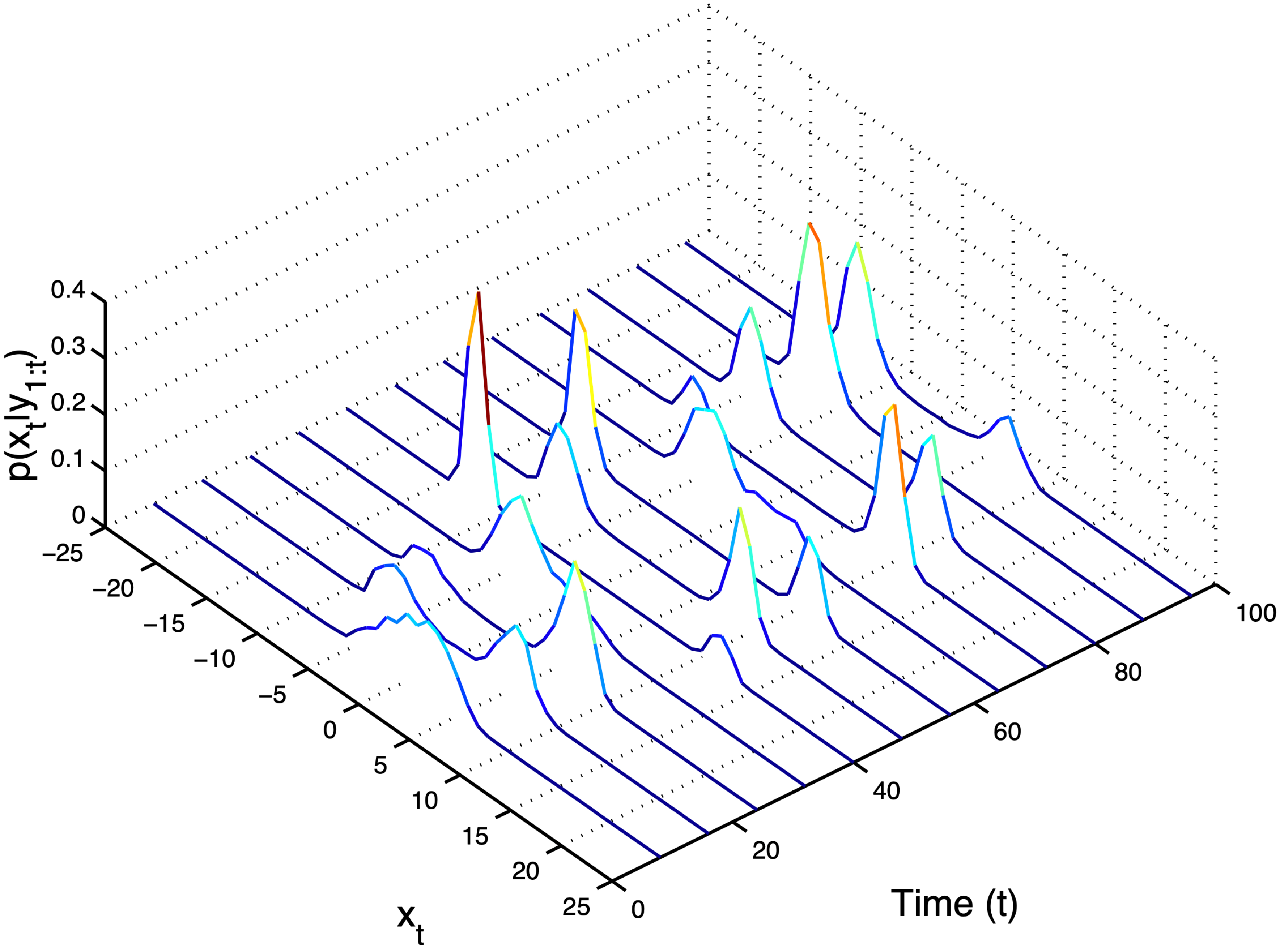
$$x_1 \sim N(0,10), v_k \sim N(0,10), w_k \sim N(0,1)$$

From "An Introduction to Sequential Monte Carlo Methods" by Arnaud Doucet, Nando De Freitas, and Neil Gordon

i=1,...,N=10 particles

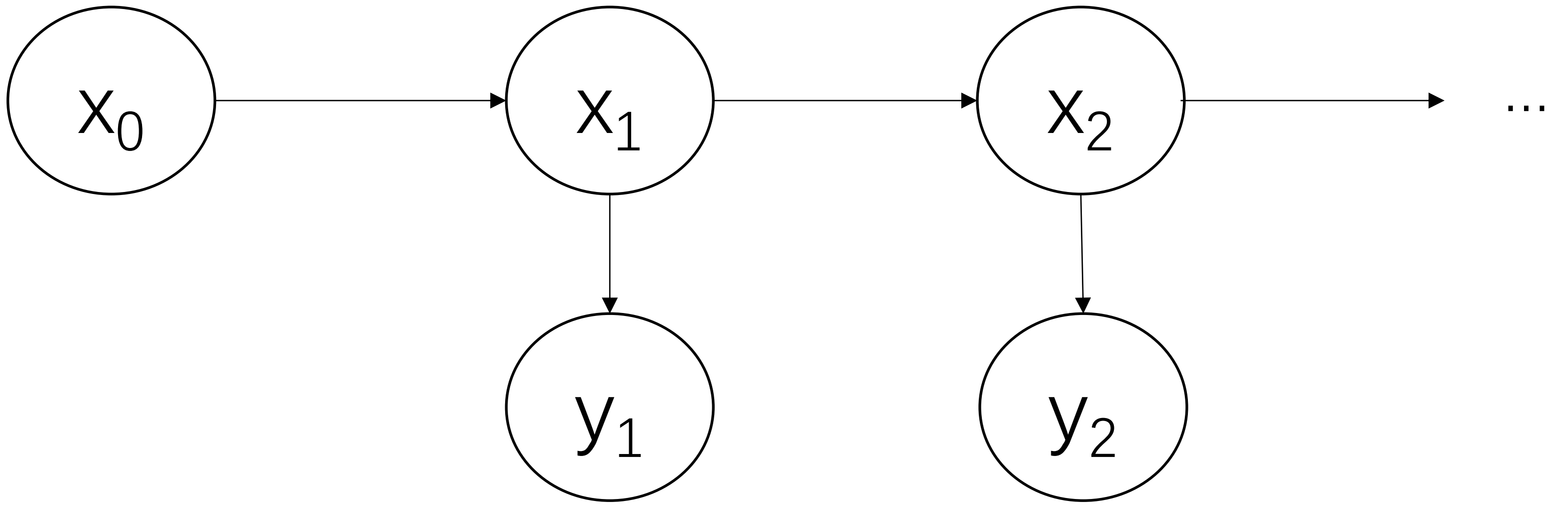


From "An Introduction to Sequential Monte Carlo Methods" by Arnaud Doucet, Nando De Freitas, and Neil Gordon



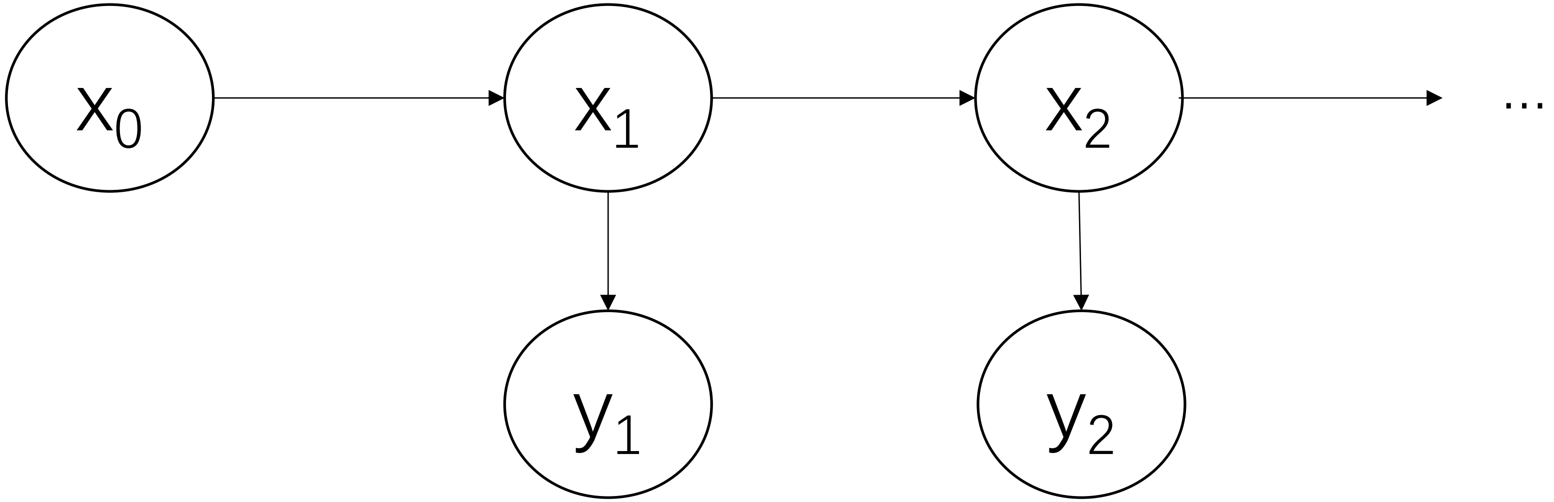
From "An Introduction to Sequential Monte Carlo Methods" by Arnaud Doucet, Nando De Freitas, and Neil Gordon

SMC in Probabilistic Programming



SMC in Probabilistic Programming

x's are the program trace excluding conditions



y's are conditions

SMC in Probabilistic Programming

- We can evaluate intermediate densities using breakpoints

- We can use the prior distribution as the proposal distribution

More on Inference in Probabilistic Programming

- There are other general methods
  - Variational Inference

- No silver bullet
  - The general problem is a counting problem
  - Some researchers are exploring programmable inference frameworks: Gen: a general-purpose probabilistic programming system with programmable inference. Cusumano-Towner, M. F.; Saad, F. A.; Lew, A. K.; and Mansinghka, V. K. In PLDI 2019:

More on Inference in Probabilistic Programming

- Implementation issues
  - How can we avoid re-running programs
    - Fork at sampling statements and conditions
    - Can be Implemented through program transformation

- For a comprehensive understanding, read http://dippl.org/chapters/03-enumeration.html

Next Lecture

- Learning in probabilistic programming