Learning to plan: Applications of search to robotics

Kevin Xie* and Homanga Bharadhwaj*

*1st year Msc. students in Computer Science

Probabilistic Planning via Sequential Monte Carlo

Model-based RL method

Control as Inference heuristic

Sequential Monte Carlo action sampling

Sequential Monte Carlo Tutorial

A method for sampling from sequential distributions.

"Perfect" Monte Carlo (MC)

Integral intractable: $$\mathbb{E}_{p(x)}[f(x)] = \int f(x)p(x)dx$$

But can sample easily. -> Approximate p(x) with N samples from p(x):

Empirical Measure

$$P_N(dx) = \sum_{i=1}^{N} \left(\frac{1}{N}\right) \delta_{x^i}(dx^i) \qquad x^i \sim p(x)$$

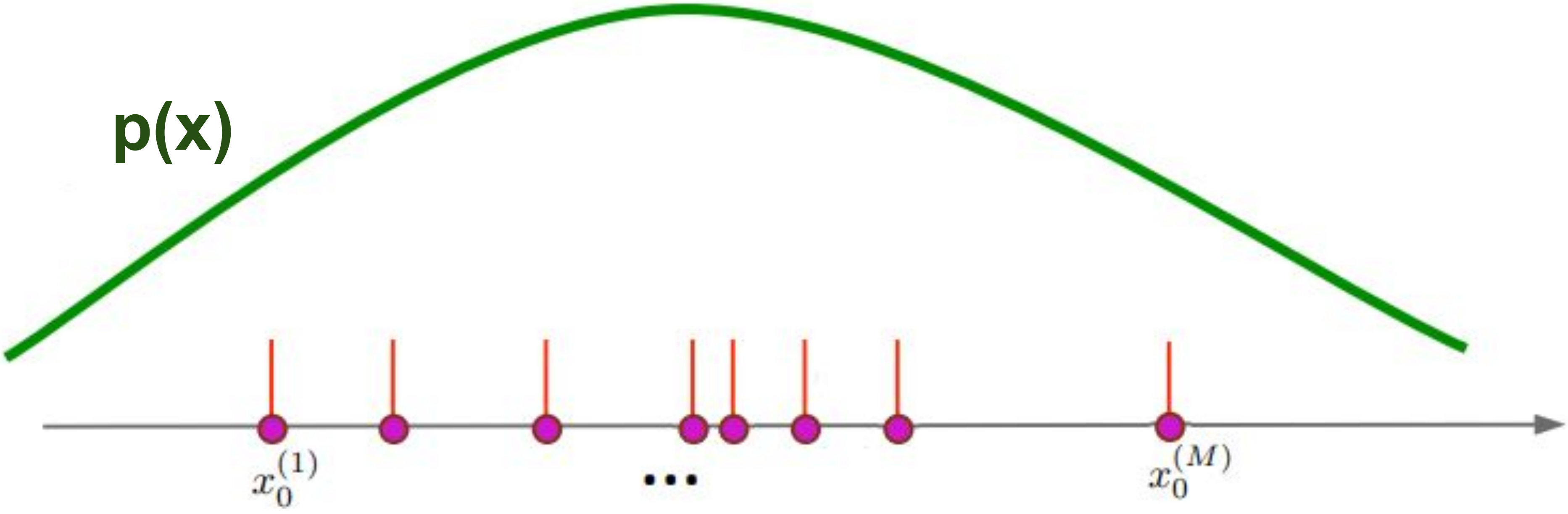
MC Estimate

$$\mathbb{E}_{p(x)}[f(x)] \approx I_N = \int P_N(x)f(x) = \frac{1}{N}\sum_{i=1}^{N} f(x^i) \qquad x^i \sim p(x)$$

https://www.stats.ox.ac.uk/~doucet/doucet_defreitas_gordon_smcbookintro.pdf [1.3.1]

"Perfect" Monte Carlo (MC)

$$P_N(dx) = \sum_{i=1}^{N} \left(\frac{1}{\underline{N}} \right) \underline{\delta_{x^i}(dx^i)} \qquad x^i \sim \underline{p(x)}$$



Importance Sampling (IS)

Integral intractable and **can't sample easily**.

But can sample from q(x). -> Approximate p(x) with N **samples from q(x)**.

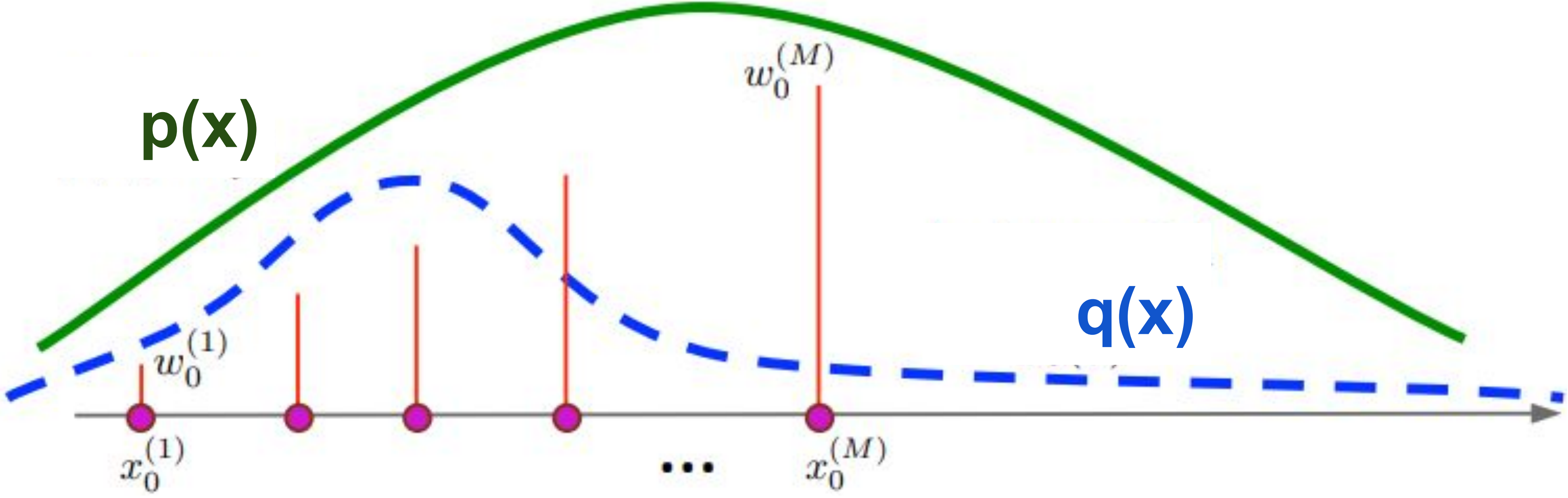
$$\mathbb{E}_{p(x)}[f(x)] = \int p(x) f(x) dx = \int q(x) \left(\frac{p(x)}{q(x)} \right) f(x) dx = \mathbb{E}_{q(x)}[w(x) f(x)]$$

$$\widehat{P}_N(dx) = \sum_{i=1}^{N} \left(\frac{w^i}{N} \right) \delta_{x^i}(dx^i) \qquad x^i \sim q(x) \qquad w^i = \frac{p(x^i)}{q(x^i)}$$

https://www.stats.ox.ac.uk/~doucet/doucet_defreitas_gordon_smcbookintro.pdf [1.3.2]

Importance Sampling

$$\widehat{P}_N(dx) = \sum_{i=1}^{N} \left(\frac{w^i}{\underline{N}}\right) \underline{\delta_{x^i}(dx^i)} \quad x^i \sim \underline{q(x)} \quad w^i = \frac{\underline{p}(x^i)}{q(x^i)}$$



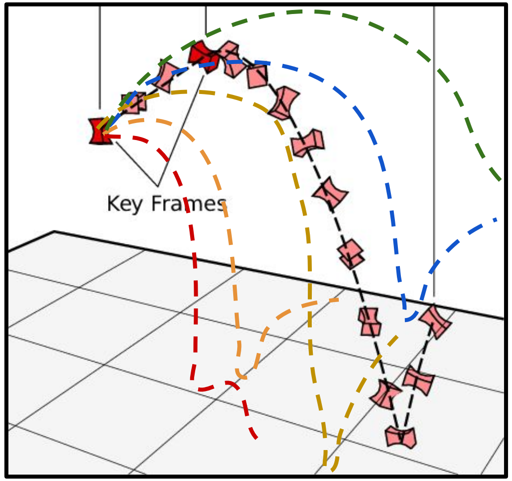
Sequential Monte Carlo (SMC)

Want to sample sequence: $x_{1:t} = \{x_j | j \in [1, t]\}$

From: $$p(x_{1:t}) = p(x_1) \prod_{j=2}^{t} p(x_j | x_{1:j-1})$$

Initial Distribution

Step



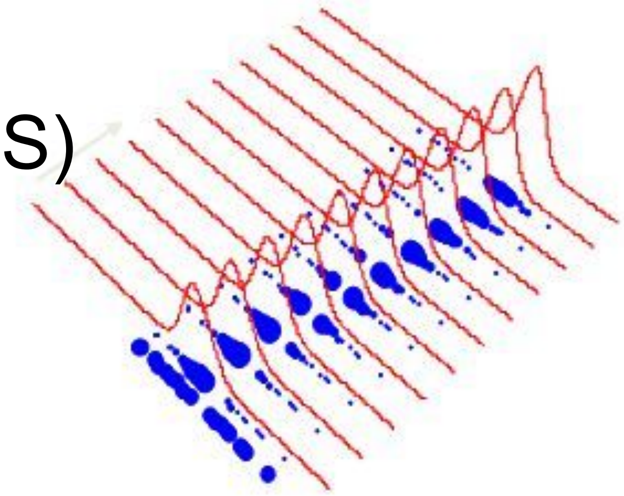
Sequential Importance Sampling (SIS)

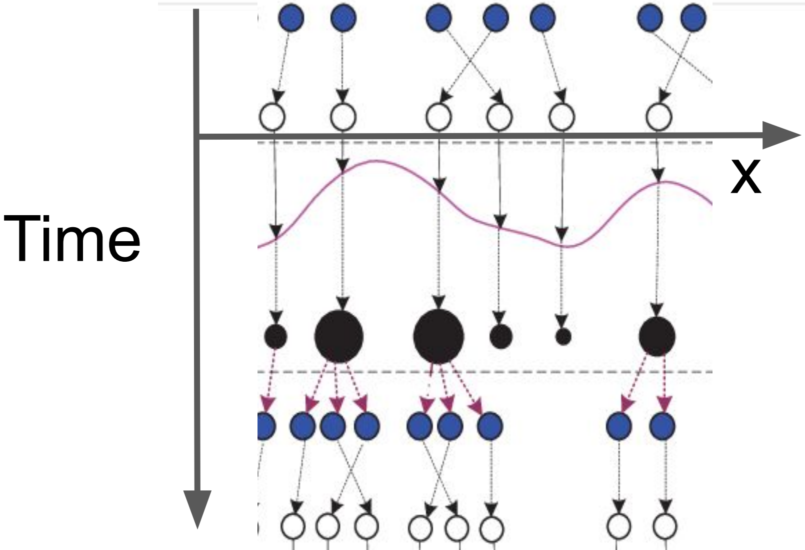
Sample from a proposal distribution:

$$q(x_{1:t}) = q(x_1) \prod_{j=2}^{t} q(x_j | x_{1:j-1})$$

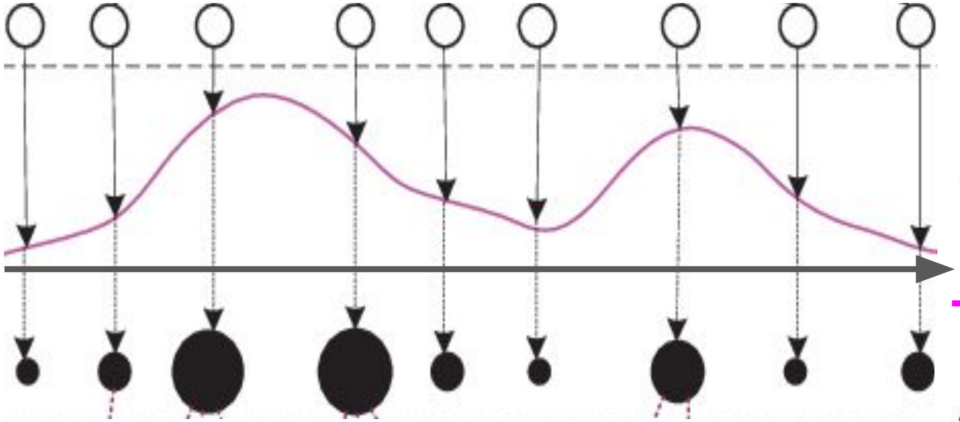
Initial Distribution

Step





t=1



$x^i \sim q(x)$ Proposal Particles

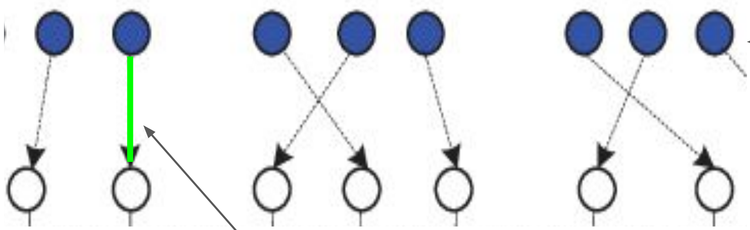
$\dfrac{p(x^i)}{\underline{q(x^i)}}$ Standard Importance Sampling

$\underline{w^i} = \dfrac{p(x^i)}{q(x^i)}$ **Time 1 Particles**

$$\widehat{P}_N(dx) = \sum_{i=1}^{N} \left(\frac{w^i}{N} \right) \delta_{x^i}(dx^i)$$

t-1

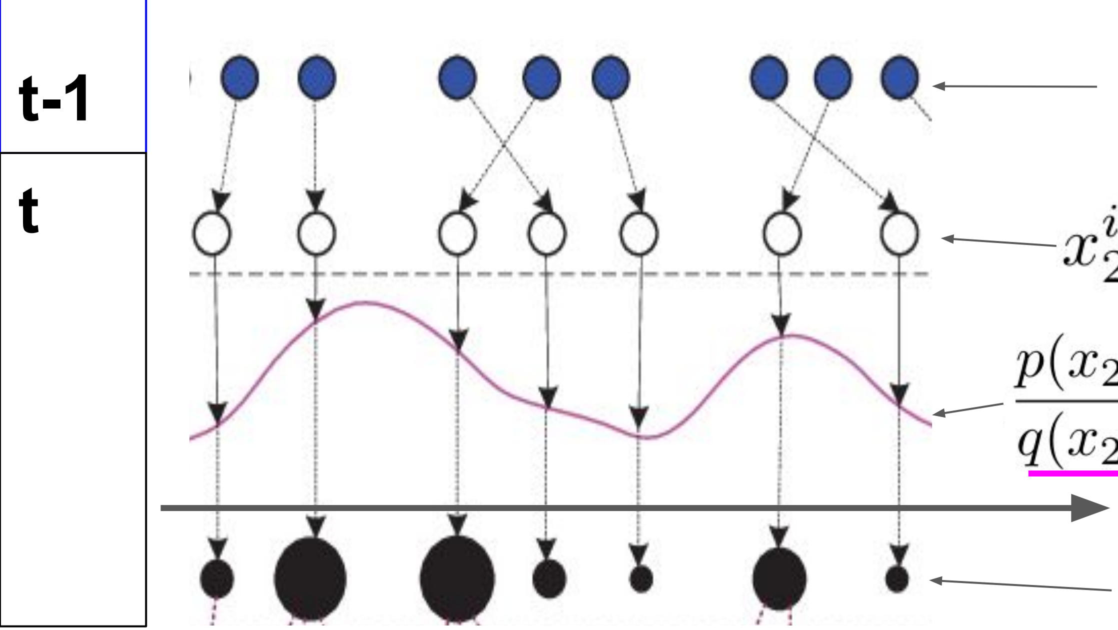
t



$\underline{x_1^i}, \underline{w_1^i}$ **Time 1 Particles**

$x_2^i \sim q(x_2 | \underline{x_1^i})$ **Time 2 Proposal Particles**

Sequence or “branch”



$$\underline{x_1^i}, \underline{w_1^i}$$

Time 1 Particles

$$x_2^i \sim q(x_2|\underline{x_1^i})$$

Time 2 Proposal Particles

$$\frac{p(x_2|x_1)}{\underline{q(x_2|x_1)}}$$

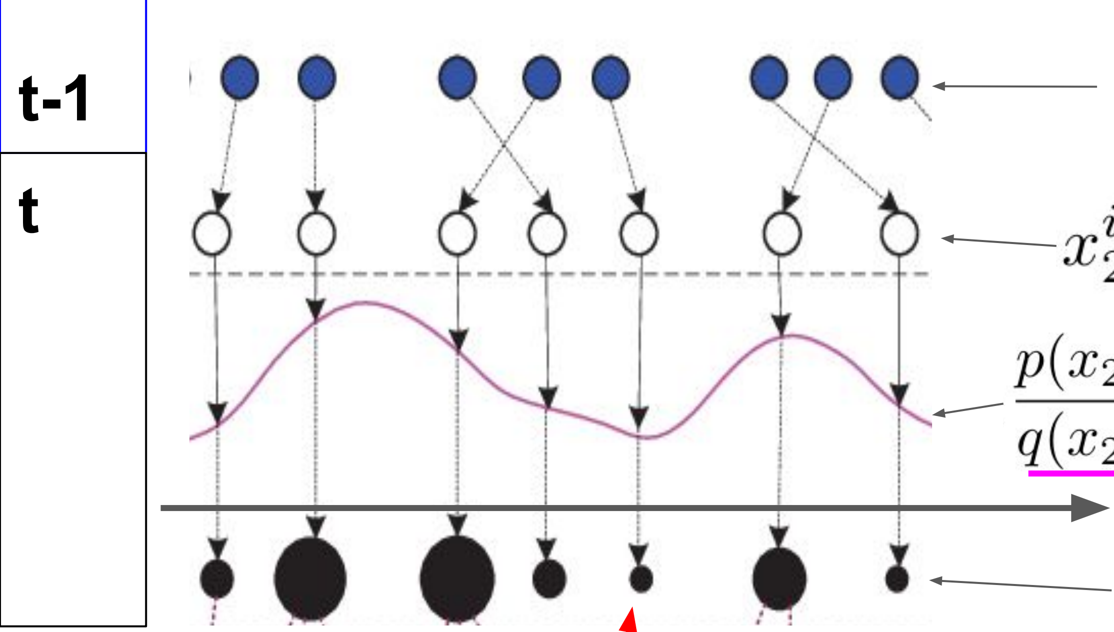
Step Importance Ratio

$$w_2 := \frac{p(x_{1:2})}{q(x_{1:2})} = \frac{p(x_1)}{\underline{q(x_1)}} \frac{p(x_2|x_1)}{\underline{q(x_2|x_1)}}$$

$$= \underline{w_1} \frac{p(x_2|x_1)}{\underline{q(x_2|x_1)}}$$

Time 2 Particles

Update Importance Weights



x_1^i, w_1^i **Time 1 Particles**

$x_2^i \sim q(x_2|x_1^i)$ **Time 2 Proposal Particles**

$\frac{p(x_2|x_1)}{q(x_2|x_1)}$ Step Importance Ratio

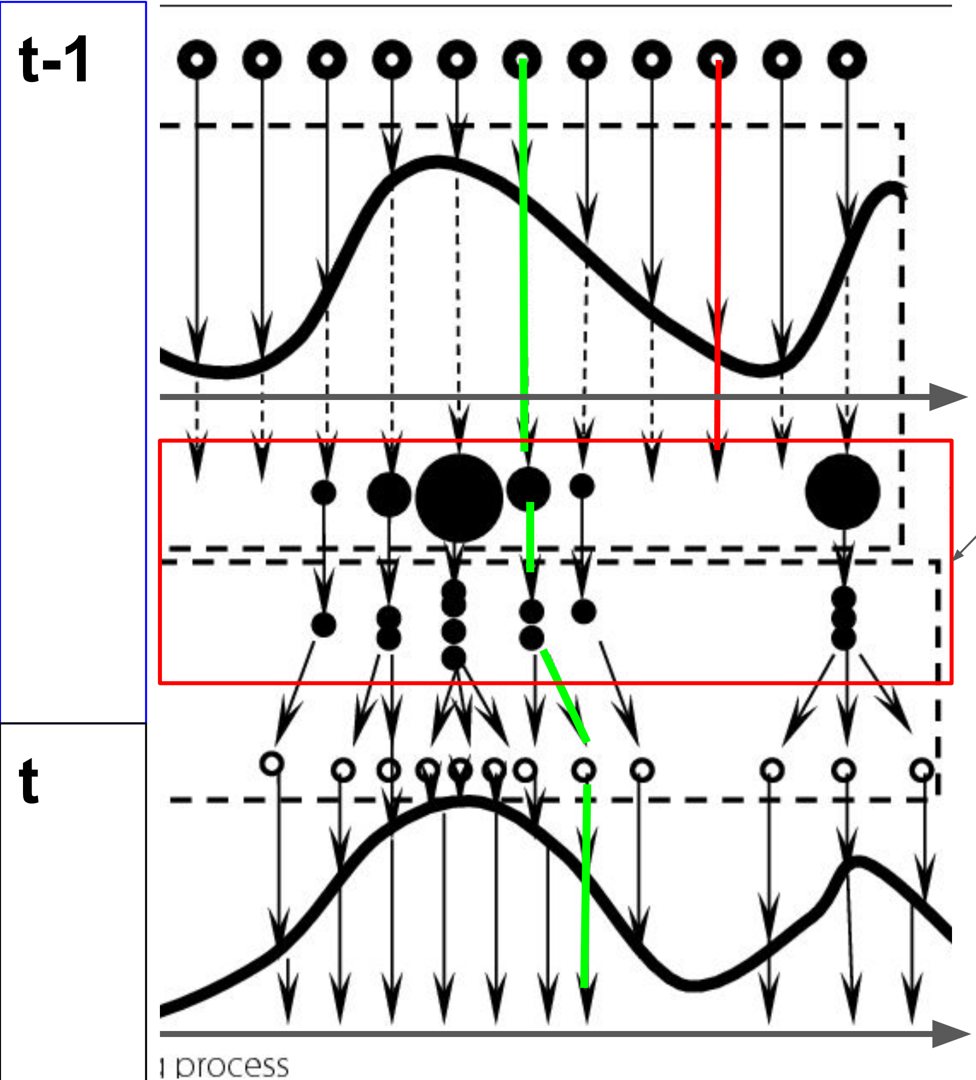
Update Importance Weights

$$w_2 := \frac{p(x_{1:2})}{q(x_{1:2})} = \frac{p(x_1)}{q(x_1)} \frac{p(x_2|x_1)}{q(x_2|x_1)}$$

$$= w_1 \frac{p(x_2|x_1)}{q(x_2|x_1)}$$

Time 2 Particles

But weights could become very small



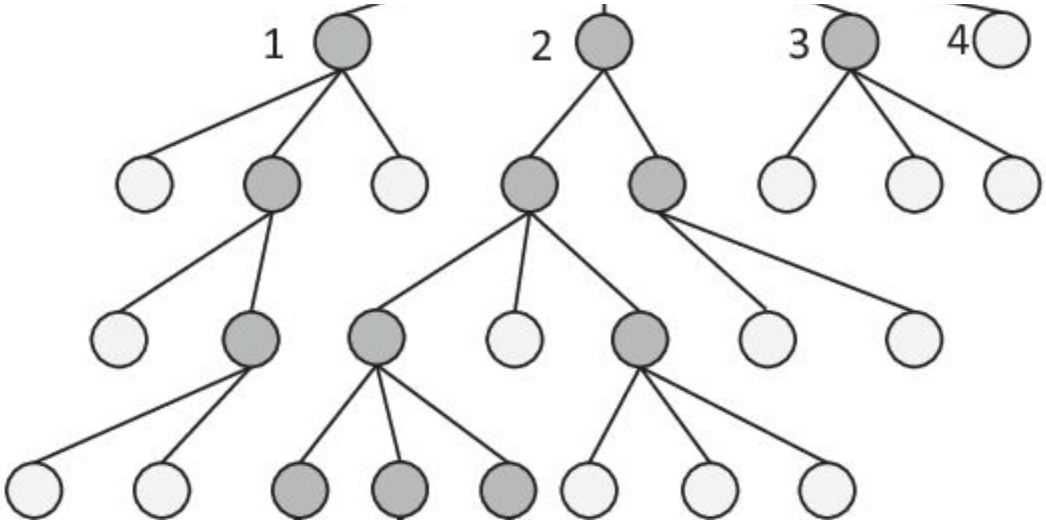
SIS with Replacement

Replacement Step:

- Discontinue low weight branches
- Refocus particles on high weight branches

$$\{\mathbf{x}_{1:i}^{(n)}\}_{n=1}^{N} \sim \mathrm{Mult}(n; w_i^{(1)}, \ldots, w_i^{(N)})$$
$$\{w_i^{(n)} = 1\}_{n=1}^{N}$$

SMC: SIS with Replacement



Only high probability branches survive.

Still representative of the overall distribution.

Model-based RL

Learns a model of the environment and uses it for RL $p_{env}(s_{t+1}|s_t, a_t)$

- **Model Predictive Planning (f.e. PETS [Chua et al. 2018])**
  - Simulate actions into the future
  - Pick ones that gave good value

Control as Inference

Proposes a heuristic for selecting actions.

Current belief of the agent:

Action A: Lose 1 dollar on average 

(higher chance to be “optimal”)

Action B: Lose 2 dollars on average 

Control as inference:

Choose Action A more often than B.

But sometimes still choose B.

Control as Inference

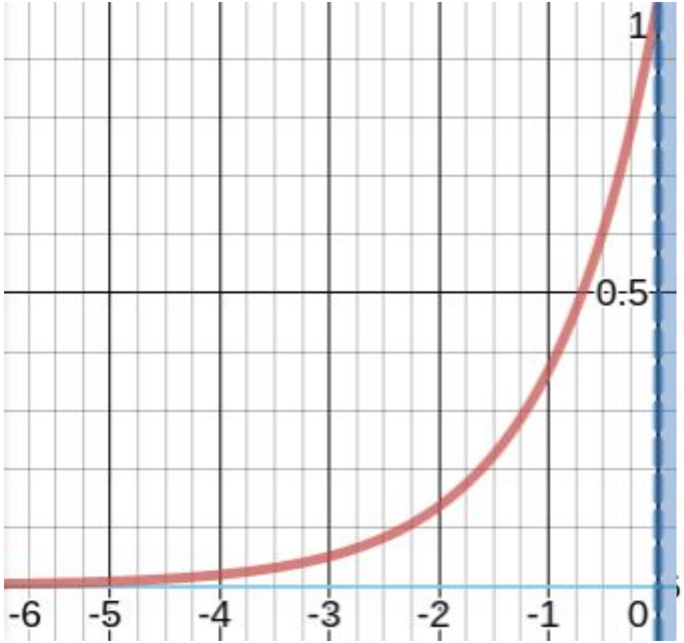
Suppose an "optimal" future.	Given that agent will lose as little money as possible,
Sample actions according to how likely they would have led to this "optimality".	which action did I likely take?

To define this formally:

Optimality Variable

$$\pi(a_1|s_1) := p(a_1|s_1, O_{1:T})$$

What is probability of “optimal”?



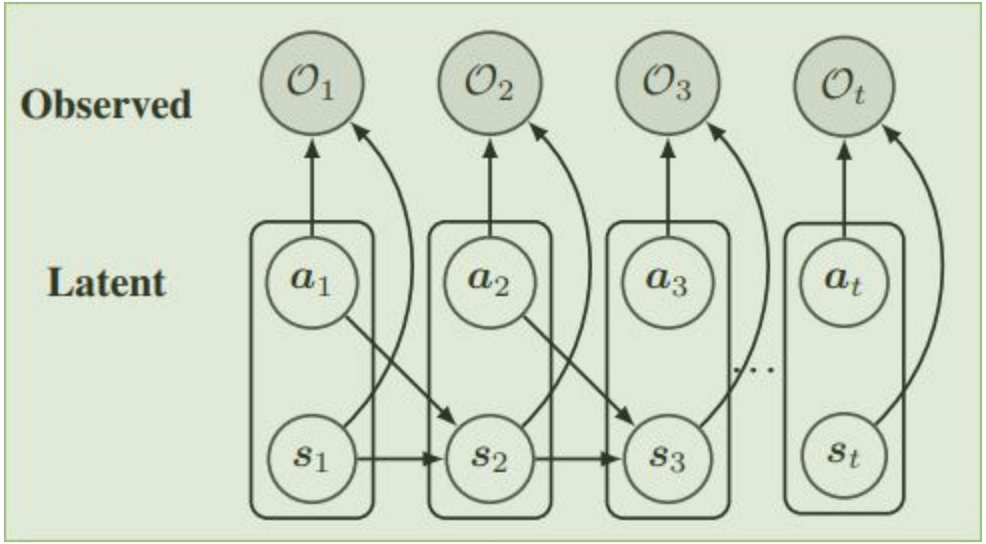
Reward (Always negative)

Heuristic: Exponential

$$p(\mathcal{O}_t = 1 | \mathbf{s}_t, \mathbf{a}_t) = \exp(r(\mathbf{s}_t, \mathbf{a}_t))$$

Lower reward
->
Exponentially less likely of being ‘optimal’
->
Exponentially less likely to be sampled

MDP Setting



MDP:

$$p_{env}(s_{t+1}|s_t, a_t)$$

Optimality at every point in time.

Choose action proportional to chance of optimality over time.

$$\pi(a_1|s_1) := p(a_1|s_1, O_{1:T})$$

But inference is hard =(

Can't efficiently sample from true posterior.

$$p(a_1|s_1, O_{1:T})$$

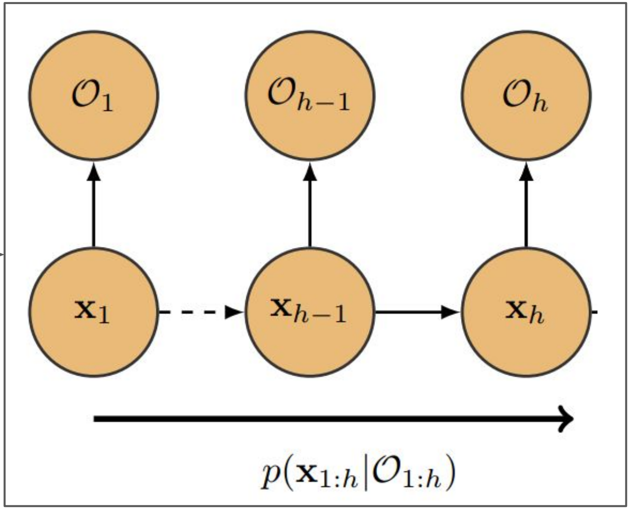
SMC to the Rescue

Want to sample futures given they are optimal:

$$p(a_1|s_1, O_{1:T})$$

$$p(\mathbf{x}_{1:h}|\mathcal{O}_{1:h})$$

How to do this?



Need a good proposal q(x1:h)

Model $p_{env}(s_{t+1}|s_t, a_t)$

Policy q(a|s)

Soft Actor Critic (SAC) [Haarnoja et. al 2018]

SAC (fairly SOTA model-free RL) learns **approximate** Control as Inference.

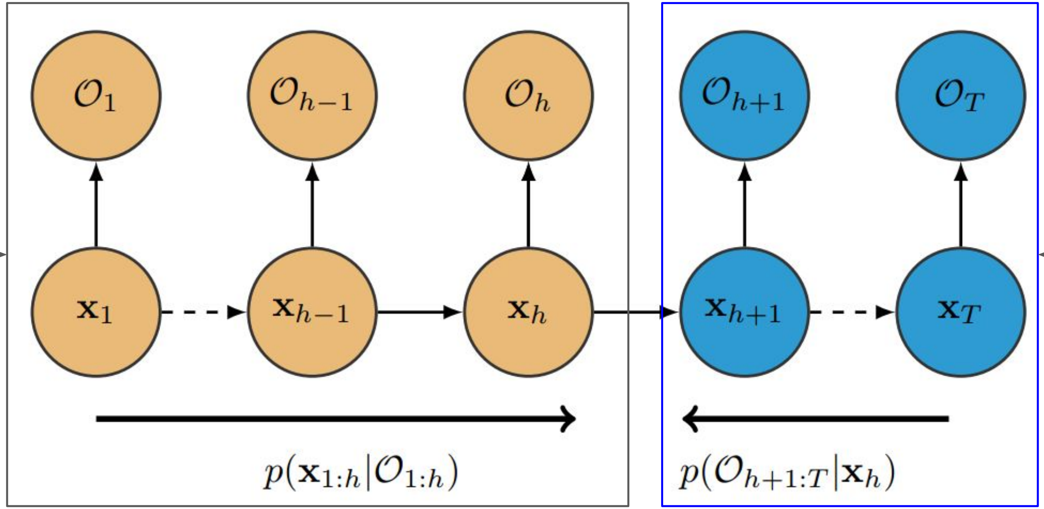
Gives us an approximate proposal policy q(a|s).

Planning as Inference

Need maximum sequence length to be practical.

$$p(\mathbf{x}_{1:h}|\mathcal{O}_{1:T}) \propto \underline{p(\mathbf{x}_{1:h}|\mathcal{O}_{1:h})} \cdot \underline{p(\mathcal{O}_{h+1:T}|\mathbf{x}_h)}$$

SMC



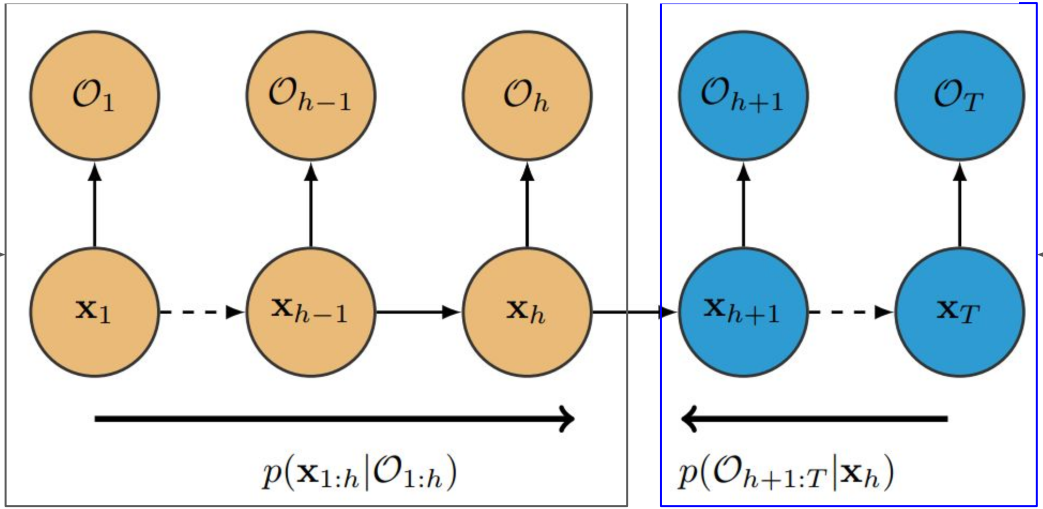
What to do about this?

Planning as Inference

Need maximum sequence length to be practical.

$$p(\mathbf{x}_{1:h}|\mathcal{O}_{1:T}) \propto \underline{p(\mathbf{x}_{1:h}|\mathcal{O}_{1:h})} \cdot \underline{p(\mathcal{O}_{h+1:T}|\mathbf{x}_h)}$$

SMC



SAC has a learned approximation.

$$p(\mathcal{O}_{h+1:T}|\mathbf{x}_h) =$$
$$\underline{\mathbb{E}_{\mathbf{s}_{h+1}|\mathbf{x}_h}\left[\exp\left(V(\mathbf{s}_{h+1})\right)\right]}$$

Planning as Inference

Related to MCTS in AlphaGo Zero.

We started with an approximate model-free proposal policy q and a value V (from SAC).

Then we looked into the future with our model via SMC.

Which allowed us to pick a more accurate action (according to Control as Inference).

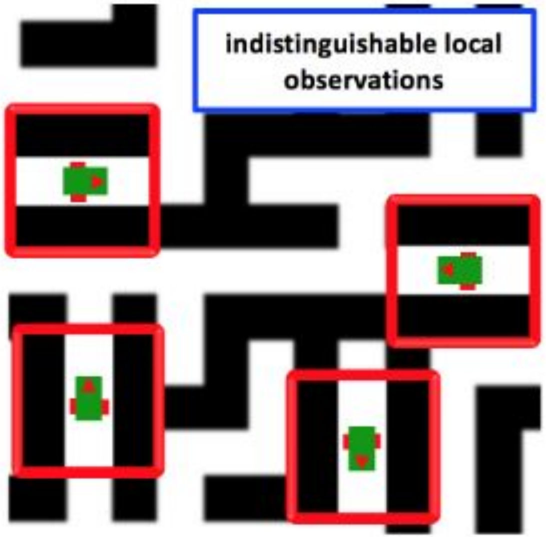
Scope and Limitations

Weight update assumes model is perfectly accurate.

When environment is stochastic, encourages risk seeking behaviours.

QMDP-Net

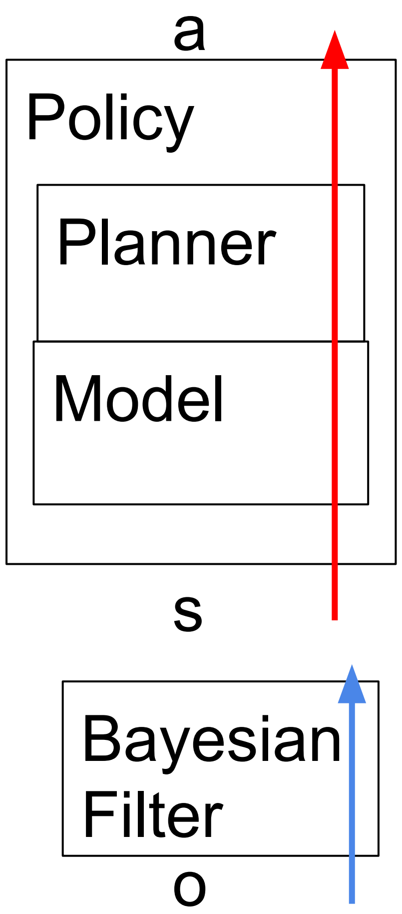
- Planning under partial observations
- Learn model of environment and planner simultaneously and end to end
- Learned model uses discrete states and actions
- Policy is trained by imitating expert data (supervised learning)



Related Work

- **Value Iteration Networks:** Fully differentiable neural network architecture for learning to plan. It embeds both a learned model of the environment and a value iteration planning module within. However, it assumes a fully observable setting and hence does not need filtering.

- **Bayesian Filtering:** Common in robotics. Continuously update a robot's belief about its state based on most recent sensor data. Recent works have shown this process to be end-to-end differentiable.



Main Contribution

- Extends VIN by also embedding a Bayesian Filter
- The entire framework is end-to-end differentiable

POMDP (Partially Observable MDP)

- **Definition:** POMDP is defined by the following components

State space	S	Latent
Action space	A	Expert Data
Observation space	O	Expert Data
State transition function	$P(s'\|s,a)$	Learned by NN
Observation transition	$\boxed{P(o\|s,a)}$	Learned by NN
Reward function	$R(s,a)$	Learned by NN

POMDP - Bayesian Filtering

- The agent does not know its exact state and maintains a belief (a probability distribution) over all the states S
- Belief is recursively updated from past history $(a_1, o_1, a_2, o_2, \ldots, a_t, o_t)$

$$b_t(s') = \eta P(o_t|s', a_t) \sum_{s \in S} P(s'|s, a_t) b_{t-1}(s)$$

New observation

Transition from previous belief

POMDP

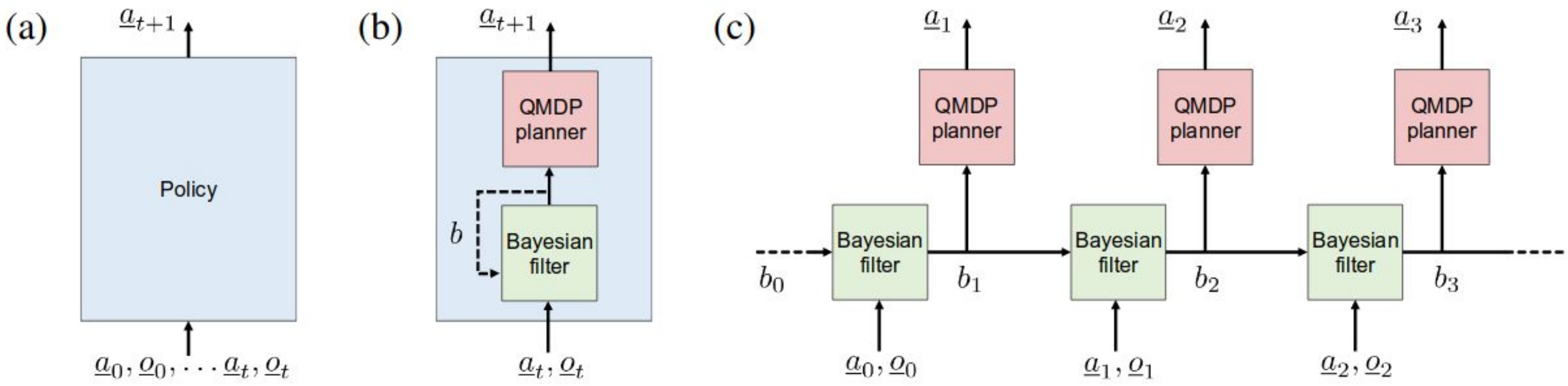
- The planning objective is to obtain a policy that maximizes the expected total discounted reward:

$$V_\pi(b_0) = \mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_{t+1}) \,\middle|\, b_0, \pi\right)$$

- Solving POMDPs exactly is computationally intractable in the worst case*** (intuitively, because we need to integrate over all states - blowup!)
- Approximate solutions needed

*** C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

QMDP-net: Overall architecture



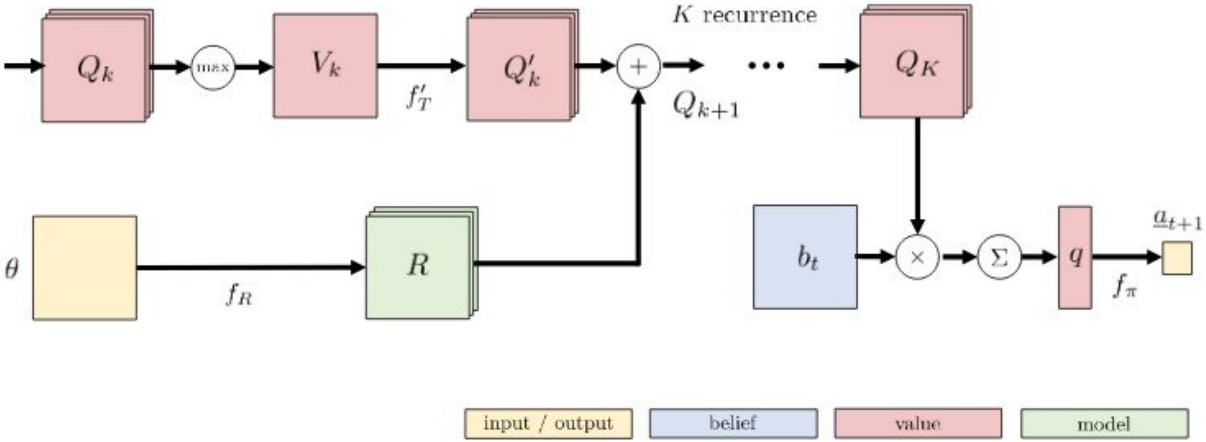
- There are two main components: the QMDP planner (similar to VIN) and the Bayesian filter

QMDP Planner Module

- The planner module performs value iteration (each step is differentiable). The architecture is very similar to Value Iteration Networks (VIN)
- Iteratively apply Bellman updates to the Q value map over states to refine it

$$Q_{k+1}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')$$

$$V_k(s) = \max_a Q_k(s,a)$$



Action selection

- The obtained Q value map is weighed by the computed belief over states to obtain a probability distribution over actions

$$q(a) = \sum_{s \in S} Q_K(s, a) b_t(s)$$

- Select the action with maximum q( ) value

Highlights, Scope, and Limitations

- Only demonstrate on Imitation Learning (RL is possible in principle)
- Bayes filter is not “exact” but “useful”
- Discrete action and state model unlikely to scale to more complicated environments

Thank you for your time!

We will be happy to take questions

Appendix... next few slides

Stuff we didn't have time for...

Importance Sampling (IS)

Integral intractable and **can’t sample easily**.

But can sample from q(x). -> Approximate p(x) with N **samples from q(x)**.

$$\mathbb{E}_{p(x)}[f(x)] = \int p(x)f(x)dx = \int q(x)\left(\frac{p(x)}{q(x)}\right) f(x)dx = \mathbb{E}_{q(x)}[w(x)f(x)]$$

$$\widehat{P}_N(dx) = \sum_{i=1}^{N}\left(\frac{w^i}{N}\right)\delta_{x^i}(dx^i) \quad x^i \sim q(x) \quad w^i = \frac{p(x^i)}{q(x^i)}$$

Also need to be able to **evaluate p(x) exactly**!

https://www.stats.ox.ac.uk/~doucet/doucet_defreitas_gordon_smcbookintro.pdf [1.3.2]

Importance Sampling with Self-Normalized Weights

Integral intractable and can't sample easily **and can't evaluate p(x)**.

But can evaluate p(x) **upto normalizing constant**. $\gamma(x) = Cp(x)$

Note: Very important for posterior inference:

$$p(x|o) = \frac{p(o|x)p(x)}{p(o)} = \frac{p(o|x)p(x)}{\int p(o|x)p(x)dx} = Cp(o|x)p(x)$$

Almost always hard

Importance Sampling with Self-Normalized Weights

Integral intractable and can’t sample easily **and can’t evaluate p(x)**.

But can evaluate p(x) **upto normalizing constant**. $\gamma(x) = Cp(x)$

If we try defining the weight, ignoring C: $w(x) = \frac{\gamma(x)}{q(x)}$

We see that our IS estimate is off by the multiplicative constant:

$$\mathbb{E}_{q(x)}[w(x)f(x)] = \int q(x)\left(\frac{Cp(x)}{q(x)}\right)f(x)dx = \int p(x)Cf(x)dx = \mathbb{E}_{p(x)}[Cf(x)]$$

Importance Sampling with Self-Normalized Weights

Integral intractable and can't sample easily **and can't evaluate p(x)**.

But can evaluate p(x) **upto normalizing constant**. $\gamma(x) = Cp(x)$

If we try defining the weight, ignoring C: $w(x) = \frac{\gamma(x)}{q(x)}$

We see that our IS estimate is off by the multiplicative constant:

$$\mathbb{E}_{q(x)}[w(x)f(x)] = \int q(x)\left(\frac{Cp(x)}{q(x)}\right) f(x)dx = \int p(x)Cf(x)dx = \mathbb{E}_{p(x)}[Cf(x)]$$

Idea: Normalize the weights!

Importance Sampling with Self-Normalized Weights

What if we normalize w(x)?

Average weight is an estimate of C:

$$\boxed{\mathbb{E}_{q(x)}[w(x)]} = \int q(x)w(x) = \int p(x)C = C$$

Normalizing by weights amounts to normalizing by C:

$$\frac{\mathbb{E}_{q(x)}[w(x)f(x)]}{\boxed{\mathbb{E}_{q(x)}[w(x)]}} = \frac{\mathbb{E}_{p(x)}[Cf(x)]}{C} = \mathbb{E}_{p(x)}[f(x)]$$

Importance Sampling with Self-Normalizing Weights

Normalizing by weights amounts to normalizing by C:

$$\frac{\mathbb{E}_{q(x)}[w(x)f(x)]}{\boxed{\mathbb{E}_{q(x)}[w(x)]}} = \frac{\mathbb{E}_{p(x)}[Cf(x)]}{C} = \mathbb{E}_{p(x)}[f(x)]$$

Which motivates:

$$\widehat{P}_N(dx) = \sum_{i=1}^{N}\left(\frac{w^i}{\boxed{\sum_{i=1}^{N} w^i}}\right)\delta_{x^i}(dx^i) \quad x^i \sim q(x) \quad w^i \propto \frac{p(x^i)}{q(x^i)}$$

We explicitly normalize the weights so that they sum to 1.

(Diverge from theory -> incurs a bias but helps with variance reduction)

Sequential Importance Sampling (SIS)

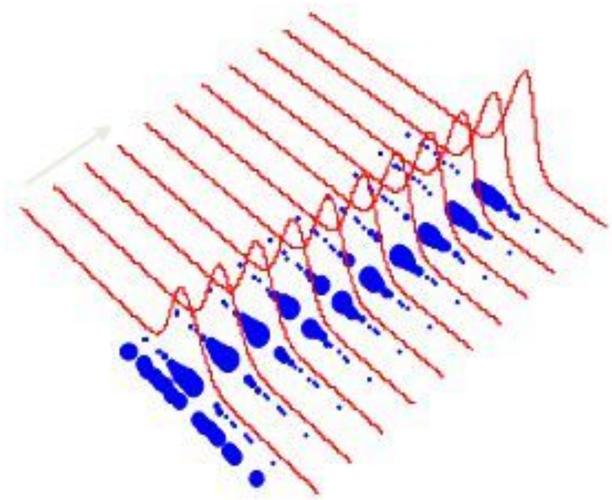
Sample from a proposal distribution:

$$q(x_{1:t}) = q(x_1) \prod_{j=2}^{t} q(x_j | x_{1:j-1})$$

Initial Distribution

Update

$$w_t := \frac{p(x_{1:t})}{q(x_{1:t})} = \boxed{\frac{p(x_{1:t-1})}{q(x_{1:t-1})}} \frac{p(x_t | x_{1:t-1})}{q(x_t | x_{1:t-1})} = \boxed{w_{t-1}} \frac{p(x_t | x_{1:t-1})}{q(x_t | x_{1:t-1})}$$



The overall algorithm

1. Sample actions from prior

Algorithm 1 SMC Planning using SIR

for t in $\{1, \ldots, T\}$ **do**
  $\{\mathbf{s}_t^{(n)} = \mathbf{s}_t\}_{n=1}^N$
  $\{w_t^{(n)} = 1\}_{n=1}^N$
  for i in $\{t, \ldots, t+h\}$ **do**
    // Update
    $\{\mathbf{a}_i^{(n)} \sim \pi(\mathbf{a}_i^{(n)} | \mathbf{s}_i^{(n)})\}_{n=1}^N$
    $\{\mathbf{s}_{i+1}^{(n)}, r_i^{(n)} \sim p_{\text{model}}(\cdot | \mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)})\}_{n=1}^N$
    $\{w_i^{(n)} \propto w_{i-1}^{(n)} \cdot \exp\left(A(\mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)}, \mathbf{s}_{i+1}^{(n)})\right)\}_{n=1}^N$
    // Resampling
    $\{\mathbf{x}_{1:i}^{(n)}\}_{n=1}^N \sim \text{Mult}(n; w_i^{(1)}, \ldots, w_i^{(N)})$
    $\{w_i^{(n)} = 1\}_{n=1}^N$
  end for
  Sample $n \sim \text{Uniform}(1, N)$.
  // Model Predictive Control
  Select $\mathbf{a}_t$, first action of $\mathbf{x}_{t:t+h}^{(n)}$
  $\mathbf{s}_{t+1}, r_t \sim p_{\text{env}}(\cdot | \mathbf{s}_t, \mathbf{a}_t)$
  Add $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ to buffer $\mathcal{B}$
  Update π, V and p_{model} with $\mathcal{B}$
end for

The overall algorithm

1. Sample actions from prior
2. Simulate with model

Algorithm 1 SMC Planning using SIR

for t in $\{1, \ldots, T\}$ **do**
  $\{\mathbf{s}_t^{(n)} = \mathbf{s}_t\}_{n=1}^N$
  $\{w_t^{(n)} = 1\}_{n=1}^N$
  for i in $\{t, \ldots, t+h\}$ **do**
    // Update
    $\{\mathbf{a}_i^{(n)} \sim \pi(\mathbf{a}_i^{(n)} | \mathbf{s}_i^{(n)})\}_{n=1}^N$
    $\{\mathbf{s}_{i+1}^{(n)}, r_i^{(n)} \sim p_{\text{model}}(\cdot | \mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)})\}_{n=1}^N$
    $\{w_i^{(n)} \propto w_{i-1}^{(n)} \cdot \exp\left(A(\mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)}, \mathbf{s}_{i+1}^{(n)})\right)\}_{n=1}^N$
    // Resampling
    $\{\mathbf{x}_{1:i}^{(n)}\}_{n=1}^N \sim \text{Mult}(n; w_i^{(1)}, \ldots, w_i^{(N)})$
    $\{w_i^{(n)} = 1\}_{n=1}^N$
  end for
  Sample $n \sim \text{Uniform}(1, N)$.
  // Model Predictive Control
  Select $\mathbf{a}_t$, first action of $\mathbf{x}_{t:t+h}^{(n)}$
  $\mathbf{s}_{t+1}, r_t \sim p_{\text{env}}(\cdot | \mathbf{s}_t, \mathbf{a}_t)$
  Add $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ to buffer $\mathcal{B}$
  Update π, V and p_{model} with $\mathcal{B}$
end for

The overall algorithm

1. Sample actions from prior
2. Simulate with model
3. Update weight of each branch using reward and SAC 'Value'

Algorithm 1 SMC Planning using SIR

for t in $\{1, \dots, T\}$ **do**

  $\{\mathbf{s}_t^{(n)} = \mathbf{s}_t\}_{n=1}^N$

  $\{w_t^{(n)} = 1\}_{n=1}^N$

  for i in $\{t, \dots, t+h\}$ **do**

    // Update

    $\{\mathbf{a}_i^{(n)} \sim \pi(\mathbf{a}_i^{(n)} | \mathbf{s}_i^{(n)})\}_{n=1}^N$

    $\{\mathbf{s}_{i+1}^{(n)}, r_i^{(n)} \sim p_{\text{model}}(\cdot | \mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)})\}_{n=1}^N$

    $\{w_i^{(n)} \propto w_{i-1}^{(n)} \cdot \exp\left(A(\mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)}, \mathbf{s}_{i+1}^{(n)})\right)\}_{n=1}^N$

    // Resampling

    $\{\mathbf{x}_{1:i}^{(n)}\}_{n=1}^N \sim \text{Mult}(n; w_i^{(1)}, \dots, w_i^{(N)})$

    $\{w_i^{(n)} = 1\}_{n=1}^N$

  end for

  Sample $n \sim \text{Uniform}(1, N)$.

  // Model Predictive Control

  Select $\mathbf{a}_t$, first action of $\mathbf{x}_{t:t+h}^{(n)}$

  $\mathbf{s}_{t+1}, r_t \sim p_{\text{env}}(\cdot | \mathbf{s}_t, \mathbf{a}_t)$

  Add $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ to buffer $\mathcal{B}$

  Update π, V and p_{model} with $\mathcal{B}$

end for

The overall algorithm

1. Sample actions from prior
2. Simulate with model
3. Update weight of each branch using reward and SAC ‘Value’
4. Reallocate search particles to more promising branches

Algorithm 1 SMC Planning using SIR

for t in $\{1, \dots, T\}$ **do**
  $\{\mathbf{s}_t^{(n)} = \mathbf{s}_t\}_{n=1}^N$
  $\{w_t^{(n)} = 1\}_{n=1}^N$
  for i in $\{t, \dots, t+h\}$ **do**
    // Update
    $\{\mathbf{a}_i^{(n)} \sim \pi(\mathbf{a}_i^{(n)} | \mathbf{s}_i^{(n)})\}_{n=1}^N$
    $\{\mathbf{s}_{i+1}^{(n)}, r_i^{(n)} \sim p_{\text{model}}(\cdot | \mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)})\}_{n=1}^N$
    $\{w_i^{(n)} \propto w_{i-1}^{(n)} \cdot \exp\left(A(\mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)}, \mathbf{s}_{i+1}^{(n)})\right)\}_{n=1}^N$
    // Resampling
    $\{\mathbf{x}_{1:i}^{(n)}\}_{n=1}^N \sim \text{Mult}(n; w_i^{(1)}, \dots, w_i^{(N)})$
    $\{w_i^{(n)} = 1\}_{n=1}^N$
  end for
  Sample $n \sim \text{Uniform}(1, N)$.
  // Model Predictive Control
  Select $\mathbf{a}_t$, first action of $\mathbf{x}_{t:t+h}^{(n)}$
  $\mathbf{s}_{t+1}, r_t \sim p_{\text{env}}(\cdot | \mathbf{s}_t, \mathbf{a}_t)$
  Add $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ to buffer $\mathcal{B}$
  Update π, V and p_{model} with $\mathcal{B}$
end for

The overall algorithm

1. Sample actions from prior
2. Simulate with model
3. Update weight of each branch using reward and SAC 'Value'
4. Reallocate search particles to more promising branches
5. Repeat until horizon

Algorithm 1 SMC Planning using SIR

for t in $\{1, \ldots, T\}$ **do**
 $\{\mathbf{s}_t^{(n)} = \mathbf{s}_t\}_{n=1}^N$
 $\{w_t^{(n)} = 1\}_{n=1}^N$
 for i in $\{t, \ldots, t+h\}$ **do**
  // Update
  $\{\mathbf{a}_i^{(n)} \sim \pi(\mathbf{a}_i^{(n)}|\mathbf{s}_i^{(n)})\}_{n=1}^N$
  $\{\mathbf{s}_{i+1}^{(n)}, r_i^{(n)} \sim p_{\text{model}}(\cdot|\mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)})\}_{n=1}^N$
  $\{w_i^{(n)} \propto w_{i-1}^{(n)} \cdot \exp\left(A(\mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)}, \mathbf{s}_{i+1}^{(n)})\right)\}_{n=1}^N$
  // Resampling
  $\{\mathbf{x}_{1:i}^{(n)}\}_{n=1}^N \sim \text{Mult}(n; w_i^{(1)}, \ldots, w_i^{(N)})$
  $\{w_i^{(n)} = 1\}_{n=1}^N$
 end for
 Sample $n \sim \text{Uniform}(1, N)$.
 // Model Predictive Control
 Select $\mathbf{a}_t$, first action of $\mathbf{x}_{t:t+h}^{(n)}$
 $\mathbf{s}_{t+1}, r_t \sim p_{\text{env}}(\cdot|\mathbf{s}_t, \mathbf{a}_t)$
 Add $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ to buffer $\mathcal{B}$
 Update π, V and p_{model} with $\mathcal{B}$
end for

The overall algorithm

1. Sample actions from prior
2. Simulate with model
3. Update weight of each branch using reward and SAC 'Value'
4. Reallocate search particles to more promising branches
5. Repeat until horizon
6. Randomly select first action from remaining branches

Algorithm 1 SMC Planning using SIR

for t in $\{1, \dots, T\}$ **do**
  $\{\mathbf{s}_t^{(n)} = \mathbf{s}_t\}_{n=1}^N$
  $\{w_t^{(n)} = 1\}_{n=1}^N$
  for i in $\{t, \dots, t+h\}$ **do**
    // Update
    $\{\mathbf{a}_i^{(n)} \sim \pi(\mathbf{a}_i^{(n)} | \mathbf{s}_i^{(n)})\}_{n=1}^N$
    $\{\mathbf{s}_{i+1}^{(n)}, r_i^{(n)} \sim p_{\text{model}}(\cdot | \mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)})\}_{n=1}^N$
    $\{w_i^{(n)} \propto w_{i-1}^{(n)} \cdot \exp\left(A(\mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)}, \mathbf{s}_{i+1}^{(n)})\right)\}_{n=1}^N$
    // Resampling
    $\{\mathbf{x}_{1:i}^{(n)}\}_{n=1}^N \sim \text{Mult}(n; w_i^{(1)}, \dots, w_i^{(N)})$
    $\{w_i^{(n)} = 1\}_{n=1}^N$
  end for
  Sample $n \sim \text{Uniform}(1, N)$.
  // Model Predictive Control
  Select $\mathbf{a}_t$, first action of $\mathbf{x}_{t:t+h}^{(n)}$
  $\mathbf{s}_{t+1}, r_t \sim p_{\text{env}}(\cdot | \mathbf{s}_t, \mathbf{a}_t)$
  Add $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ to buffer $\mathcal{B}$
  Update π, V and p_{model} with $\mathcal{B}$
end for

Deriving weight updates (read the paper for details)

$$
\begin{aligned}
w_t &= \frac{p(\mathbf{x}_{1:t}|\mathcal{O}_{1:T})}{q(\mathbf{x}_{1:t})} \\
&= \frac{p(\mathbf{x}_{1:t-1}|\mathcal{O}_{1:T})}{q(\mathbf{x}_{1:t-1})} \frac{p(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathcal{O}_{1:T})}{q(\mathbf{x}_t|\mathbf{x}_{1:t-1})} \\
&= w_{t-1} \cdot \frac{p(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathcal{O}_{1:T})}{q(\mathbf{x}_t|\mathbf{x}_{1:t-1})} \\
&= w_{t-1} \frac{1}{q(\mathbf{x}_t|\mathbf{x}_{1:t-1})} \frac{p(\mathbf{x}_{1:t}|\mathcal{O}_{1:T})}{p(\mathbf{x}_{1:t-1}|\mathcal{O}_{1:T})}
\end{aligned}
$$

We use there the forward-backward equation 3.1 for the numerator and the denominator

$$
\begin{aligned}
&\propto w_{t-1} \frac{1}{q(\mathbf{x}_t|\mathbf{x}_{1:t-1})} \frac{p(\mathbf{x}_{1:t}|\mathcal{O}_{1:t})}{p(\mathbf{x}_{1:t-1}|\mathcal{O}_{1:t-1})} \frac{p(\mathcal{O}_{t+1:T}|\mathbf{x}_t)}{p(\mathcal{O}_{t:T}|\mathbf{x}_{t-1})} \\
&= w_{t-1} \frac{p(\mathbf{x}_t|\mathbf{x}_{1:t-1})}{q(\mathbf{x}_t|\mathbf{x}_{1:t-1})} p(\mathcal{O}_t|\mathbf{x}_t) \frac{p(\mathcal{O}_{t+1:T}|\mathbf{x}_t)}{p(\mathcal{O}_{t:T}|\mathbf{x}_{t-1})} \\
&= w_{t-1} \frac{p_{\text{env}}(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})}{p_{\text{model}}(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})} \frac{\exp(r_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \frac{\mathbb{E}_{\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t}[\exp(V(\mathbf{s}_{t+1}))]}{\mathbb{E}_{\mathbf{s}_t|\mathbf{s}_{t-1},\mathbf{a}_{t-1}}[\exp(V(\mathbf{s}_t))]}
\end{aligned}
$$

Connection to MCTS in AlphaGo Zero

	Planning with SMC	AlphaGo Zero
Move selection criteria	$p(O_{1:T} \| x_1)$	Q upper confidence bound
Environment model	Learned p_model	Self-play p
Amortised prior policy	q from SAC	Learned prior p
Amortised prior “value”	V from SAC	V upper confidence

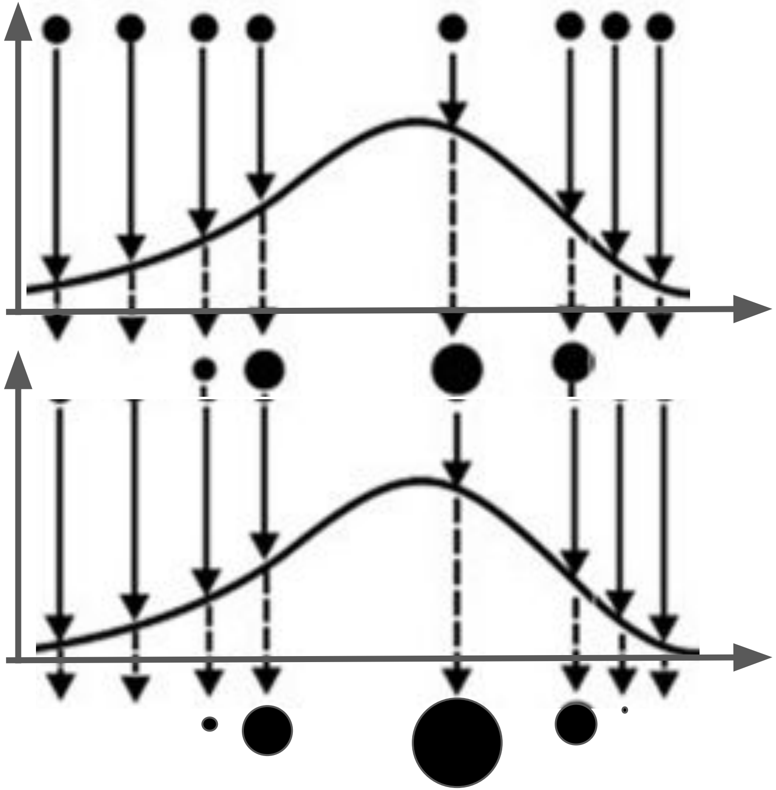
Sequential Importance Sampling

Grow sequence incrementally:

$$x_t \sim q(x_t | x_{1:t-1})$$

Update w recursively:

$$w_t := \frac{p(x_{1:t})}{q(x_{1:t})} = \boxed{\frac{p(x_{1:t-1})}{q(x_{1:t-1})}} \frac{p(x_t | x_{1:t-1})}{q(x_t | x_{1:t-1})} = \boxed{w_{t-1}} \frac{p(x_t | x_{1:t-1})}{q(x_t | x_{1:t-1})}$$



But most particles might become useless (w->0)