Monte Carlo Methods in Learning

Nando de Freitas

University of British Columbia

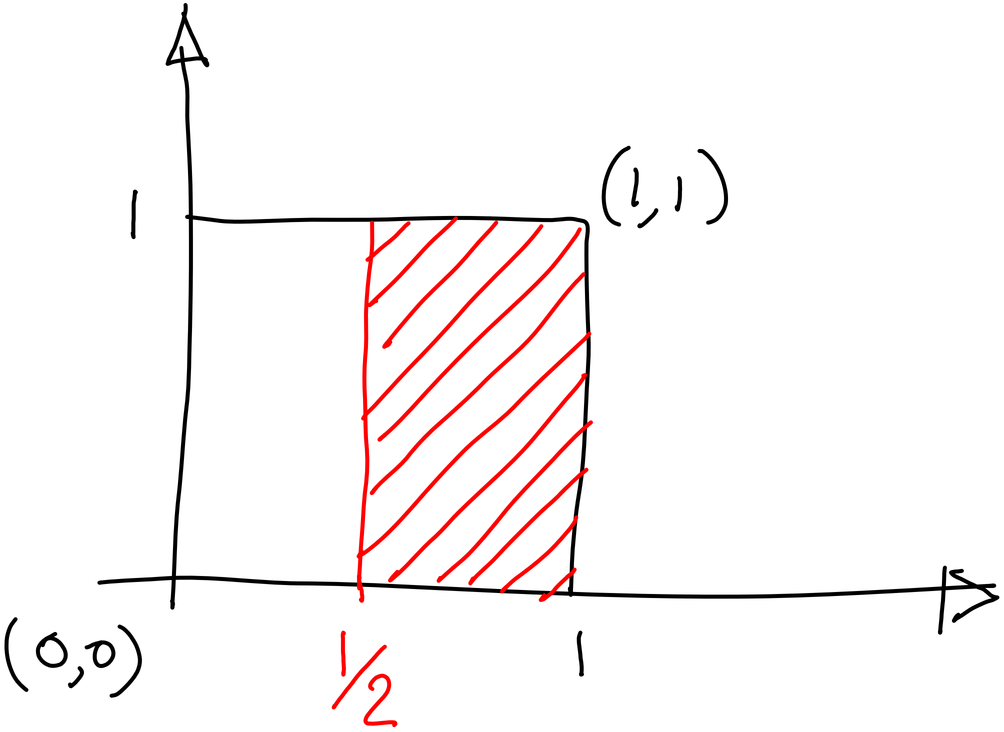
ANU Machine learning summer school 2008

Overview

- History and applications
- Rejection sampling
- Importance sampling
- Particle filters
- Markov chain Monte Carlo
  - Metropolis-Hastings
  - Gibbs sampling
- Advanced Monte Carlo methods
  - Mixtures of kernels
  - Trans-dimensional Monte Carlo
  - Hybrid Monte Carlo
  - Monte Carlo EM
  - Particle methods with artificial dynamics

The idea

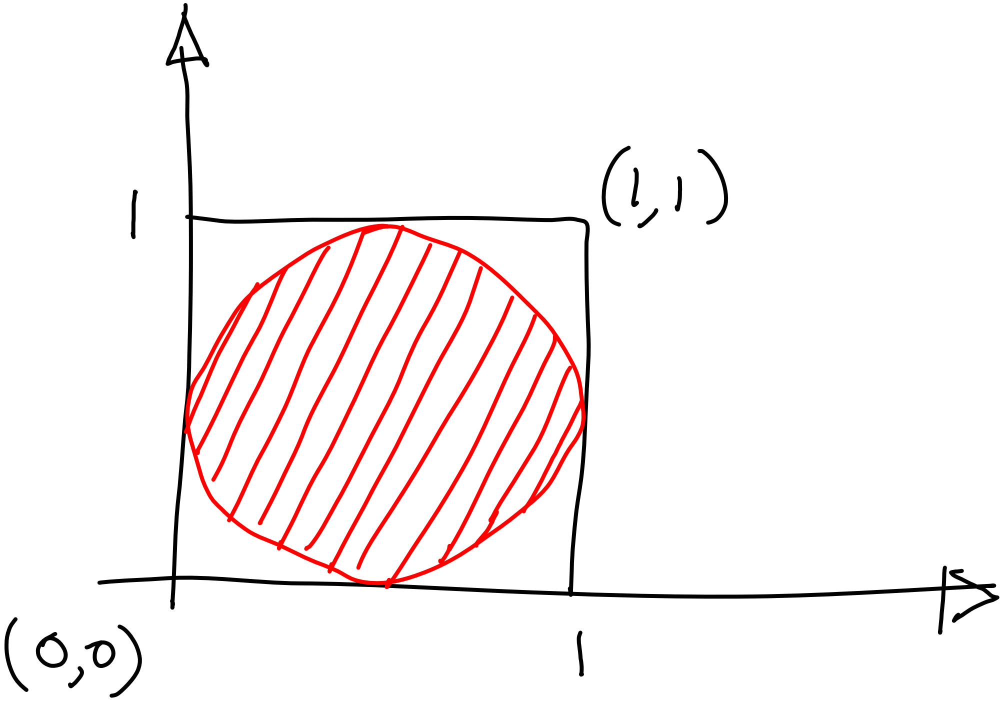
What is the probability that a dart thrown uniformly at random will hit the red area?



$$P(\text{area}) = 1/2$$

The idea

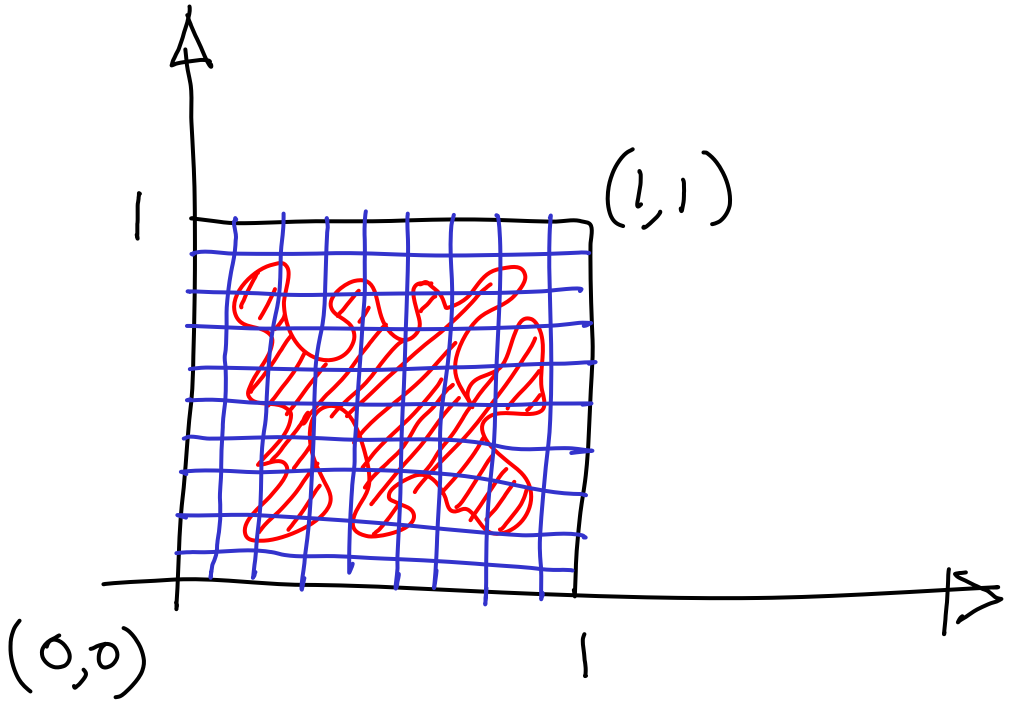
What is the probability that a dart thrown uniformly at random will hit the red area?



$$P(\text{area}) = \frac{\pi\left(\frac{1}{2}\right)^2}{} = \frac{\pi}{4}$$

The idea

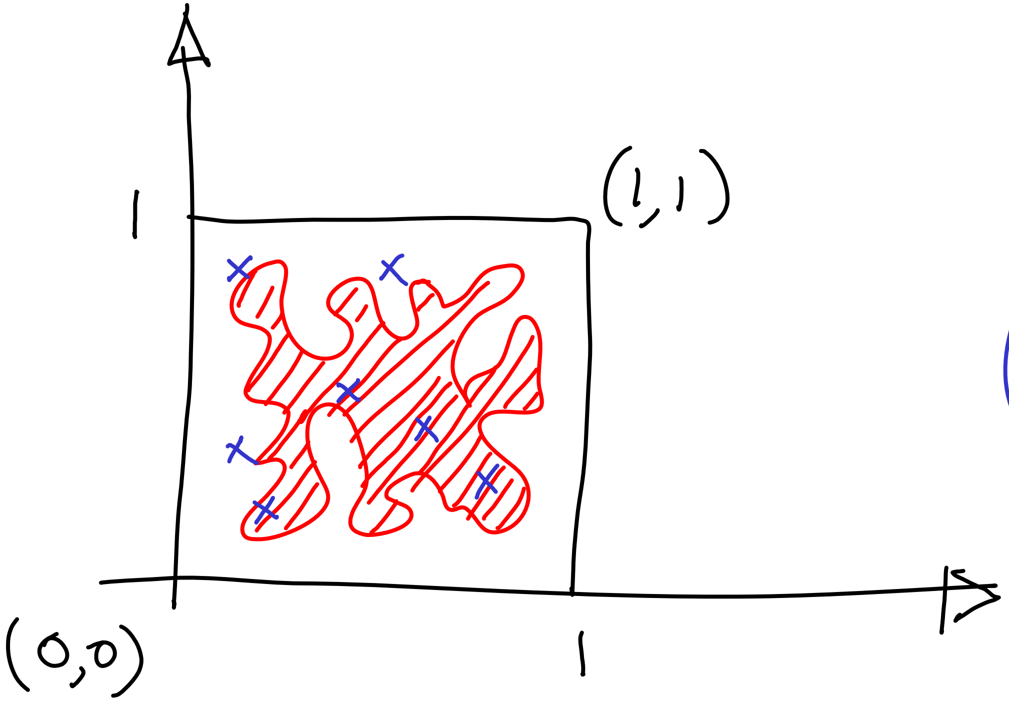
What is the probability that a dart thrown uniformly at random will hit the red area?



$$P(\text{area}) = \frac{\#\text{ red boxes}}{\#\text{ white boxes}}$$

The idea

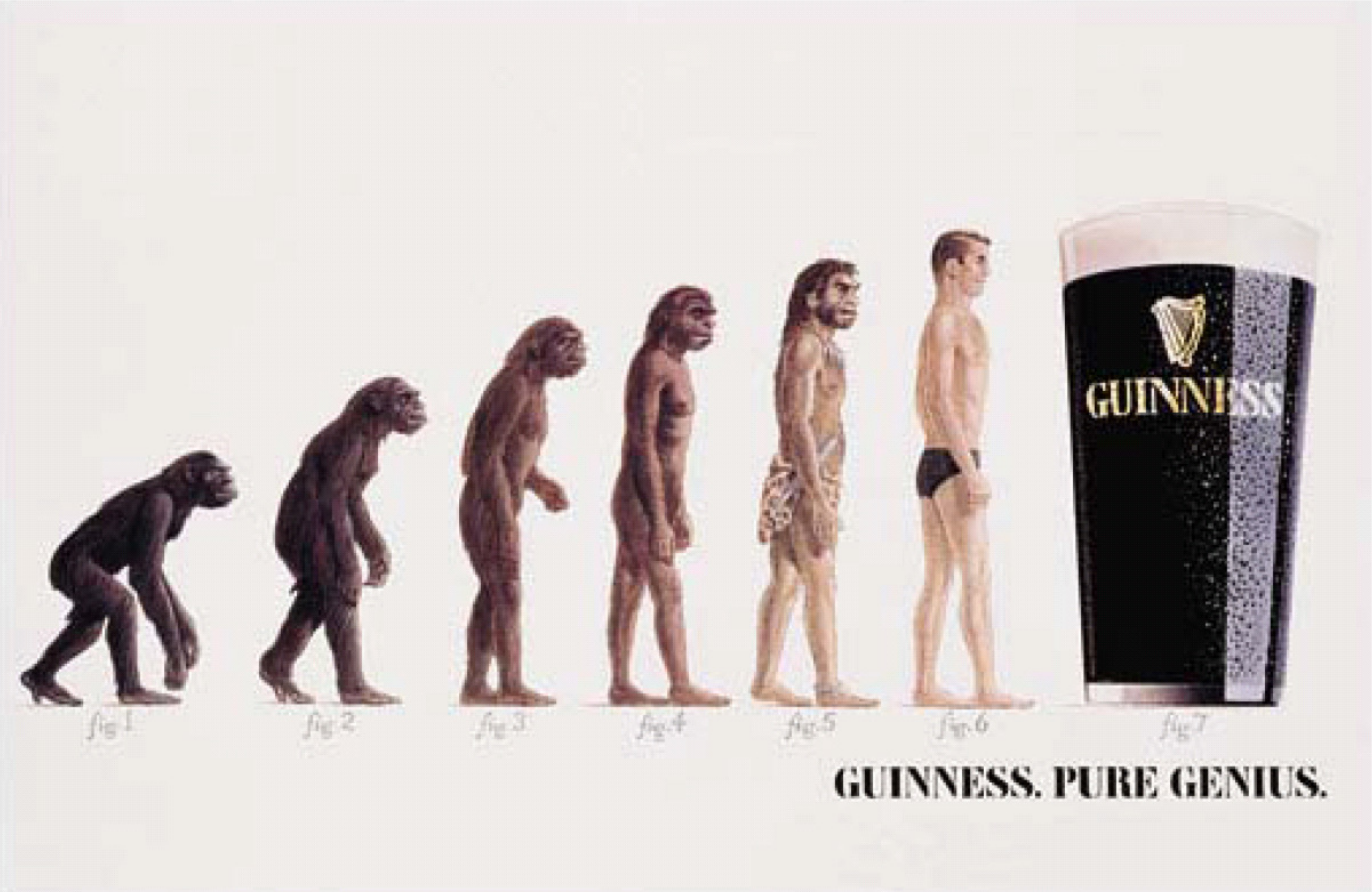
What is the probability that a dart thrown uniformly at random will hit the red area?



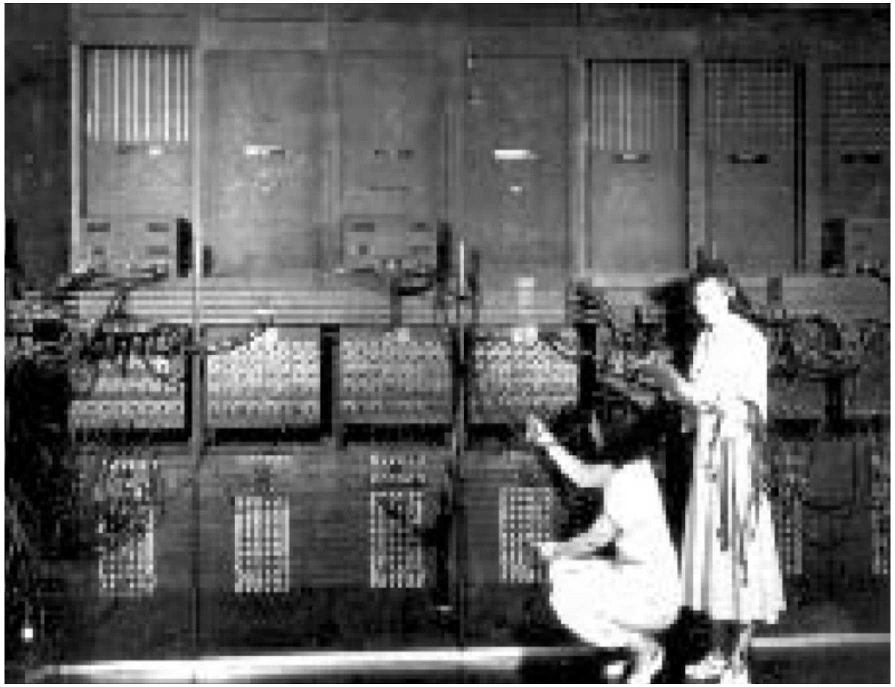
$\boxed{4/7}$

$$P(\text{area}) = \frac{\#\ \text{darts in}\ \blacksquare}{\#\ \text{darts in}\ \square}$$

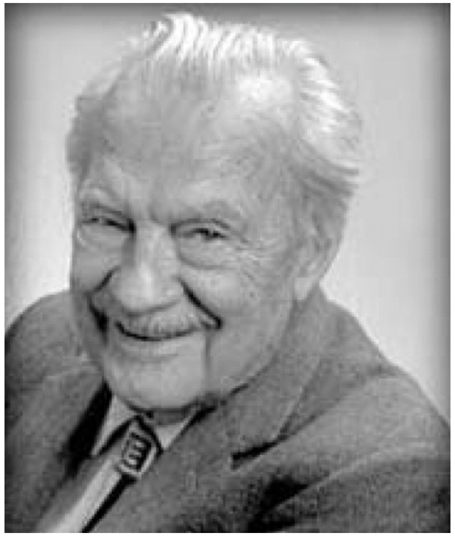
History of the Monte Carlo method



History of the Monte Carlo method: The bomb and ENIAC



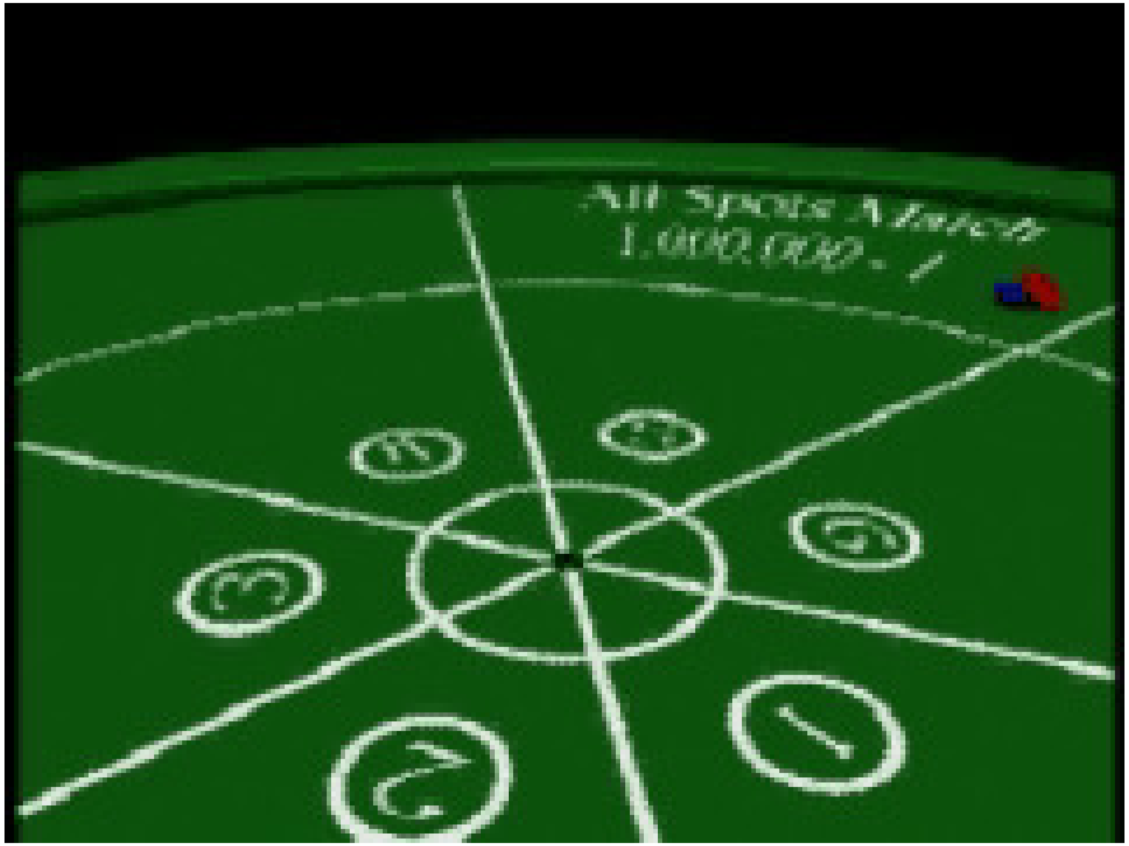
History of the Monte Carlo method



Applications of Monte Carlo

- Simulation
  - Animation
  - Physical simulation (e.g. estimating neutron diffusion time)
- Optimization
  - Generalization of simulated annealing
  - Monte Carlo expectation maximization (EM)
- Integration
  - Bayesian statistics: normalizing constants, expectations, marginalization
  - Computing expected utilities and best responses toward Nash equilibria
  - Computing volumes in high-dimensions
  - Computing eigen-functions and values of operators (e.g. Shrodinger's)
  - Statistical physics
  - Counting many things as fast as possible

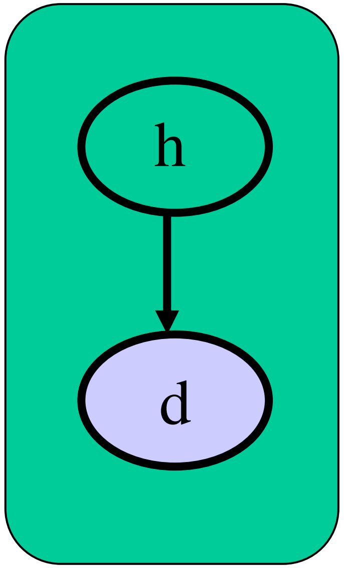
A Simulation Example

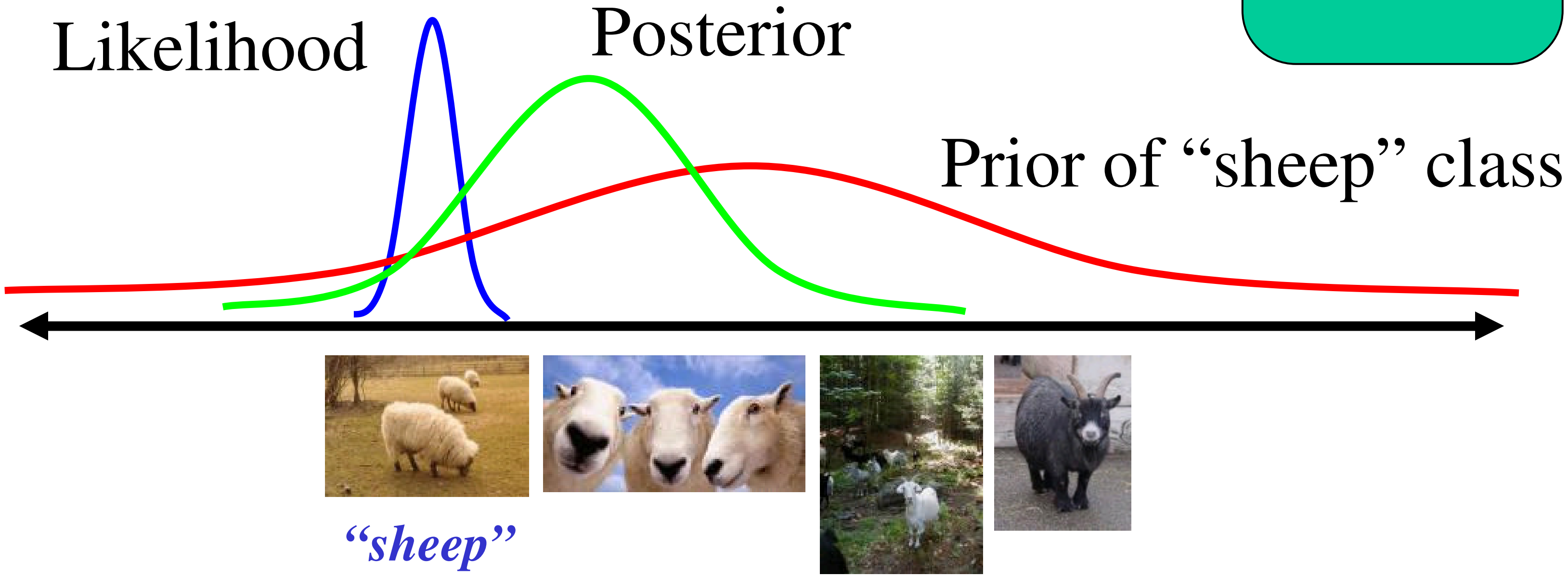


[Chenney and Forsyth, 2000]

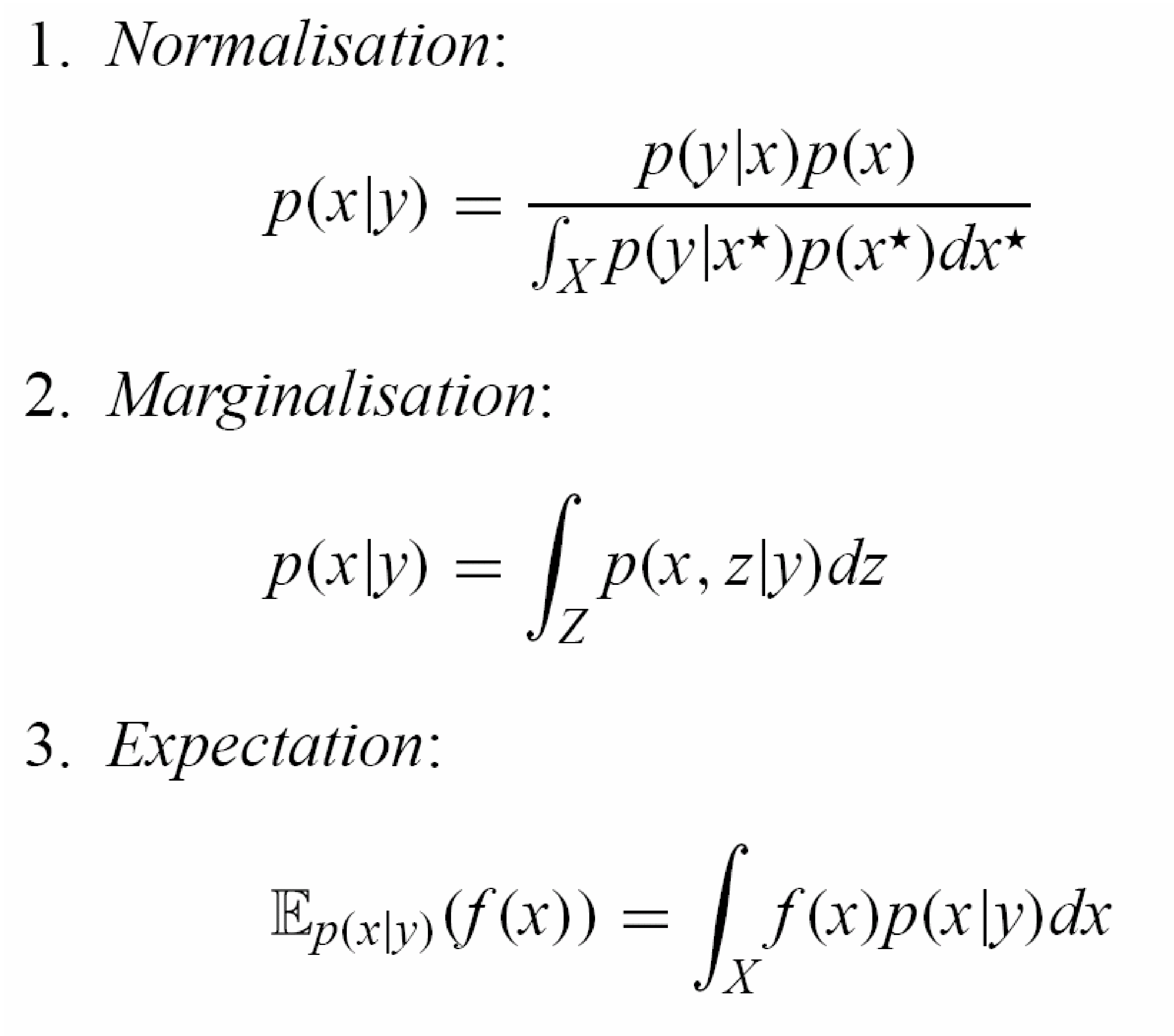
Learning and Bayesian inference

$$p(h \mid d) = \frac{p(d \mid h)\, p(h)}{\sum_{h' \in H} p(d \mid h')\, p(h')}$$





Integrals in Probabilistic Inference



1. *Normalisation*:

$$p(x|y) = \frac{p(y|x)p(x)}{\int_X p(y|x^\star)p(x^\star)dx^\star}$$

2. *Marginalisation*:

$$p(x|y) = \int_Z p(x,z|y)dz$$

3. *Expectation*:

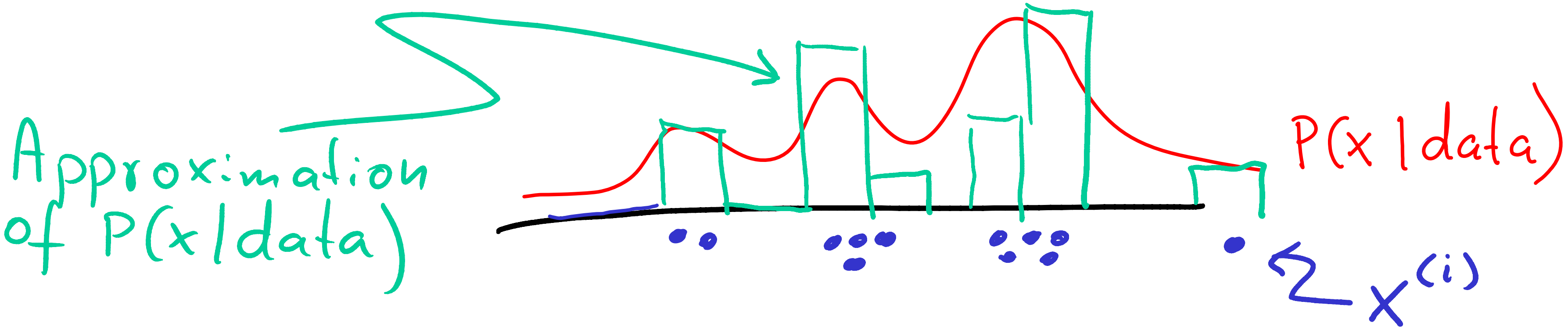
$$\mathbb{E}_{p(x|y)}(f(x)) = \int_X f(x)p(x|y)dx$$

Monte Carlo Integration

Suppose we want to compute

$$I = \int f(x)\, P(x \mid data)\, dx$$

(i) Simulate $x^{(i)}\Big|_{i=1}^{N}$ from $P(x \mid data)$



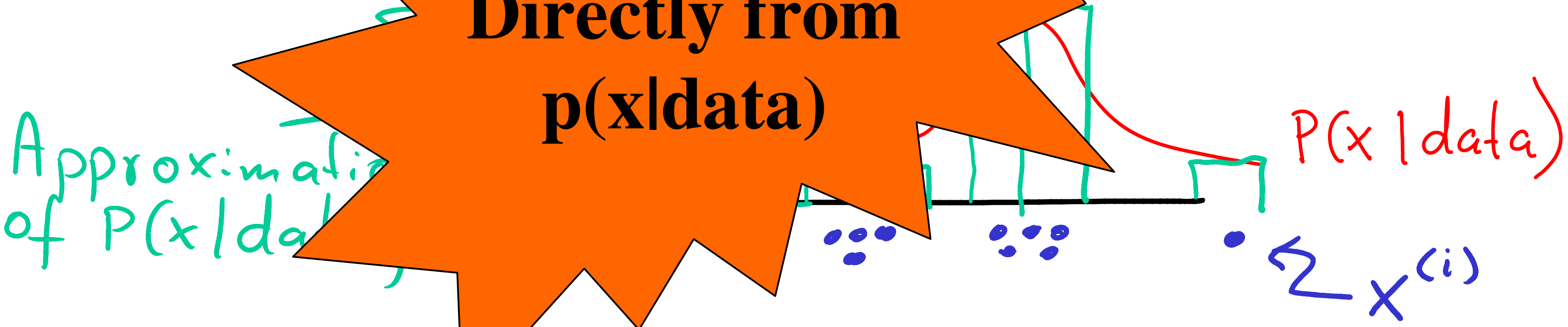
(ii) Replace nasty integral with simple sum: $I \approx \frac{1}{N} \sum_{i=1}^{N} f(x^{(i)})$ ▨

Monte Carlo Integration

Suppose we want to compute

$$I = \quad P(\quad |data)\, dx$$

(i) Simu P(x|data)



Cannot sample Directly from p(x|data)

(ii) Replace nasty integral with simple sum: $I \approx \frac{1}{N} \sum_{i=1}^{N} f(x^{(i)})$ ▨

Monte Carlo Integration Formally

The idea of Monte Carlo simulation is to draw an i.i.d. set of samples $\{x^{(i)}\}_{i=1}^{N}$ from a target density $p(x)$ defined on a high-dimensional space $\mathcal{X}$. These N samples can be used to approximate the target distribution with the following empirical point-mass function (think of it as a histogram):

$$p_N\left(dx\right) = \frac{1}{N}\sum_{i=1}^{N}\delta_{x^{(i)}}\left(dx\right),$$

where $\delta_{x^{(i)}}\left(dx\right)$ denotes the delta-Dirac mass located at $x^{(i)}$.

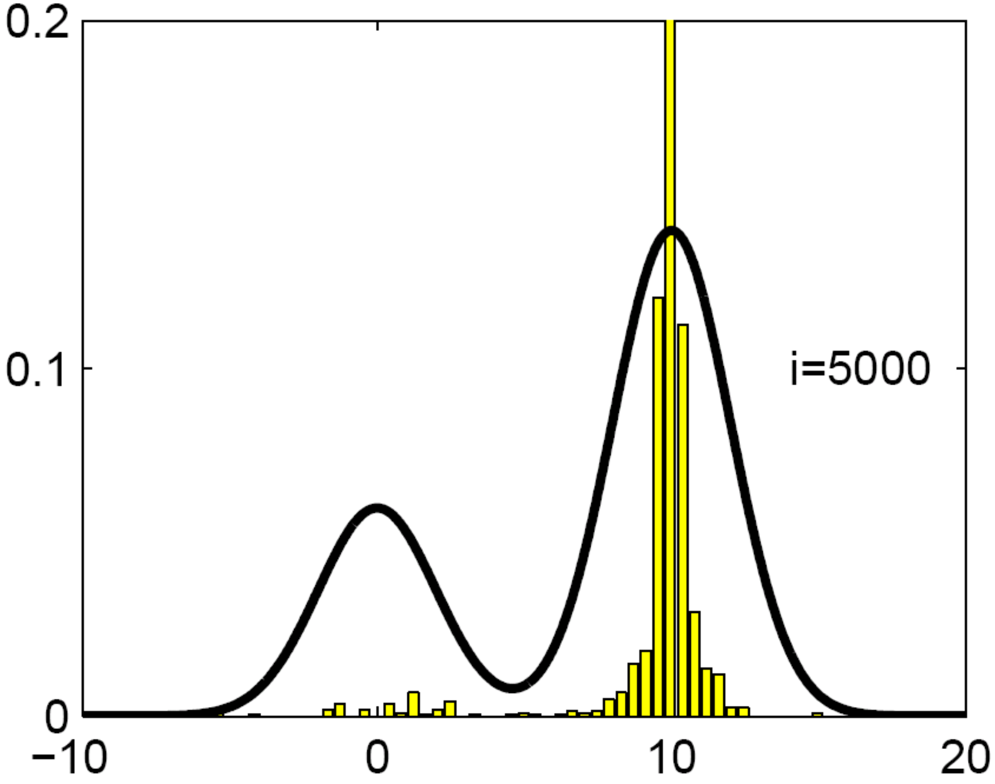
Monte Carlo Integration Formally

Consequently, one can approximate the integrals (or very large sums) $I(f)$ with tractable sums $I_N(f)$ as follows

$$I(f) = \int_{\mathcal{X}} f(x)p(x)dx.$$

Optimisation:
Concentrate Samples on Modes

$$\widehat{x} = \underset{x^{(i)};i=1,\ldots,N}{\arg\max}\ p\left(x^{(i)}\right)$$

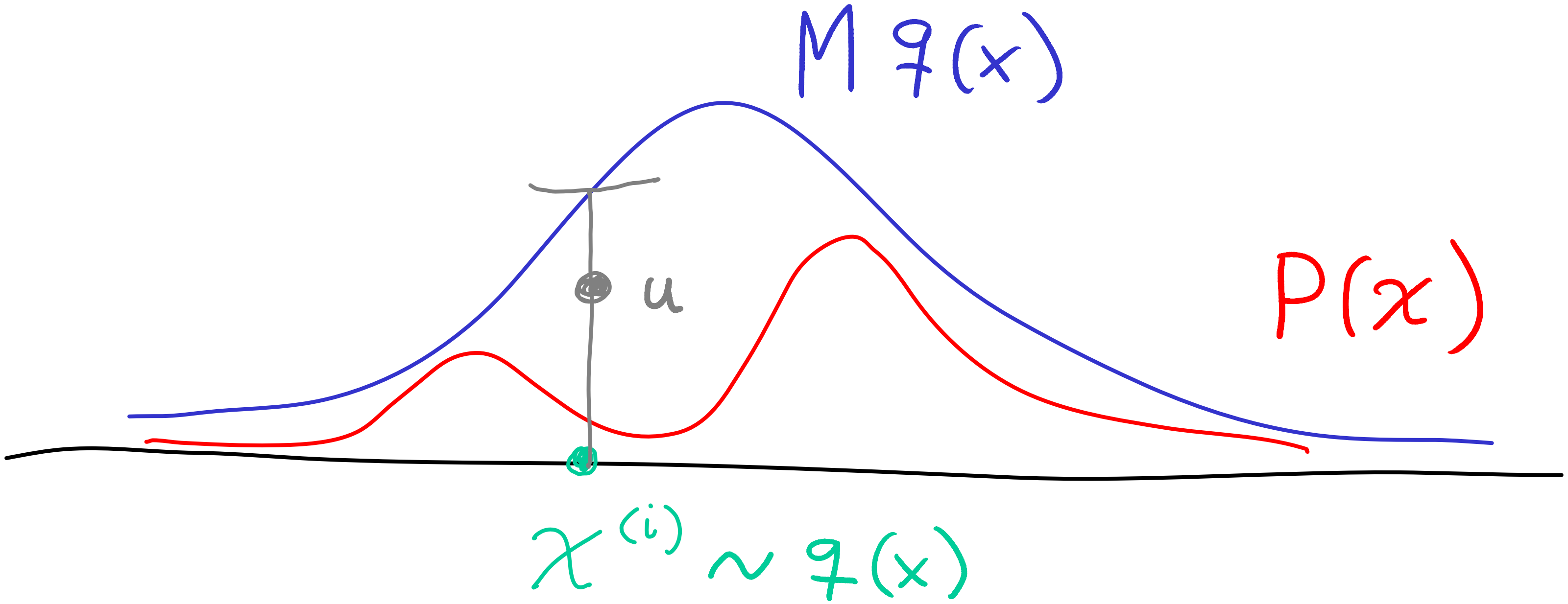


Overview

- History and applications
- Rejection sampling
- Importance sampling
- Particle filters
- Markov chain Monte Carlo
  - Metropolis-Hastings
  - Gibbs sampling
- Advanced Monte Carlo methods
  - Mixtures of kernels
  - Trans-dimensional Monte Carlo
  - Hybrid Monte Carlo
  - Monte Carlo EM
  - Particle methods with artificial dynamics

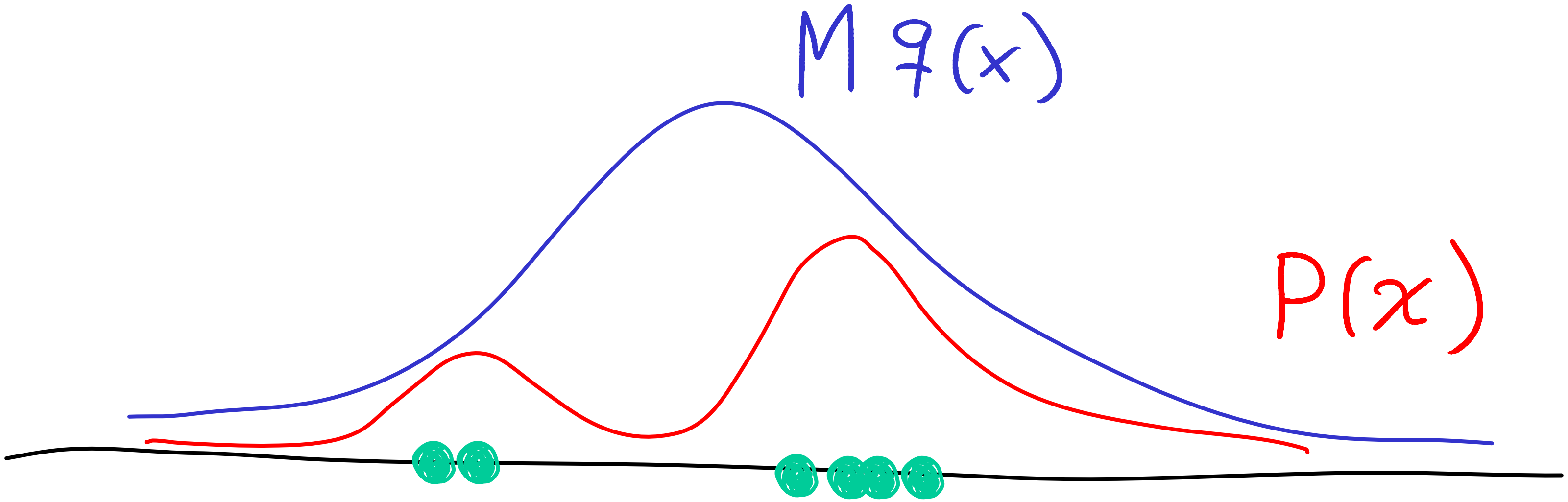
Rejection sampling

```
Set $i = 1$
Repeat until $i = N$
    1. Sample $x^{(i)} \sim q(x)$ and $u \sim \mathcal{U}_{(0,1)}$.
    2. If $u < \frac{p(x^{(i)})}{Mq(x^{(i)})}$ then accept $x^{(i)}$ and increment the
       counter $i$ by 1. Otherwise, reject.
```



Rejection sampling

```
Set $i = 1$
Repeat until $i = N$
    1. Sample $x^{(i)} \sim q(x)$ and $u \sim \mathcal{U}_{(0,1)}$.
    2. If $u < \frac{p(x^{(i)})}{Mq(x^{(i)})}$ then accept $x^{(i)}$ and increment the
       counter $i$ by 1. Otherwise, reject.
```



Overview

- History and applications
- Rejection sampling
- Importance sampling
- Particle filters
- Markov chain Monte Carlo
  - Metropolis-Hastings
  - Gibbs sampling
- Advanced Monte Carlo methods
  - Mixtures of kernels
  - Trans-dimensional Monte Carlo
  - Hybrid Monte Carlo
  - Monte Carlo EM
  - Particle methods with artificial dynamics

Importance Sampling

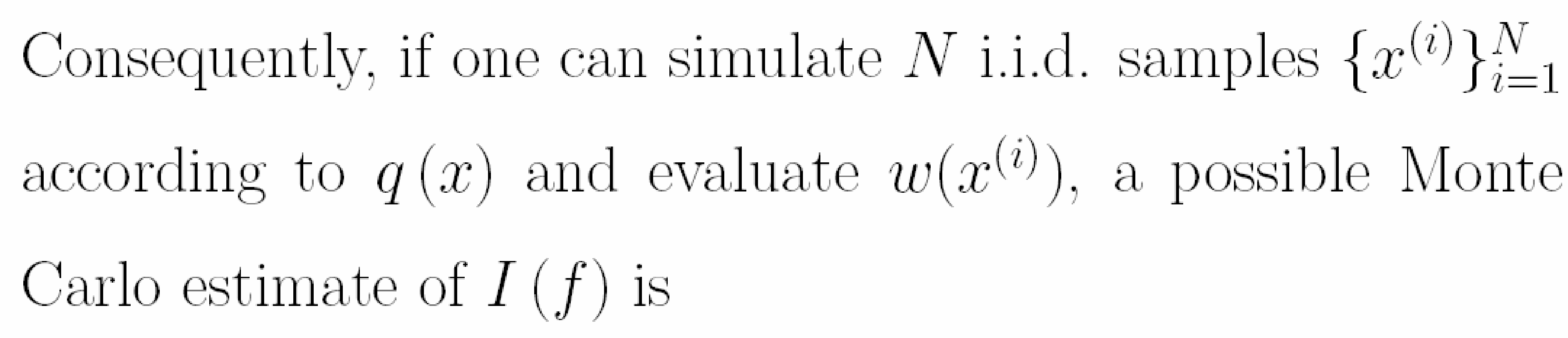
Importance sampling is a "classical" solution that goes back to the 1940's. Let us introduce an arbitrary importance proposal distribution $q(x)$ such that its support includes the support of $p(x)$ and such that we can sample from it. Then we can rewrite $I(f)$ as follows

$$I\left(f\right) = \int f\left(x\right) w\left(x\right) q\left(x\right) dx$$

where $w\left(x\right) \triangleq \frac{p(x)}{q(x)}$ is known as the *importance weight.*

Importance Sampling

Importance Sampling



Consequently, if one can simulate N i.i.d. samples $\{x^{(i)}\}_{i=1}^{N}$ according to $q\left(x\right)$ and evaluate $w(x^{(i)})$, a possible Monte Carlo estimate of $I\left(f\right)$ is

$$\widehat{I}_N\left(f\right) =$$

Importance Sampling

This estimator is unbiased and, under weak assumptions, the strong law of large numbers applies, that is $\widehat{I}_N(f) \xrightarrow[N\rightarrow\infty]{a.s.} I(f)$. It is clear that this integration method can also be interpreted as a sampling method where the posterior density $p(x)$ is approximated by:

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

Some proposal distributions $q(x)$ will obviously be preferable to others.

Normalized Importance Sampling

When the normalising constant of $p(x)$ is unknown, it is still possible to apply the importance sampling method:

Normalized Importance Sampling

The Monte Carlo estimate of $I(f)$ becomes

$$\widetilde{I}_N(f) = \frac{\frac{1}{N}\sum_{i=1}^{N} f\left(x^{(i)}\right) w(x^{(i)})}{\frac{1}{N}\sum_{j=1}^{N} w\left(x^{(i)}\right)} = \sum_{i=1}^{N} f\left(x^{(i)}\right) \widetilde{w}(x^{(i)})$$

where $\widetilde{w}(x^{(i)})$ is a normalised importance weight. For N finite, $\widetilde{I}_N(f)$ is biased (ratio of two estimates) but asymptotically, under weak assumptions, the strong law of large numbers applies, that is $\widetilde{I}_N(f) \xrightarrow[N\to\infty]{a.s.} I(f)$.

Sampling-Importance Sampling (SIR)

If one is interested in obtaining M *i.i.d.* samples from $\widehat{p}_N(x)$, then an asymptotically $(N/M \rightarrow \infty)$ valid method consists of resampling M times according to the discrete distribution $\widehat{p}_N(x)$.

Sampling-Importance Sampling (SIR)

This procedure results in M samples $\widetilde{x}^{(i)}$ with the possibility that $\widetilde{x}^{(i)} = \widetilde{x}^{(j)}$ for $i \neq j$. After resampling, the approximation of the target density is

$$\widetilde{p}_M\left(dx\right) = \frac{1}{M} \sum_{i=1}^{M} \delta_{\widetilde{x}^{(i)}}\left(dx\right)$$

Sampling-Importance Sampling (SIR)

Set $i = 1$

Repeat until $i = N$

1. Sample $x^{(i)} \sim q\left(x\right)$
2. Evaluate $p(x^{(i)}|y)$ up to a normalising constant.
3. Evaluate $q(x^{(i)})$ up to a normalising constant.
4. Compute $w(x^{(i)})$.

Normalise $w(x^{(i)})$ to obtain $\widetilde{w}(x^{(i)})$.

Resample $\{x^{(i)}, \widetilde{w}(x^{(i)})\}_{i=1}^{N} \quad \longrightarrow \quad \{\widetilde{x}^{(i)}, 1/N\}_{i=1}^{N}$

What is the best proposal?

The IS estimator is unbiased, but has variance

$$\operatorname{var}_{q(x)}\left(\widehat{I_N}(f)\right) = \mathbb{E}_{q(x)}\left(f^2(x)w^2(x)\right) - I^2(f)$$

This variance is minimised when

$$q^\star(x) = \frac{|f(x)|p(x)}{\int |f(x)|p(x)dx}$$

What is the best proposal?

Introduce parametric proposals and adapt the parameters so as to minimise the variance

$$\theta_{t+1} = \theta_t - \alpha \frac{1}{N} \sum_{i=1}^{N} f^2(x^{(i)}) w(x^{(i)}, \theta_t) \frac{\partial w(x^{(i)}, \theta_t)}{\partial \theta_t}$$

where α is a learning rate and $x^{(i)} \sim q(x, \theta)$.

Proposal distributions that adapt to the data are also very widely used.

IS Example: Logistic Regression

Given the input-output i.i.d. data sets $x \triangleq x_{1:T} \triangleq \{x_0, x_1, \ldots, x_T\}$ and $y \triangleq y_{1:T} \triangleq \{y_0, y_1, \ldots, y_T\}$, where $x_t \in \mathbb{R}$ and $y_t \in \{0, 1\}$. The idea is to come up with a model that takes a new input x_{T+1} and produces as output $p(y_{T+1} = 1|x_{T+1})$ and $p(y_{T+1} = 0|x_{T+1})$. This classification problem arises in several areas of technology, including condition monitoring and binary decision systems. For example, when monitoring patients, we might wish to decide whether they require an increase in drug intake based on new evidence.

IS Example: Logistic Regression

For practical reasons, we parameterise our model. In particular, we introduce the following Bernoulli likelihood function:

$$p(y_t|x_t,\theta) = \left[\frac{1}{1+\exp\left(-\theta x_t\right)}\right]^{y_t}\left[1-\frac{1}{1+\exp\left(-\theta x_t\right)}\right]^{1-y_t}$$

where θ are the model parameters. The logistic function $p(y_t = 1|x_t) = \frac{1}{1+\exp(-\theta x_t)}$ is conviniently bounded between 0 and 1.

IS Example: Logistic Regression

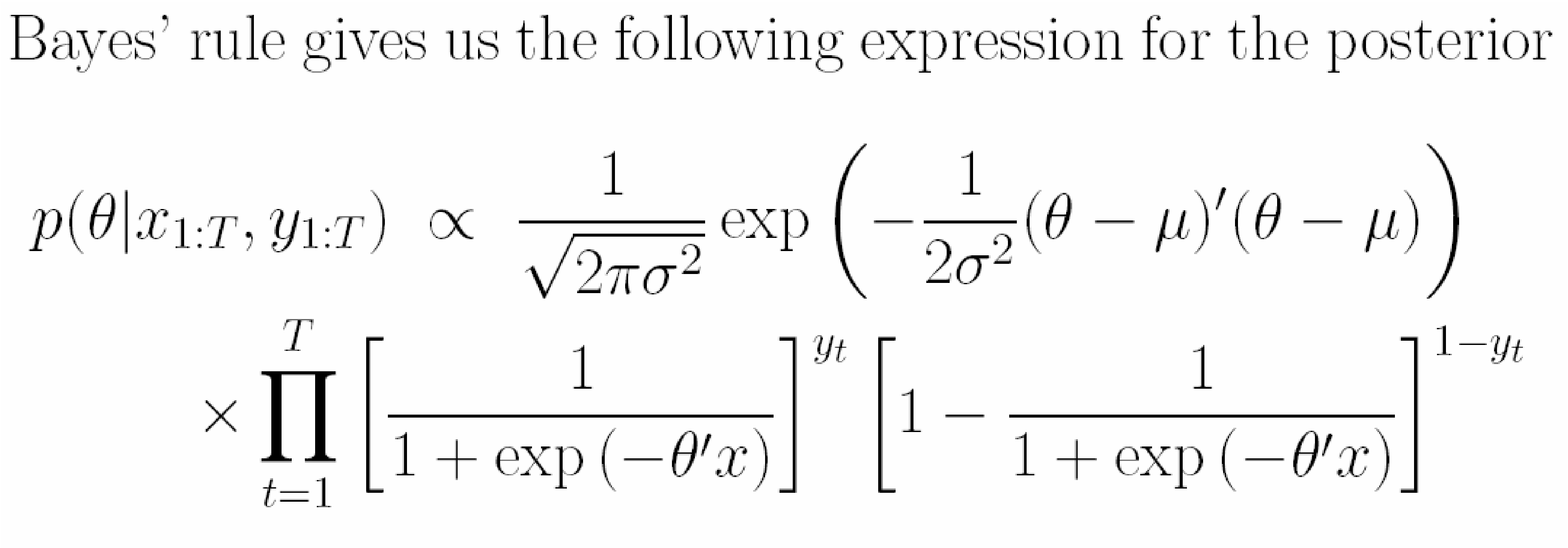
We also assume a Gaussian prior

$$p(\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(\theta - \mu)'(\theta - \mu) \right)$$

The goal of the analysis is then to compute the posterior distribution $p(\theta|x_{1:T}, y_{1:T})$. This distribution will enable us to classify new data as follows

$$p(y_{T+1}|x_{1:T+1}) = \int_{\Theta} p(y_{T+1}|x_{T+1}, \theta)p(\theta|x_{1:T}, y_{1:T})d\theta$$

IS Example: Logistic Regression



Bayes' rule gives us the following expression for the posterior

$$p(\theta|x_{1:T}, y_{1:T}) \;\propto\; \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(\theta-\mu)'(\theta-\mu)\right)$$
$$\times \prod_{t=1}^{T} \left[\frac{1}{1+\exp\left(-\theta' x\right)}\right]^{y_t} \left[1-\frac{1}{1+\exp\left(-\theta' x\right)}\right]^{1-y_t}$$

IS Example: Logistic Regression

$$p(y_t|x_t,\theta) = \left[\frac{1}{1+\exp\left(-\theta x_t\right)}\right]^{y_t}\left[1-\frac{1}{1+\exp\left(-\theta x_t\right)}\right]^{1-y_t}$$

$$p(\theta) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}(\theta-\mu)'(\theta-\mu)\right)$$

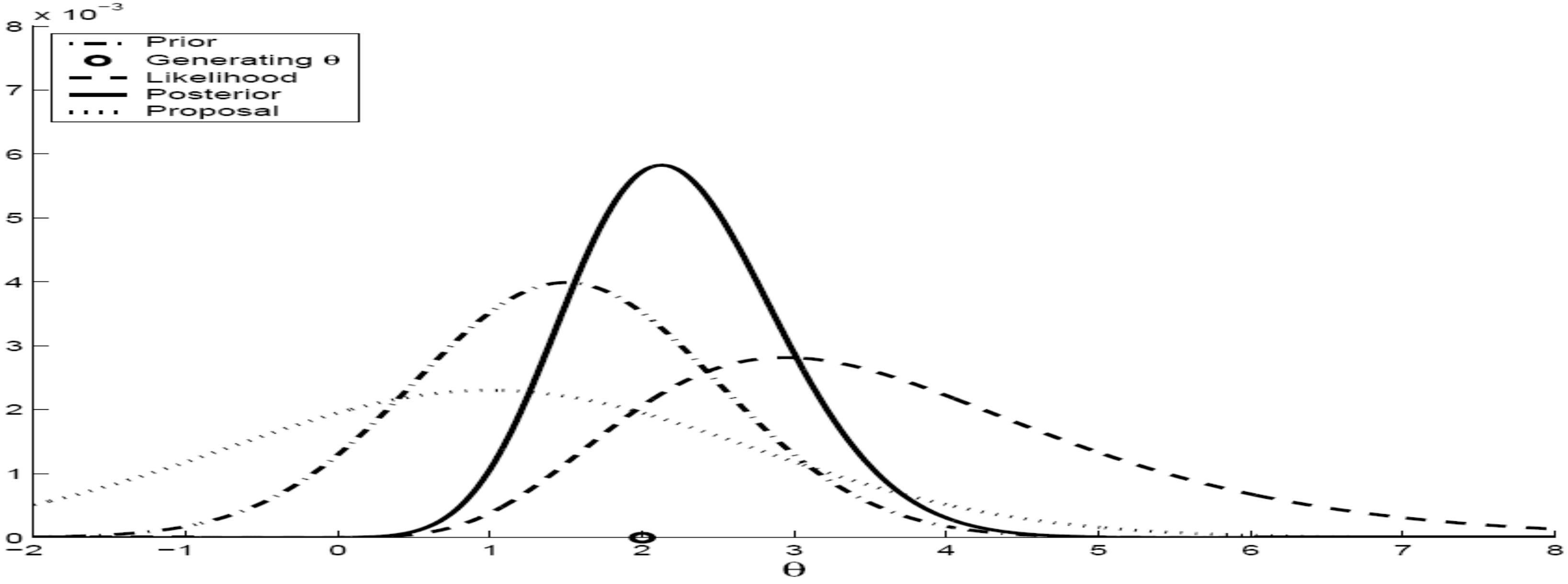
$$\begin{aligned} p(\theta|x_{1:T},y_{1:T}) \quad &\propto \quad \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}(\theta-\mu)'(\theta-\mu)\right) \\ &\times\prod_{t=1}^{T}\left[\frac{1}{1+\exp\left(-\theta' x\right)}\right]^{y_t}\left[1-\frac{1}{1+\exp\left(-\theta' x\right)}\right]^{1-y_t} \end{aligned}$$

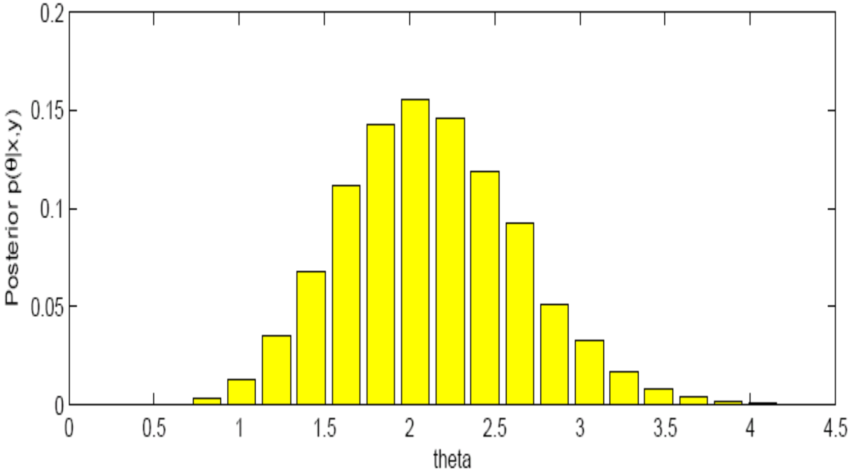
$x_t \in \mathbb{R}$ and $y_t \in \{0,1\}$

IS Example: Logistic Regression

The problem is that in this case we can't solve the normalising integral analytically. So we have to use numerical methods — in this case importance sampling — to approximate $p(\theta|x_{1:T}, y_{1:T})$. Note that we cannot sample from $p(\theta|x_{1:T}, y_{1:T})$ directly because we don't know the normalising constant. So instead we sample from a proposal distribution $q(\theta)$ (say a Gaussian) and weight the samples using importance sampling. After obtaining N samples of θ from the posterior, we can classify new data as follows

IS Example: Logistic Regression

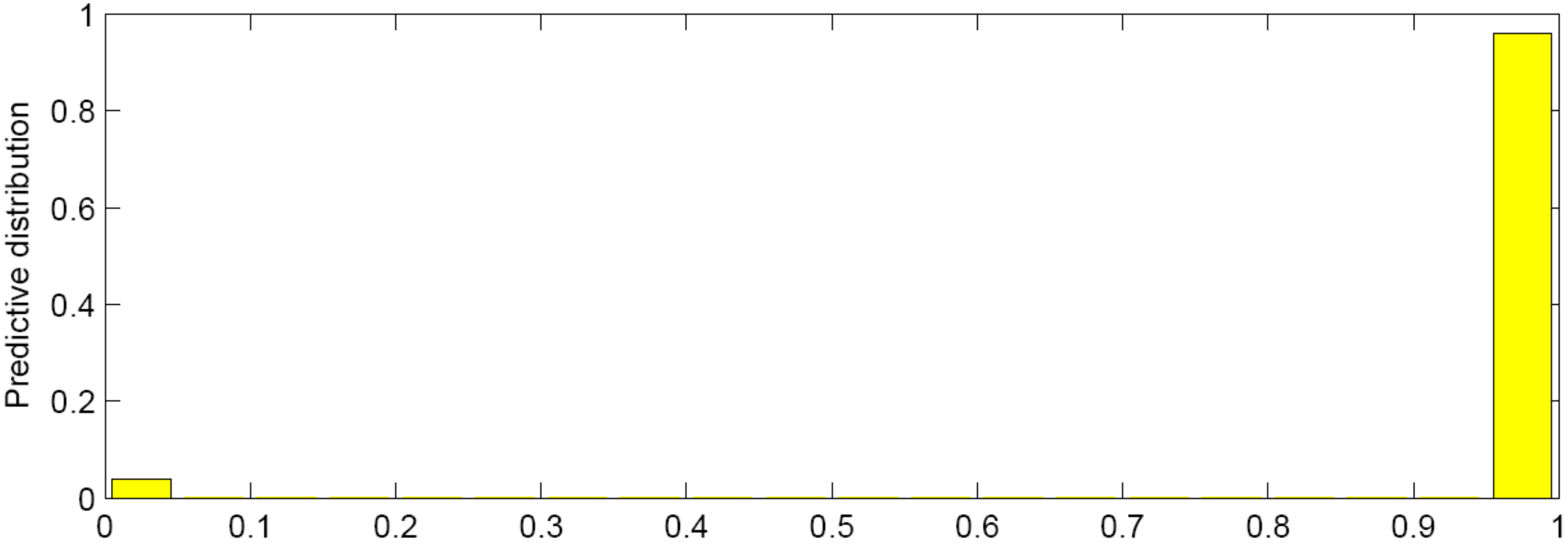




IS Example: Logistic Regression

$$p(y_{T+1}|x_{1:T+1}) = \int_{\Theta} p(y_{T+1}|x_{T+1},\theta)p(\theta|x_{1:T},y_{1:T})d\theta$$

$$p(y_{T+1}|x_{1:T+1}) = \frac{1}{N}\sum_{i=1}^{N} p(y_{T+1}|x_{T+1},\theta^{(i)})$$



Overview

- History and applications
- Rejection sampling
- Importance sampling
- Particle filters
- Markov chain Monte Carlo
  - Metropolis-Hastings
  - Gibbs sampling
- Advanced Monte Carlo methods
  - Mixtures of kernels
  - Trans-dimensional Monte Carlo
  - Hybrid Monte Carlo
  - Monte Carlo EM
  - Particle methods with artificial dynamics

Dynamic models and particle filtering

- Unknown states: $\mathbf{x}_{0:t} = \{\mathbf{x}_0, \ldots, \mathbf{x}_t\}$.
- Observations: $\mathbf{y}_{1:t} = \{\mathbf{y}_1, \ldots, \mathbf{y}_t\}$.
- Model:

$$\begin{aligned} & p(\mathbf{x}_0) & \\ & p(\mathbf{x}_t | \mathbf{x}_{t-1}) & \text{for } t \geq 1 \\ & p(\mathbf{y}_t | \mathbf{x}_t) & \text{for } t \geq 1 \end{aligned}$$

Dynamic models and particle filtering

Stochastic Volatility

$$\begin{aligned}
y_t &= e^{\alpha_t/2}\sigma_t\varepsilon_t \\
\log\sigma_t^2 &= \beta\log\sigma_{t-1}^2 + v_t \\
\alpha_t &= \alpha_{t-1} + u_{t1} \\
\beta_t &= \beta_{t-1} + u_{t2}
\end{aligned}$$

Dynamic models and particle filtering

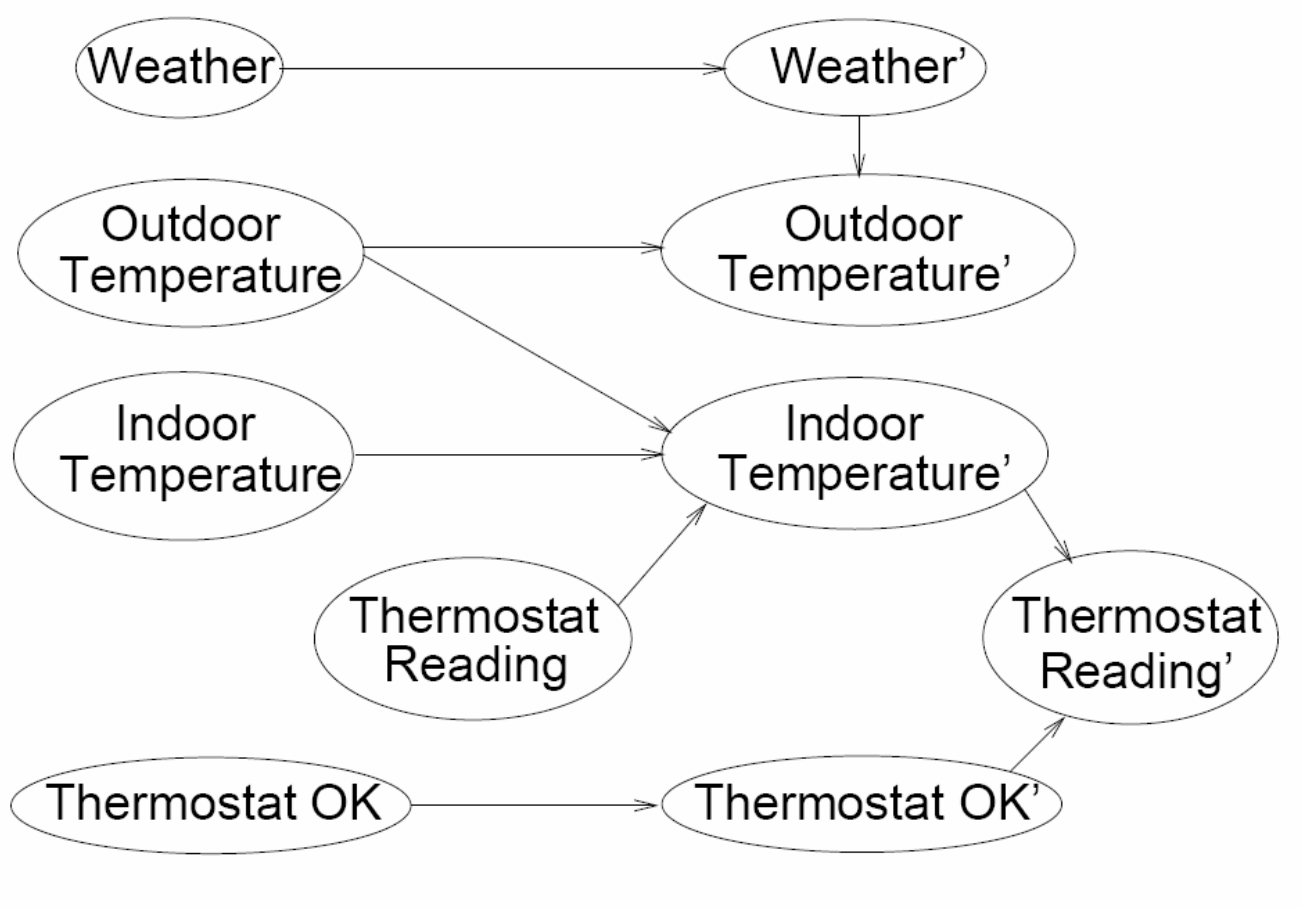
Target Tracking

$$\begin{pmatrix} s_{t,1} \\ s_{t,2} \\ v_{t,1} \\ v_{t,2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s_{t-1,1} \\ s_{t-1,2} \\ v_{t-1,1} \\ v_{t-1,2} \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \epsilon_{t,1} \\ \epsilon_{t,2} \end{pmatrix}$$

$$\begin{pmatrix} y_{t,1} \\ y_{t,2} \end{pmatrix} = \begin{pmatrix} s_{t,1} \\ s_{t,2} \end{pmatrix} + \begin{pmatrix} e_{t,1} \\ e_{t,2} \end{pmatrix}$$

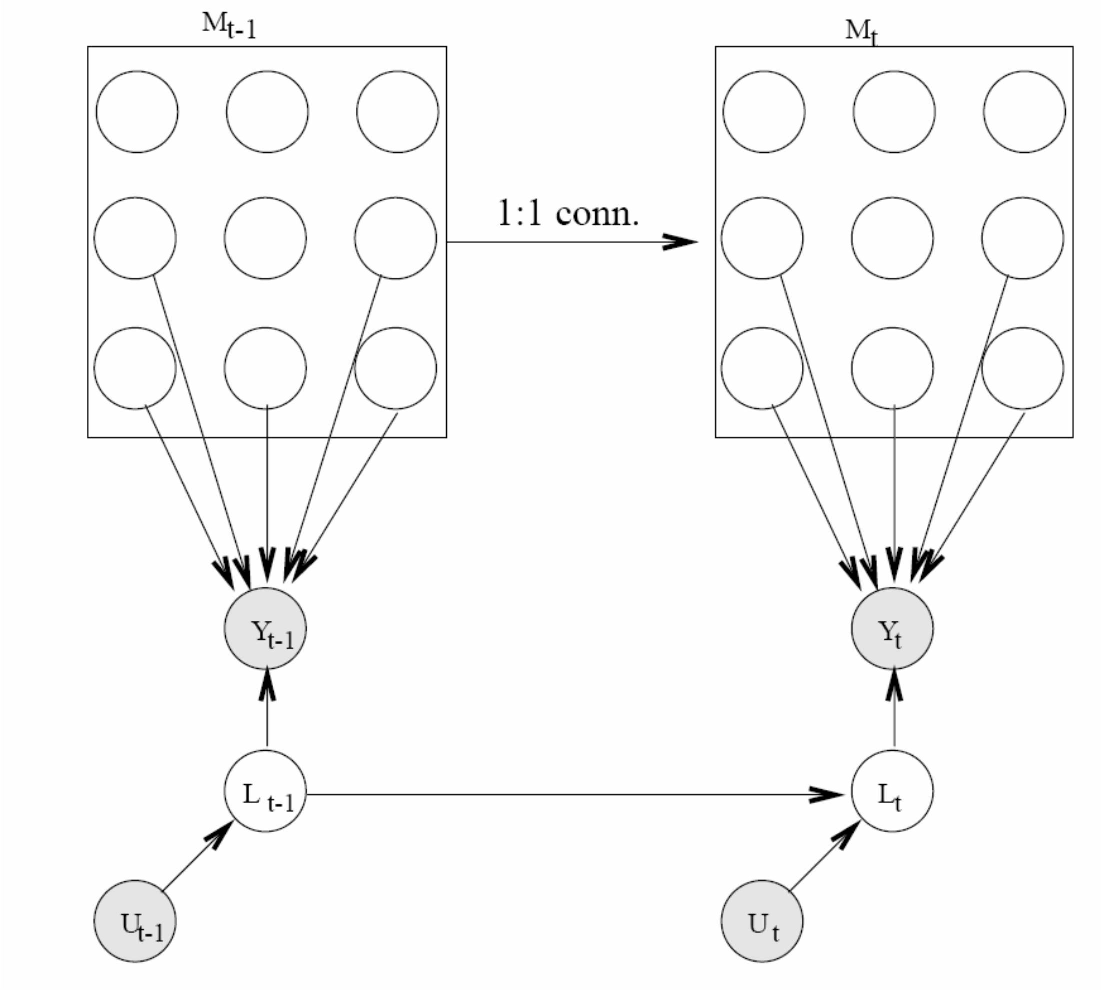
Dynamic models and particle filtering

Dynamic Bayesian Networks



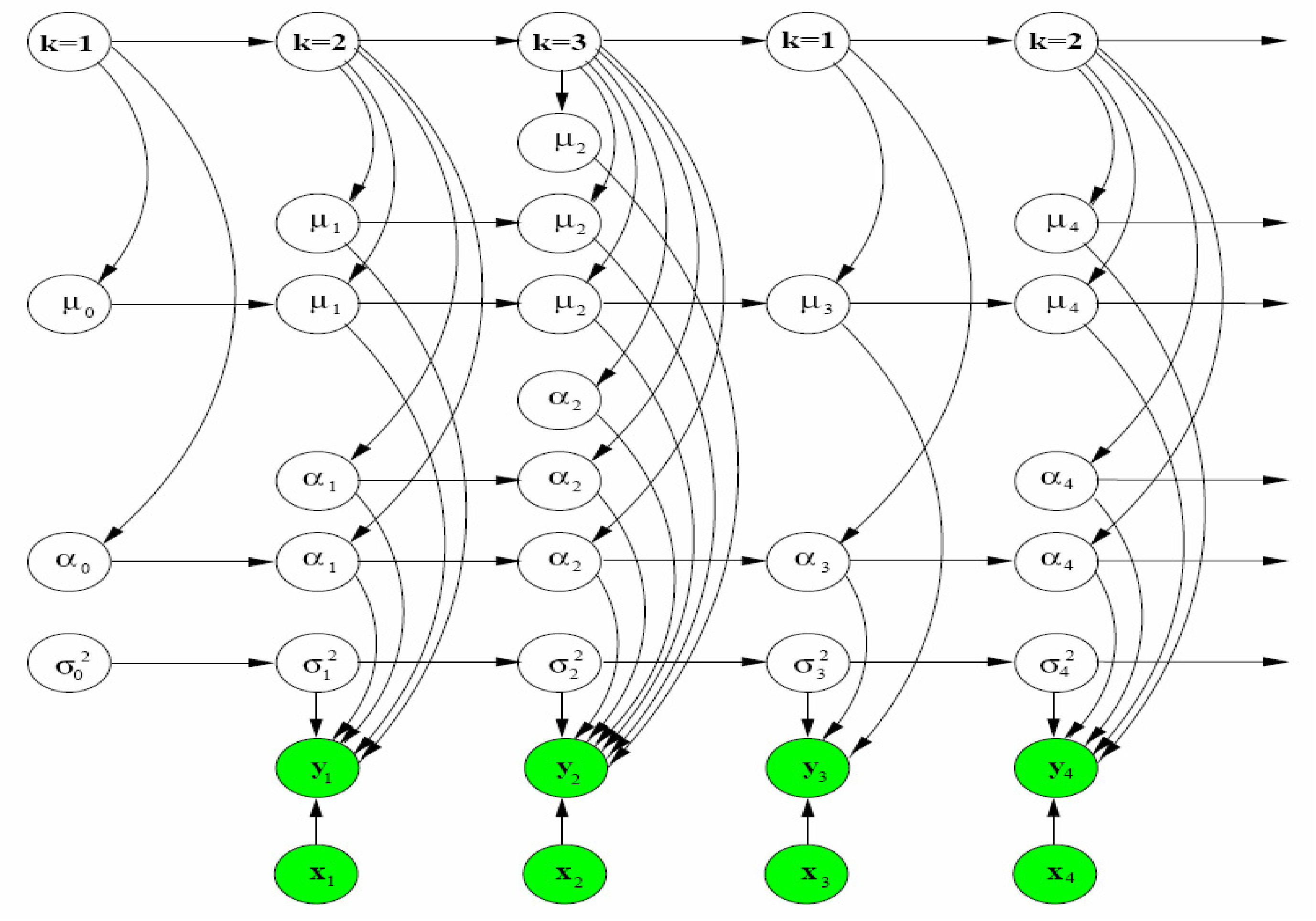
Dynamic models and particle filtering

Robot Localisation and Map Learning

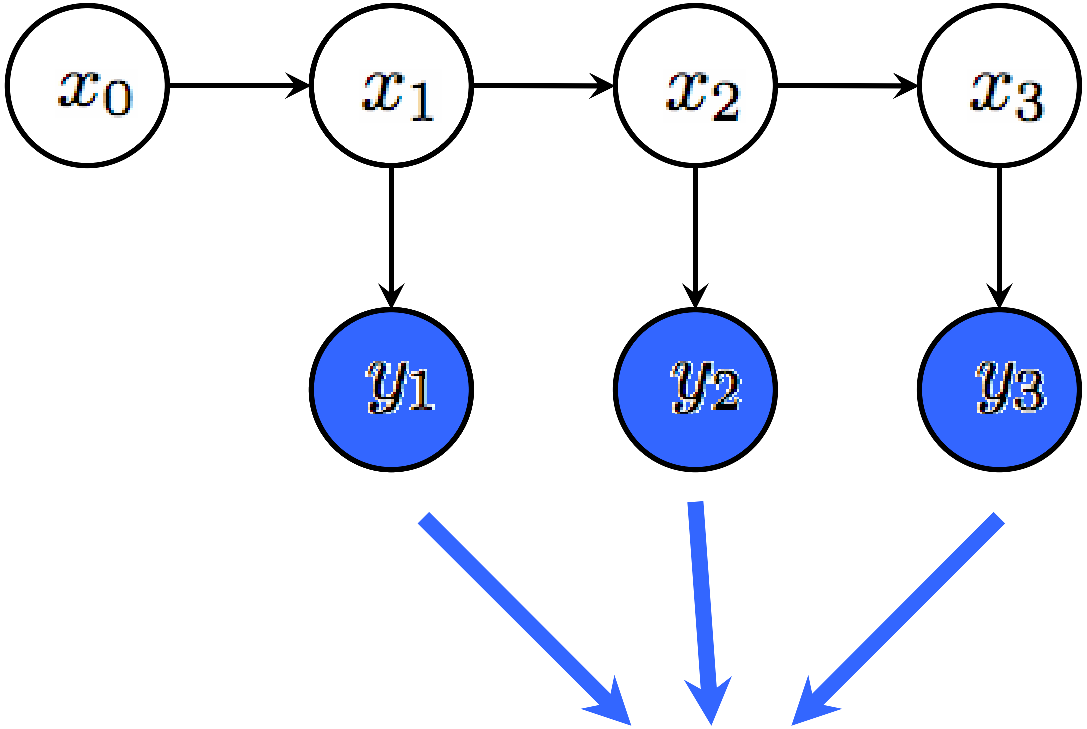


Dynamic models and particle filtering

Model Selection



Non-linear non-Gaussian filtering



$$p(x_3|y_{1:3})$$

$$p(x_t|y_{1:t}) \propto p(y_t|x_t) \int p(x_t|x_{t-1})p(x_t|y_{1:t-1})dx_{t-1}$$

Nasty integral

Importance Sampling for Optimal Filtering / Tracking

- Input: $p\left(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}\right)$
- Prediction:

$$p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}\right) = \int p\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right) p\left(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}\right) d\mathbf{x}_{t-1}$$

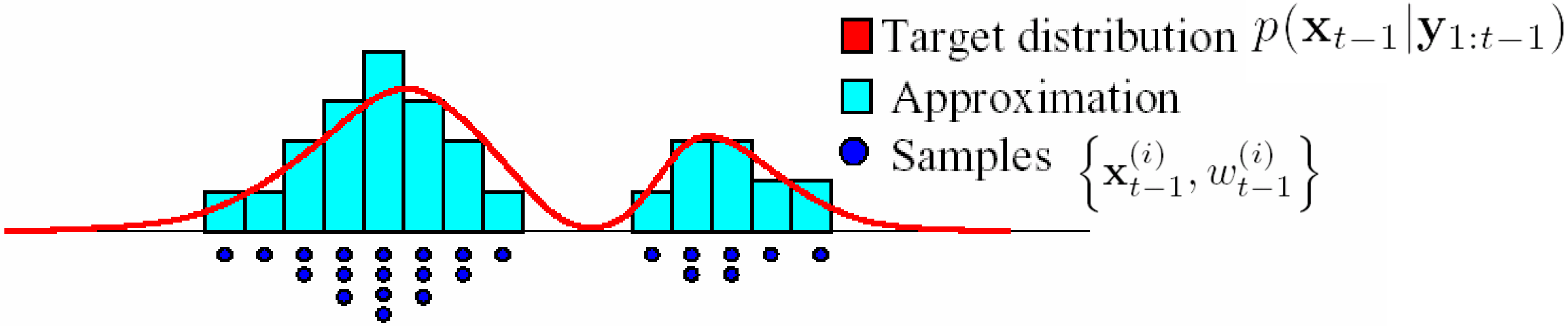
- Bayes update:

$$p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t}\right) = \frac{p\left(\mathbf{y}_t \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}\right)}{\int p\left(\mathbf{y}_t \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}\right) d\mathbf{x}_t}$$

- Output: $p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t}\right)$

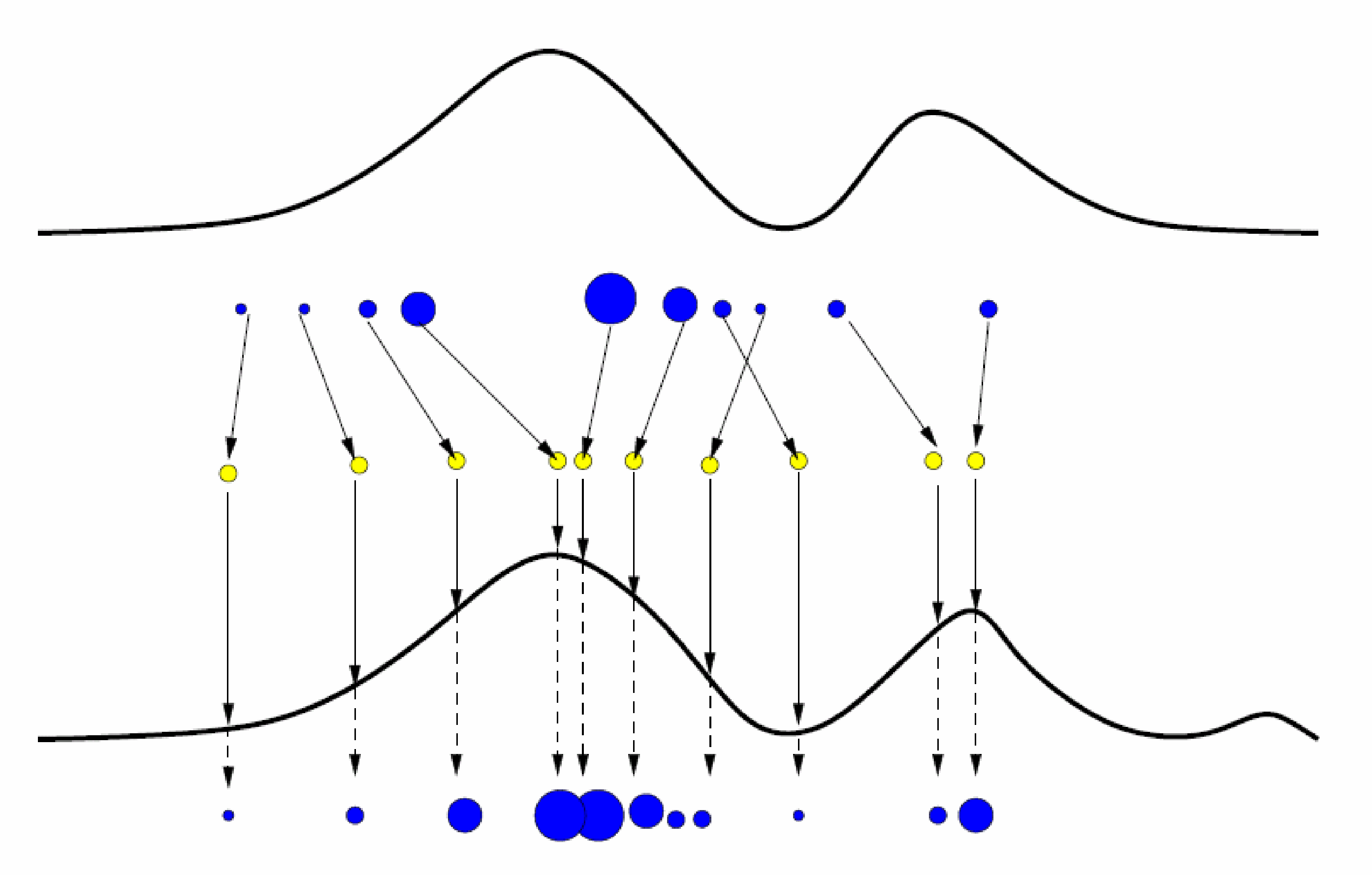
One can Compute the Integrals Recursively in Time

Given the samples $\left\{ \mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)} \right\}$ from $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$

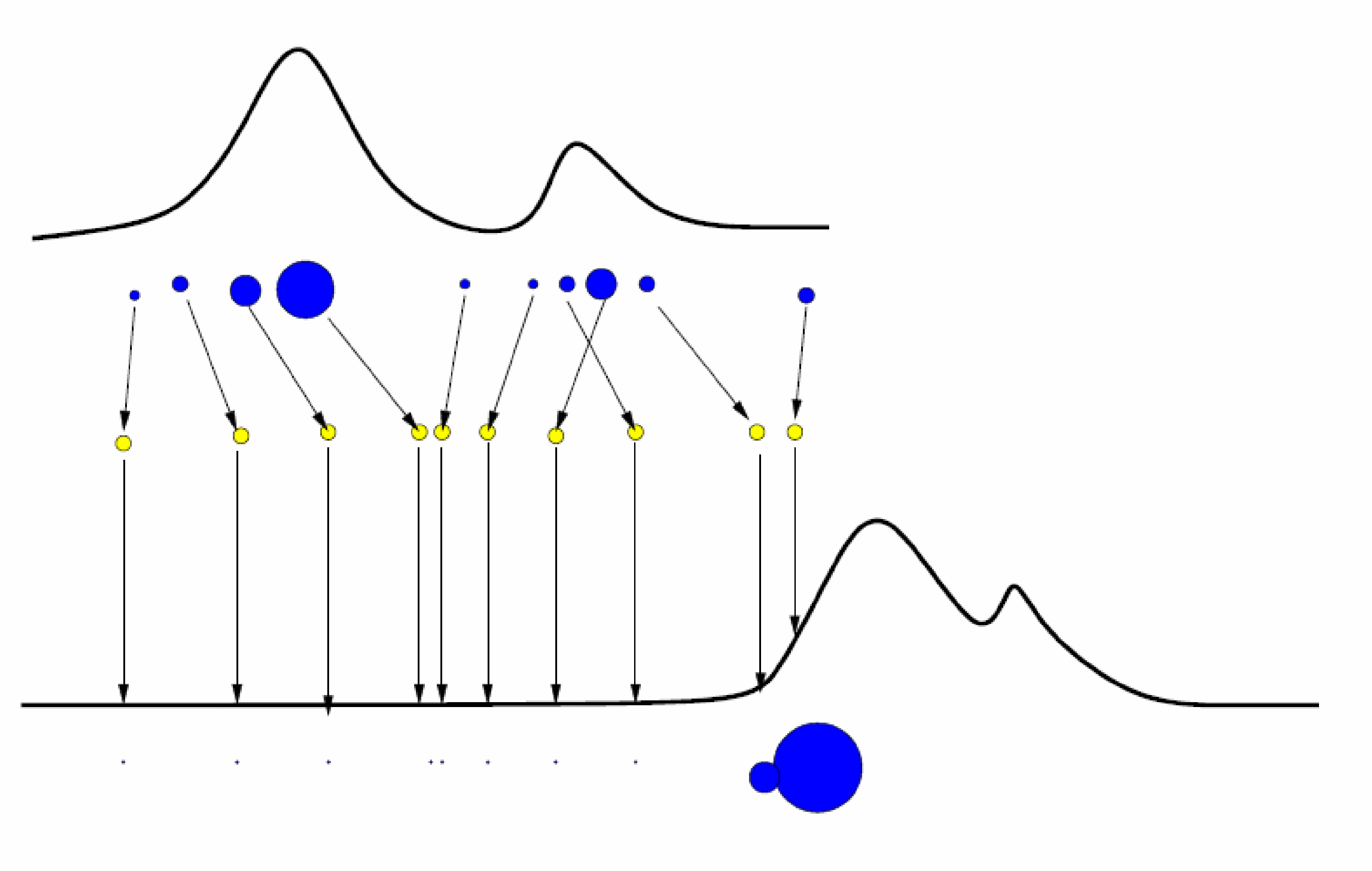


$$\int p(\mathbf{x}_t|\mathbf{x}_{t-1})\,p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\,d\mathbf{x}_{t-1} \rightarrow \sum_{j=1}^{N} w_{t-1}^{(j)} p\left(\mathbf{x}_t \middle| \mathbf{x}_{t-1}^{(j)}\right)$$

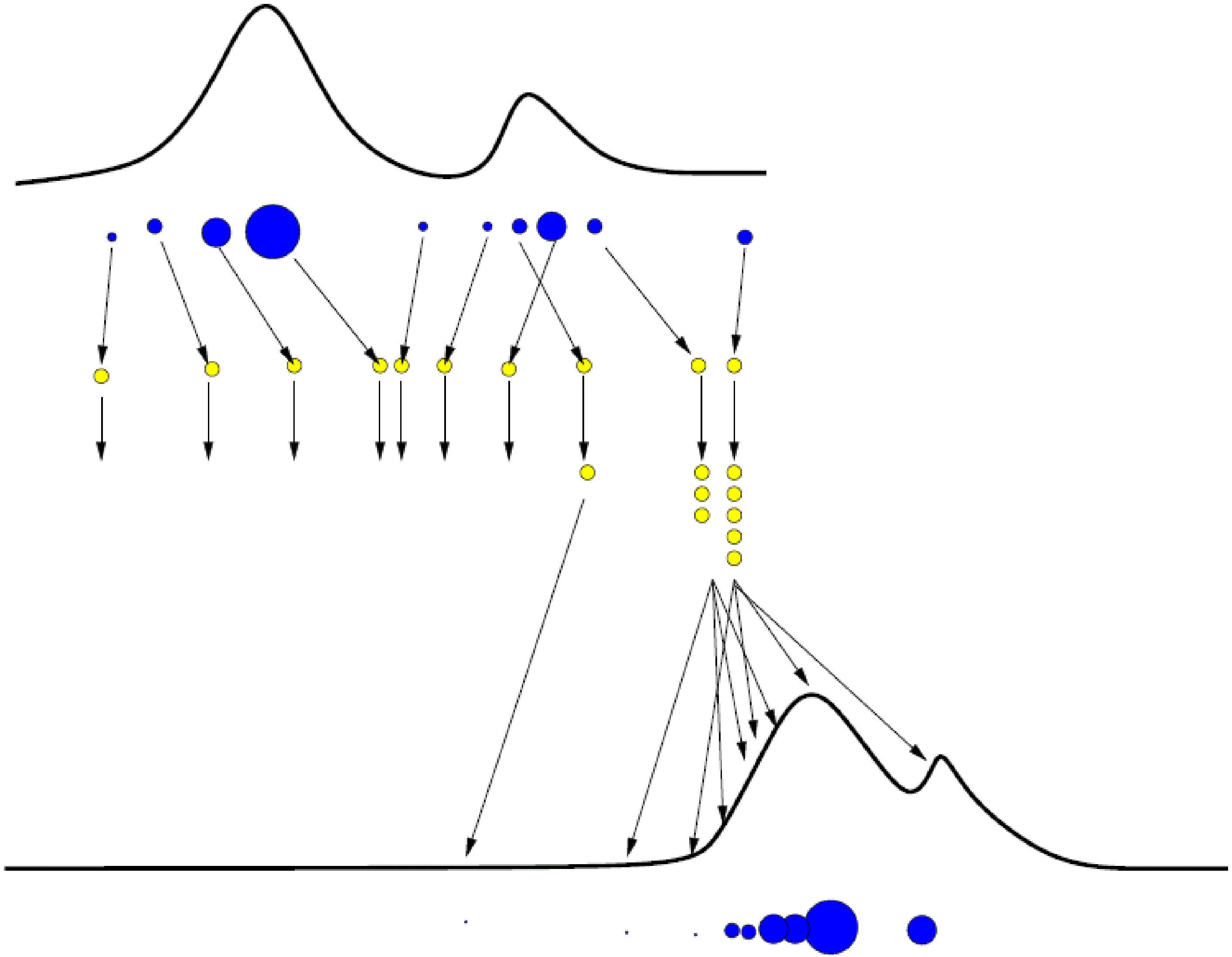
Particle Filtering (SIS)



Particle Filtering (SIS)



Particle Filtering (SIR)



Particle Filtering Code

- For $i = 1, ..., N$, sample $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$ and set $t = 1$.

- For $i = 1, ..., N$, sample $\widetilde{\mathbf{x}}_t^{(i)} \sim p\left(\mathbf{x}_t \middle| \mathbf{x}_{t-1}^{(i)}\right)$.
- For $i = 1, ..., N$, evaluate the importance weights

$$\widetilde{w}_t^{(i)} = p\left(\mathbf{y}_t \middle| \widetilde{\mathbf{x}}_t^{(i)}\right)$$

- Normalise the importance weights.
- Select fittest samples (black-box).

Particle Filtering Example

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

where $x_0 \sim \mathcal{N}\left(0, \sigma_1^2\right)$, v_t and w_t are mutually independent white Gaussian noises, $v_t \sim \mathcal{N}\left(0, \sigma_v^2\right)$ and $w_t \sim \mathcal{N}\left(0, \sigma_w^2\right)$

Particle Filtering Example

- For $i = 1, ..., N$, sample $\mathbf{x}_0^{(i)} \sim \mathcal{N}\left(0, \sigma_1^2\right)$
- For $i = 1, ..., N$, sample

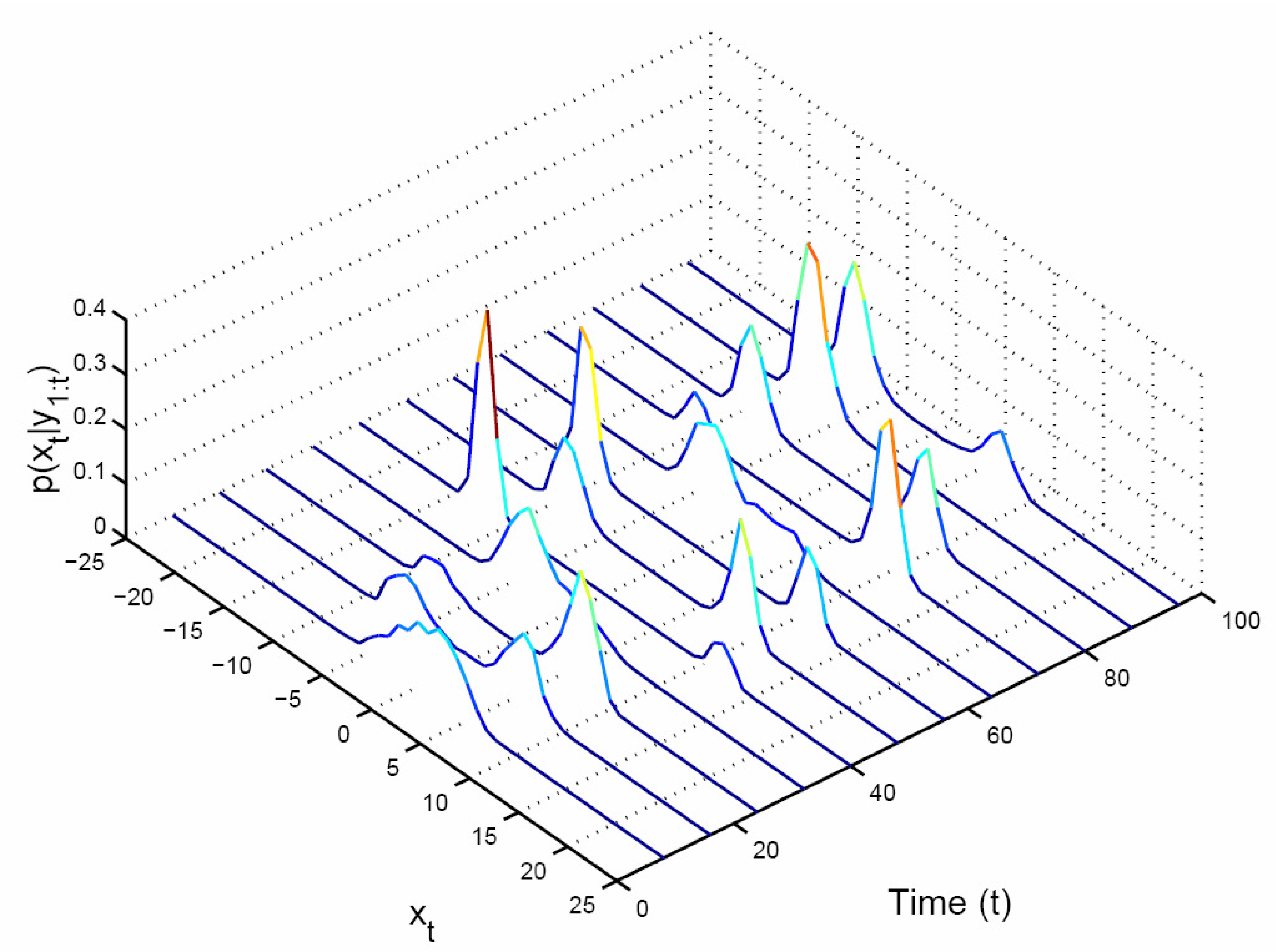
$$x_t^{(i)} = \frac{1}{2}x_{t-1}^{(i)} + 25\frac{x_{t-1}^{(i)}}{1 + x_{t-1}^{2(i)}} + 8\cos\left(1.2t\right) + \mathcal{N}\left(0, \sigma_v^2\right)$$

- For $i = 1, ..., N$, evaluate the importance weights

$$\widetilde{w}_t^{(i)} = \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{1}{2\sigma_w^2}\left(y - \frac{x_t^{2(i)}}{20}\right)^2}$$

- Normalise the importance weights.
- Resample fittest samples (black-box).

Particle Filtering Example



Particle Methods More Generally

The goal is to approximate a target distribution over a sequence of states $\mathbf{x}_{1:n} \triangleq \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ that is growing with "time" as well as the partition function.

$$\pi_n(\mathbf{x}_{1:n}) = Z_n^{-1} f_n(\mathbf{x}_{1:n}) \qquad Z_n \triangleq \int f_n(\mathbf{x}_{1:n}) d\mathbf{x}_{1:n}$$

We do this using sequential importance sampling (M&U, 49)

$$w_n = \frac{f_n(\mathbf{x}_{1:n})}{q_n(\mathbf{x}_{1:n})} = \frac{f_n(\mathbf{x}_{1:n})}{f_{n-1}(\mathbf{x}_{1:n-1})} \frac{1}{q(\mathbf{x}_n | \mathbf{x}_{1:n-1})} w_{n-1}$$

e.g. For filtering, we use: $f_n(\mathbf{x}_{1:n}) = \prod_{t=1}^{n} p(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{y}_t | \mathbf{x}_t)$

Example 2: Quantum Monte Carlo

$$\left(-\frac{1}{2}\sum_{i=1}^{N}\nabla_i^2+\sum_{i=1}^{N}v(\mathbf{r}_i)+\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\(j\neq i)}}^{N}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|}\right)\Psi(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_N)=E\Psi(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_N)$$

$$\int\mu(\mathbf{x})K(\mathbf{y}|\mathbf{x})d\mathbf{x}=\lambda\mu(\mathbf{y})$$

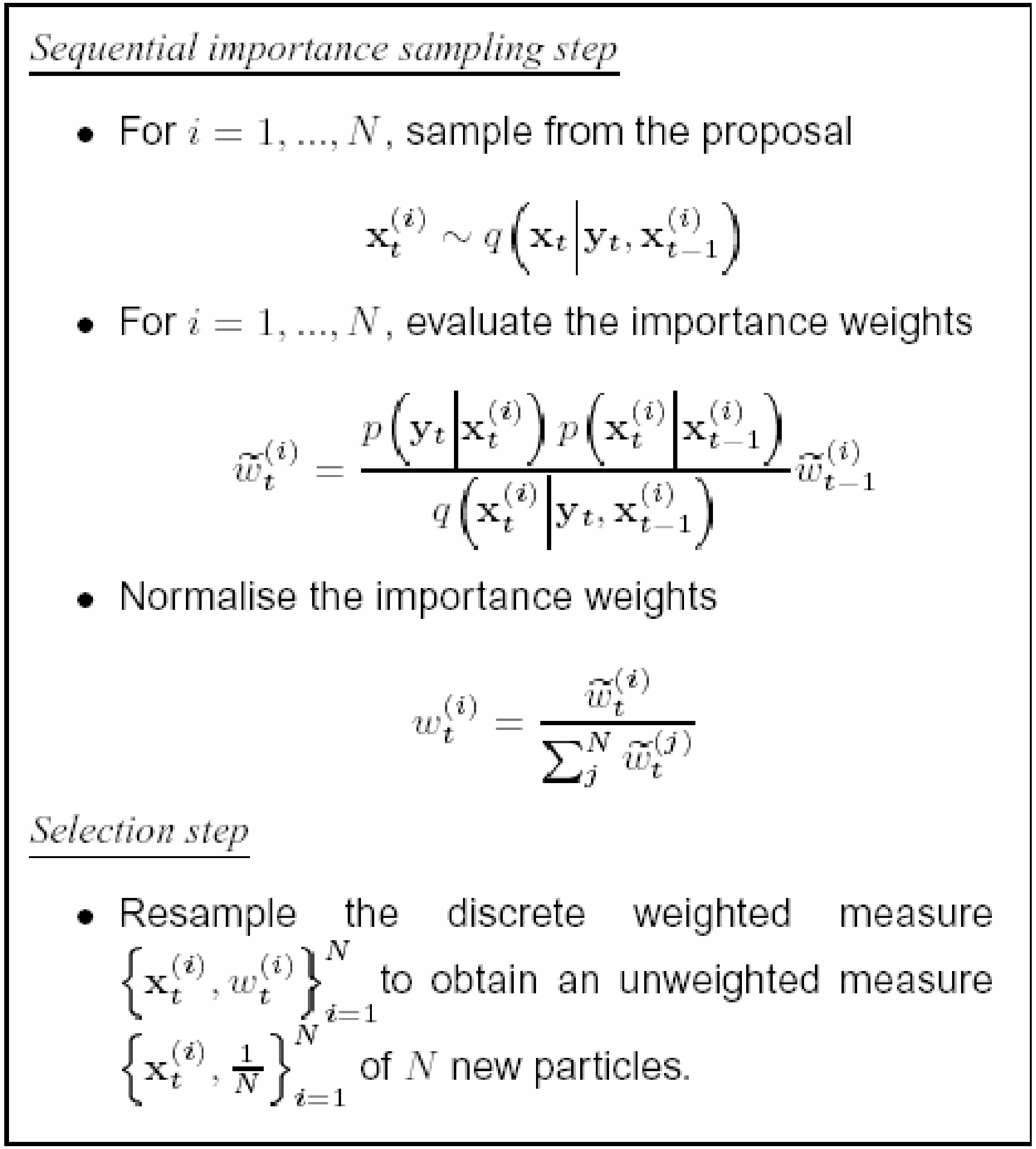
We use a particle implementation of the **power method** (time = iterations = kernel “multiplications”)

$$\int\cdots\int \underbrace{v(\mathbf{x}_1)\prod_{k=2}^{n}K(\mathbf{x}_k|\mathbf{x}_{k-1})}_{f_n(\mathbf{x}_{1:n})}d\mathbf{x}_{1:n-1}\approx c_1\lambda_1^n\mu(\mathbf{x}_n)$$

The target distribution (eigenfunction) and ground energy (eigenvalue) can be easily computed

$$\lambda_1=\frac{Z_n}{Z_{n-1}}$$

Particle Filtering



Sequential importance sampling step

- For $i = 1, ..., N$, sample from the proposal

$$\mathbf{x}_t^{(i)} \sim q\left(\mathbf{x}_t \middle| \mathbf{y}_t, \mathbf{x}_{t-1}^{(i)}\right)$$

- For $i = 1, ..., N$, evaluate the importance weights

$$\widetilde{w}_t^{(i)} = \frac{p\left(\mathbf{y}_t \middle| \mathbf{x}_t^{(i)}\right) p\left(\mathbf{x}_t^{(i)} \middle| \mathbf{x}_{t-1}^{(i)}\right)}{q\left(\mathbf{x}_t^{(i)} \middle| \mathbf{y}_t, \mathbf{x}_{t-1}^{(i)}\right)} \widetilde{w}_{t-1}^{(i)}$$

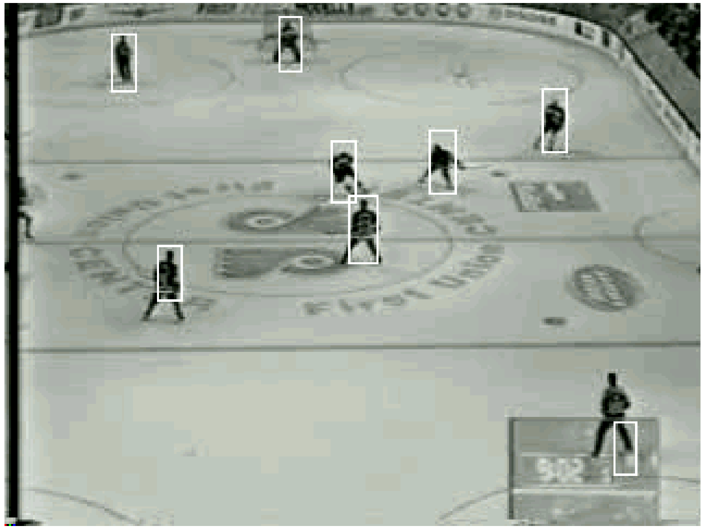
- Normalise the importance weights

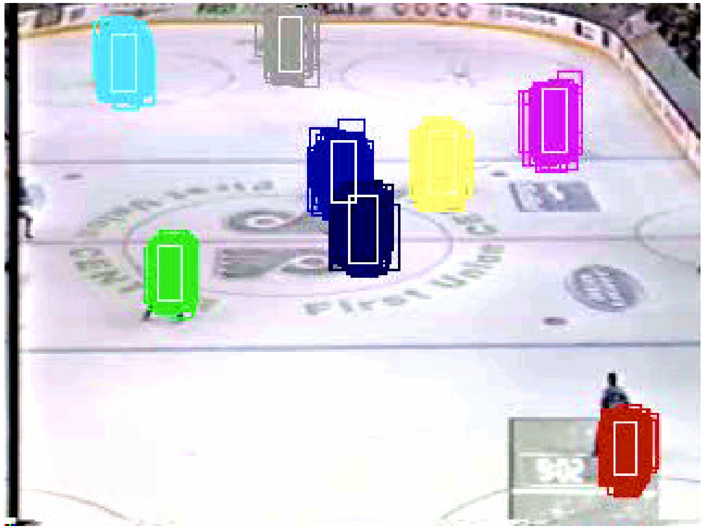
$$w_t^{(i)} = \frac{\widetilde{w}_t^{(i)}}{\sum_j^N \widetilde{w}_t^{(j)}}$$

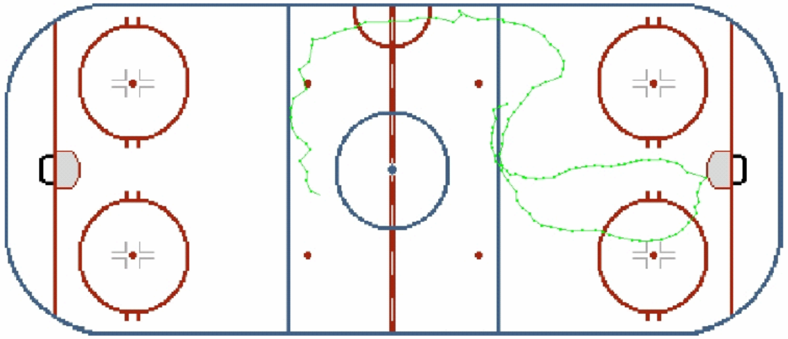
Selection step

- Resample the discrete weighted measure $\left\{\mathbf{x}_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N}$ to obtain an unweighted measure $\left\{\mathbf{x}_t^{(i)}, \frac{1}{N}\right\}_{i=1}^{N}$ of N new particles.

Using Clever Proposals: e.g. Boosting

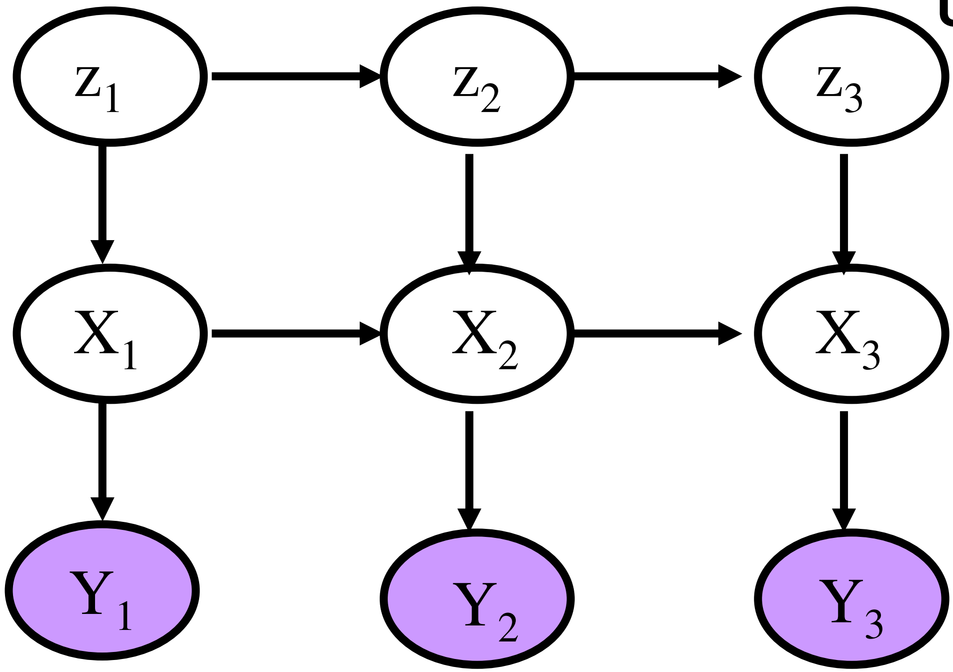






Using Clever Proposals: e.g. Boosting

Autonomous robots and self-diagnosis



Unknown internal discrete state

Unknown continuous signals

Sensor readings

Rao-Blackwellised Particle Filtering

- Robot gathers **observations** $y_t \in \mathbb{R}^{n_y}$ one-at-a-time using internal and external sensors.

- Robot has internal **continuous states** $x_t \in \mathbb{R}^{n_x}$.

- Robot has internal **discrete states** $z_t \in \mathcal{Z} = \{1, \ldots, n_z\}$ (*e.g.* "`stuck rear wheel`", "`walking`", "`damaged camera`", "`spotting alliens`").

- **Goal**: Obtain a recursive estimate of $p(x_{0:t}, z_{0:t} | y_{1:t})$ from which we can derive $p(z_t | y_{1:t})$, where $x_{0:t} \triangleq \{x_0, x_1, \ldots, x_t\}$.

Rao-Blackwellised Particle Filtering

$$\begin{aligned} z_t &\sim P(z_t|z_{t-1}) \\ x_t &= A(z_t)x_{t-1} + B(z_t)w_t + F(z_t)u_t \\ y_t &= C(z_t)x_t + D(z_t)v_t + G(z_t)u_t \end{aligned}$$

- $u_t \in \mathcal{U}$ is a known control signal.
- i.i.d noise processes: $w_t \sim \mathcal{N}(0, I)$ and $v_t \sim \mathcal{N}(0, I)$.
- The parameters $(A, B, C, D, E, F, P(z_t|z_{t-1}))$ are known matrices; see [Andrieu, de Freitas, Doucet, 1999] for parameter estimation and model selection.
- Initial states: $x_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ and $z_0 \sim P(z_0)$.

Naïve solution with PF

- For $i = 1, ..., N$, sample from the transition priors

$$\widetilde{z}_t^{(i)} \sim \Pr(z_t | z_{t-1}^{(i)}) \qquad \widetilde{x}_t^{(i)} \sim p(x_t | x_{t-1}^{(i)}, \widetilde{z}_t^{(i)})$$

- For $i = 1, ..., N$, evaluate and normalize the weights

$$\widetilde{w}_t^{(i)} \propto p\left(y_t \left| \widetilde{x}_t^{(i)}, \widetilde{z}_t^{(i)} \right.\right)$$

- Select fittest particles.

RBPF: The conditioning argument

- Exploit the following factorisation

$$p\left(x_{0:t}, z_{0:t} \middle| y_{1:t}\right) = p\left(x_{0:t} \middle| y_{1:t}, z_{0:t}\right) p\left(z_{0:t} \middle| y_{1:t}\right)$$

- The density $p\left(x_{0:t} \middle| y_{1:t}, z_{0:t}\right)$ is Gaussian and can be computed analytically if we know the marginal posterior density $p\left(z_{0:t} \middle| y_{1:t}\right)$.

- This marginal density satisfies the alternative recursion

$$p\left(z_{0:t} | y_{1:t}\right) = p\left(z_{0:t-1} | y_{1:t-1}\right) \frac{p\left(y_t | y_{1:t-1}, z_{0:t}\right) p\left(z_t | z_{t-1}\right)}{p\left(y_t | y_{1:t-1}\right)}$$

RBPF: Do it Analytically if you Can!

Given $\{z_{0:t}^{(i)}, w_t^{(i)}\}_{i=1}^{N}$, we have

$$\widehat{P}_N(z_{0:t}|y_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta_{z_{0:t}^{(i)}}(z_{0:t})$$

and the marginal density of $x_{0:t}$ is a Gaussian mixture:

$$\begin{aligned}\widehat{p}_N(x_{0:t}|y_{1:t}) &= \sum_{\mathcal{Z}^{t+1}} p(x_{0:t}|z_{0:t}, y_{1:t})\widehat{P}_N(z_{0:t}|y_{1:t}) \\ &= \sum_{i=1}^{N} w_t^{(i)} p(x_{0:t}|y_{1:t}, z_{0:t}^{(i)})\end{aligned}$$

that can be computed efficiently with a bank of Kalman filters

RBPF: Do it Analytically if you Can!

We sample $z_t^{(i)}$ and then propagate the mean $\mu_t^{(i)}$ and covariance $\Sigma_t^{(i)}$ of x_t with a Kalman filter

$$
\begin{aligned}
\mu_{t|t-1}^{(i)} &= A(z_t^{(i)})\mu_{t-1|t-1}^{(i)} + F(z_t^{(i)})u_t \\
\Sigma_{t|t-1}^{(i)} &= A(z_t^{(i)})\Sigma_{t-1|t-1}^{(i)} A(z_t^{(i)})^{\mathrm{T}} + B(z_t^{(i)})B(z_t^{(i)})^{\mathrm{T}} \\
S_t^{(i)} &= C(z_t^{(i)})\Sigma_{t|t-1}^{(i)} C(z_t^{(i)})^{\mathrm{T}} + D(z_t^{(i)})D(z_t^{(i)})^{\mathrm{T}} \\
y_{t|t-1}^{(i)} &= C(z_t^{(i)})\mu_{t|t-1}^{(i)} + G(z_t^{(i)})u_t \\
\mu_{t|t}^{(i)} &= \mu_{t|t-1}^{(i)} + \Sigma_{t|t-1}^{(i)} C(z_t^{(i)})^{\mathrm{T}} S_t^{-1(i)} (y_t - y_{t|t-1}^{(i)}) \\
\Sigma_{t|t}^{(i)} &= \Sigma_{t|t-1}^{(i)} - \Sigma_{t|t-1}^{(i)} C(z_t^{(i)})^{\mathrm{T}} S_t^{-1(i)} C(z_t^{(i)}) \Sigma_{t|t-1}^{(i)},
\end{aligned}
$$

and with $\widehat{z}_t^{(i)} \sim \Pr(z_t | z_{t-1}^{(i)})$, we have

$$
w_t = p\left(y_t | y_{1:t-1}, z_{0:t}^{(i)}\right) = \mathcal{N}\left(y_{t|t-1}^{(i)}, S_t^{(i)}\right)
$$

RBPF Algorithm

- For $i = 1, \ldots, N$ sample $\widetilde{z}_t^{(i)} \sim \Pr(z_t | z_{t-1}^{(i)})$

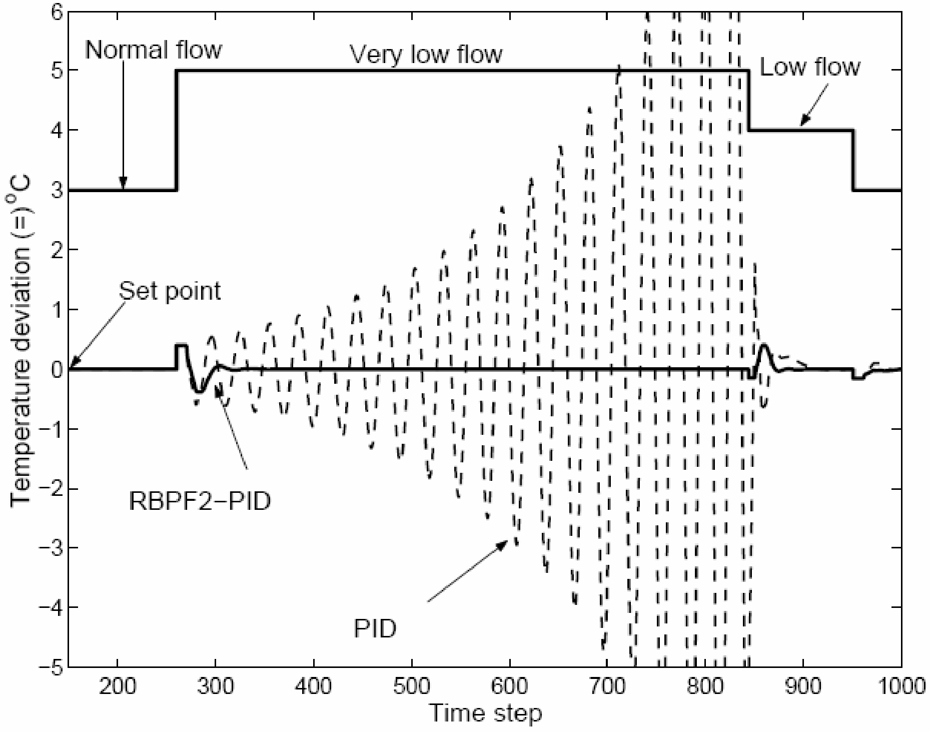
- For $i = 1, \ldots, N$, evaluate and normalize the weights

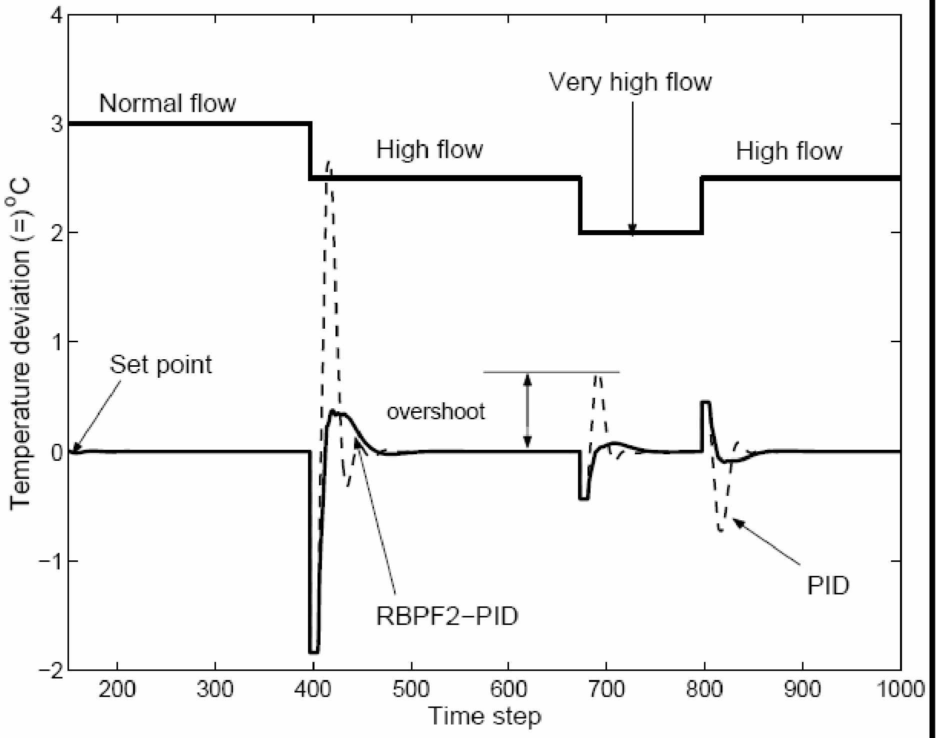
$$\widetilde{w}_t^{(i)} \propto p\left(y_t | y_{1:t-1}, \widetilde{z}_t^{(i)}\right)$$

- Select fittest particles.

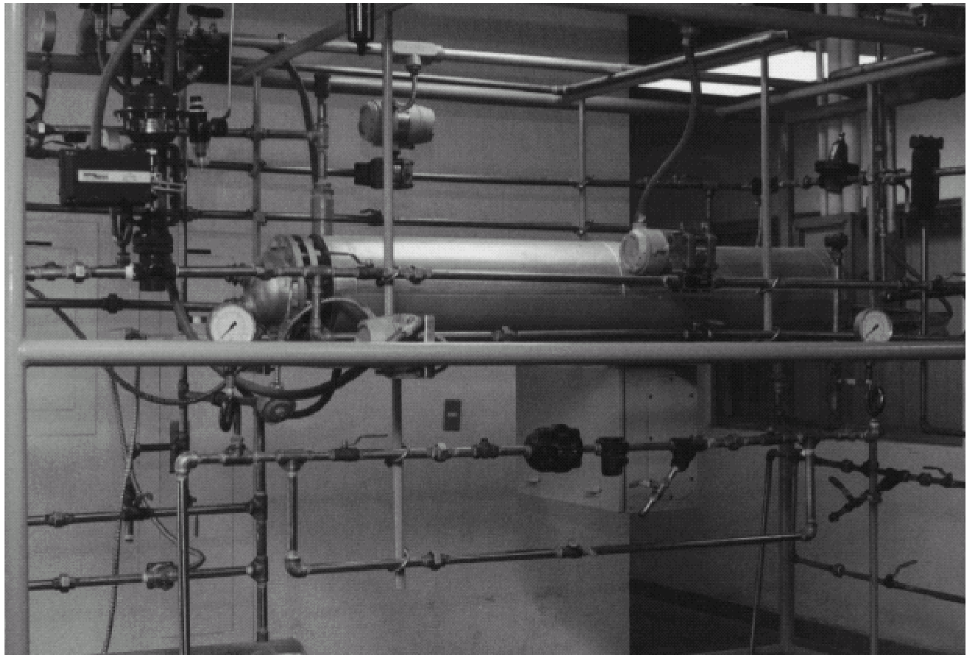
- For $i = 1, \ldots, N$, use one step of the Kalman recursion to compute the minimum statistics $\left\{\mu_{t+1|t}^{(i)}, \Sigma_{t+1|t}^{(i)}, y_{t+1|t}^{(i)}, S_{t+1}^{(i)}\right\}$ given $\left\{z_t^{(i)}, \mu_{t|t-1}^{(i)}, \Sigma_{t|t-1}^{(i)}\right\}$.

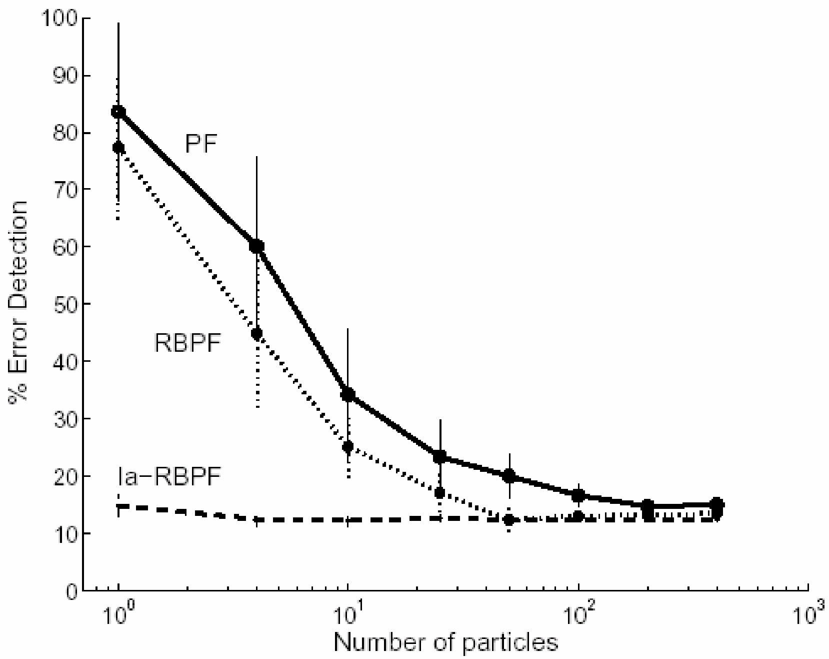
RBPF for Hybrid Control with PIDs

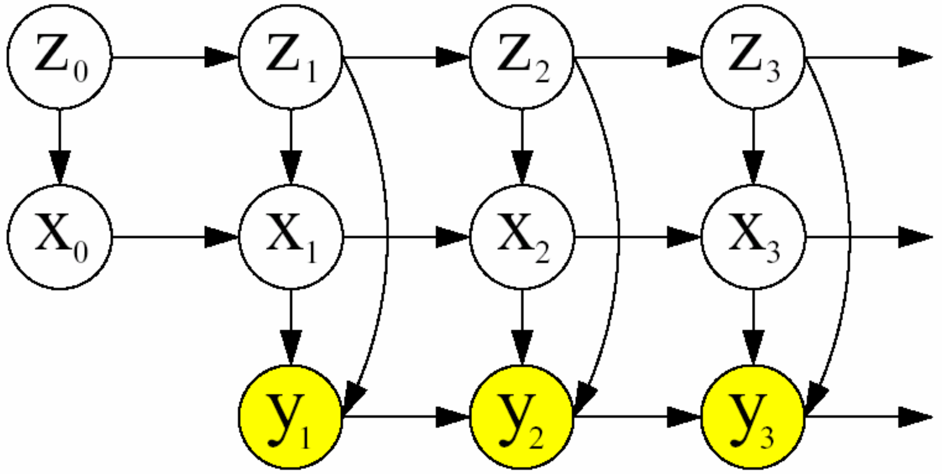




RBPF Real-time diagnosis

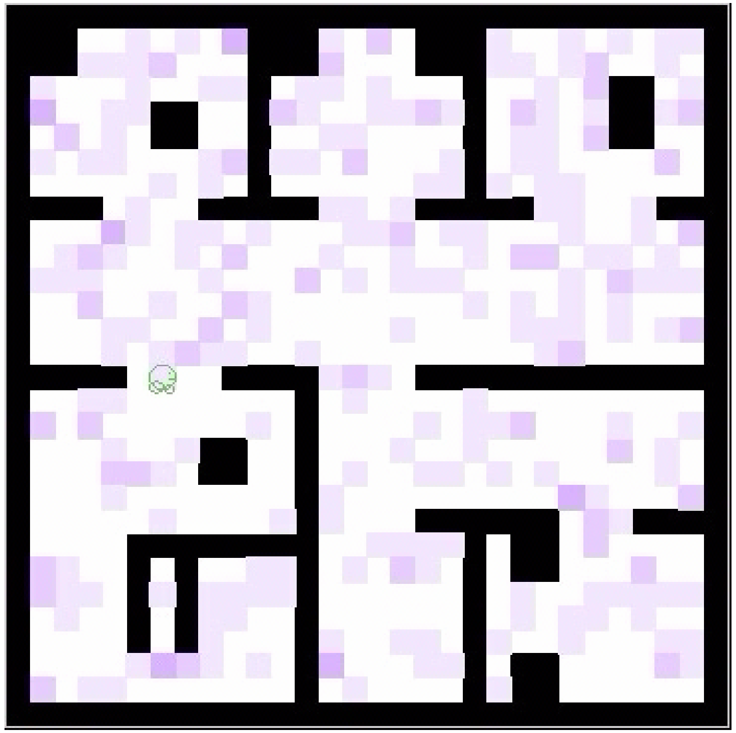


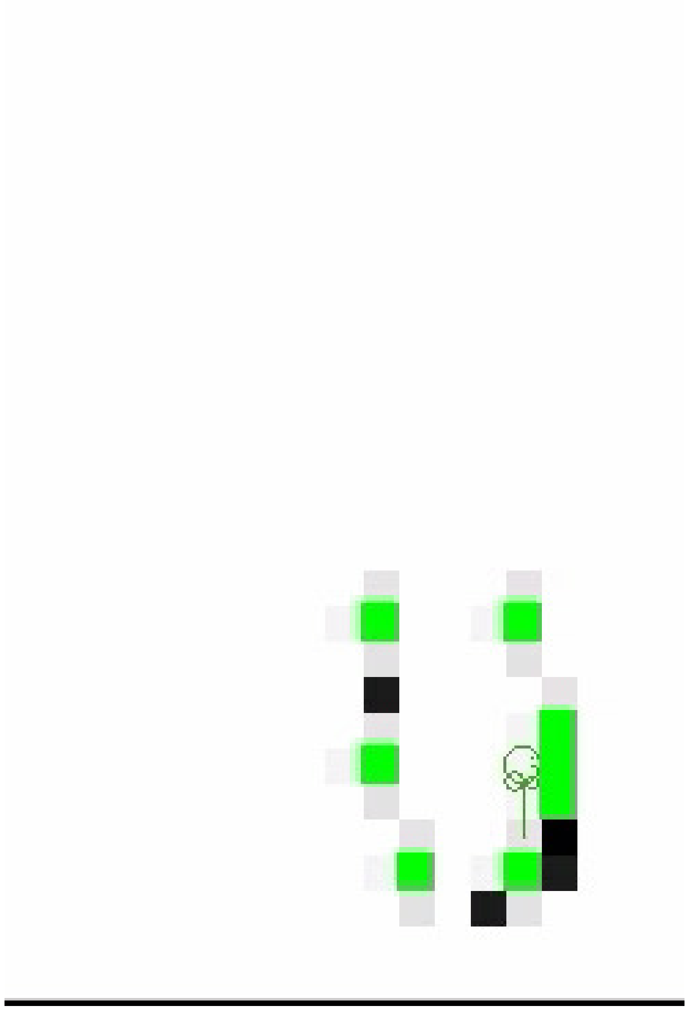




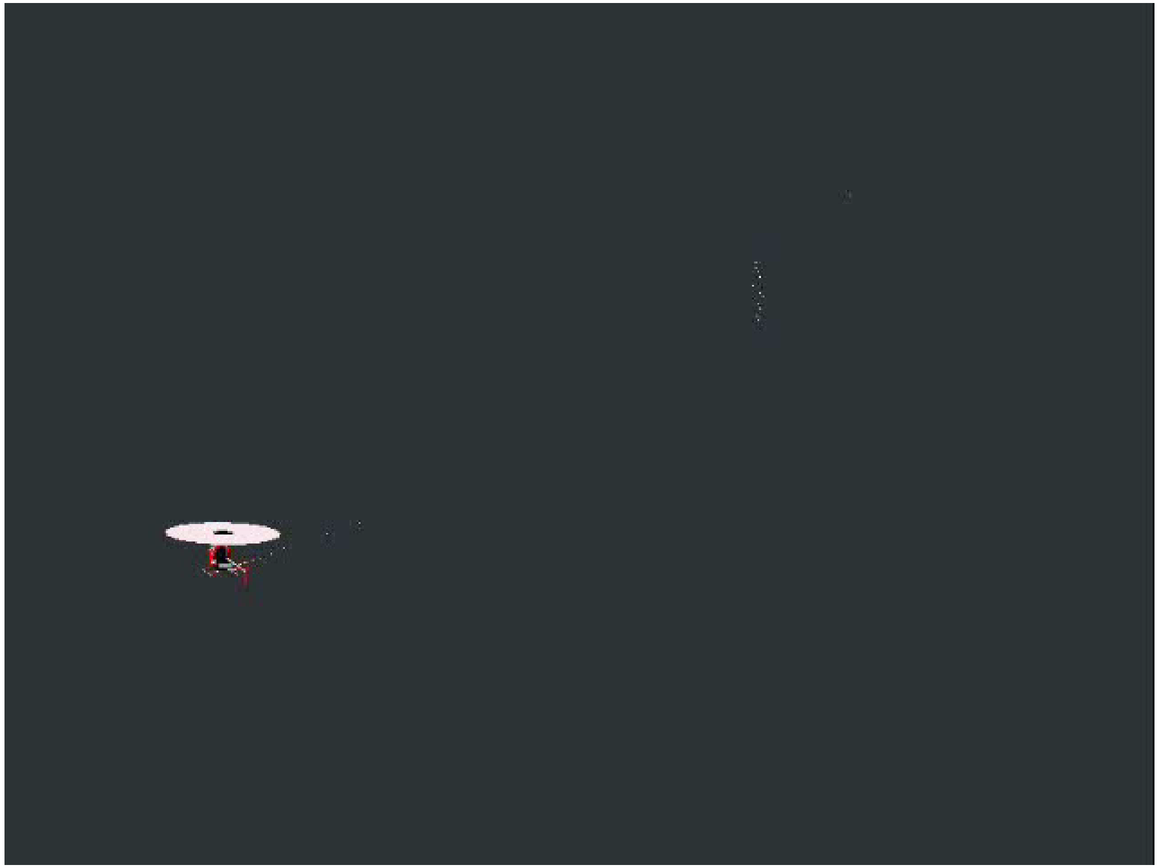
Given samples of z, we can solve for x exactly with a mixture of Kalman filters.

RBPF for SLAM





RBPF for SLAM



See work of Sebastian Thrun, Frank Dellaert and Dieter Fox

Beyond Filtering

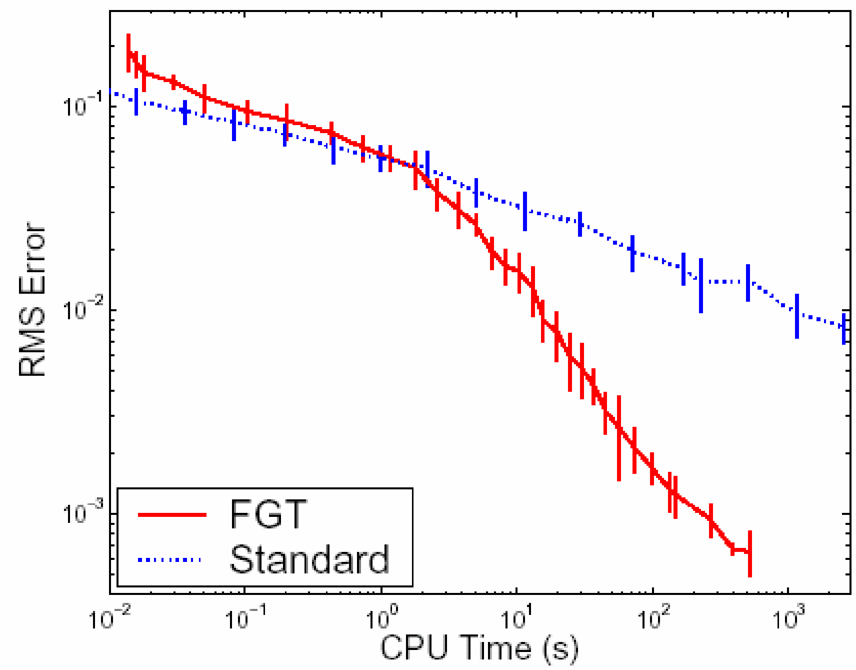
Filtering: $p\left(x_t \middle| y_{1:t}\right)$

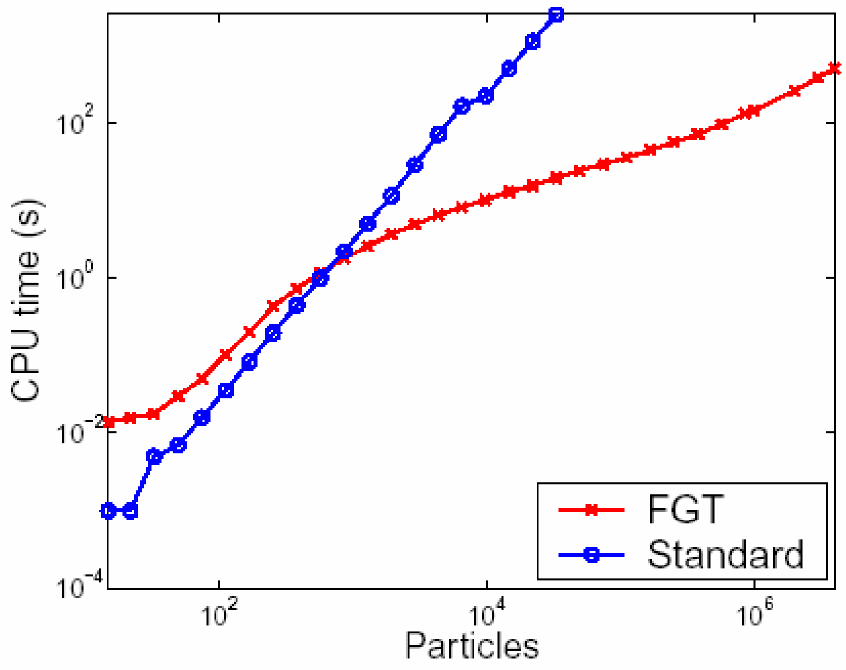
Smoothing: $p\left(x_t \middle| y_{1:T}\right)$

Viterbi: $\arg\max_{x_{1:T}} p\left(x_{1:T} \middle| y_{1:T}\right)$

Filtering is O(N), but smoothing and Viterbi are O(N ^{2})

Solution: **Fast multipole methods** (Greengard and Rohklin), **dual metric trees** (Gray and Moore) and **Huttenlocher's tricks** – no FFT.





Bayesian smoothing

When filtering we compute

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

When smoothing we are interested in

$$p(x_t|y_{1:T})$$

Smoothing is more informative:
If we have all the observations, why not use them?

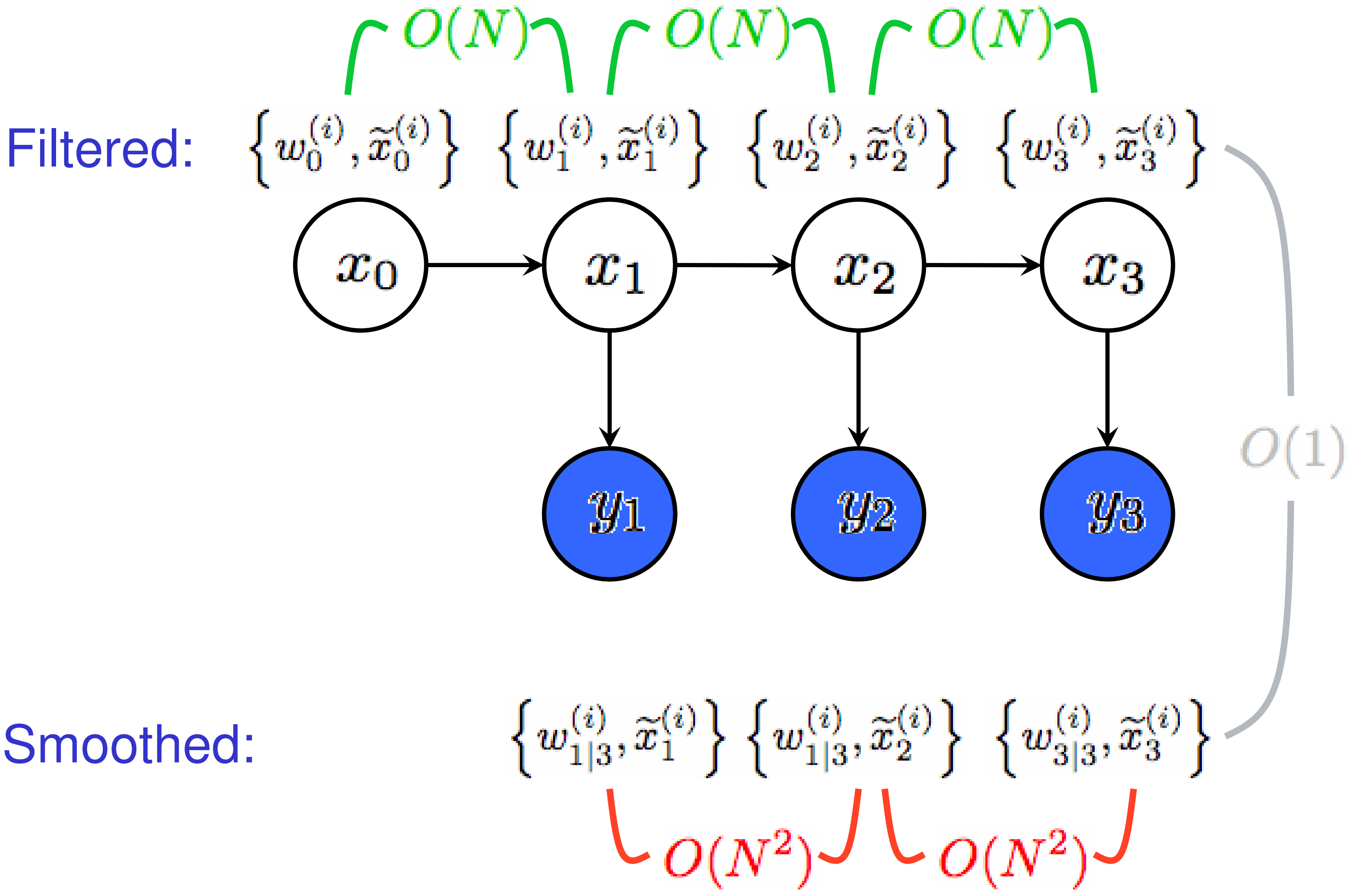
Particle Smoothing Methods

Common methods:

- Forward-Backward smoother (FBS)
  - (Kitagawa, 1996)
- Two-Filter smoother (TFS)
  - (Kitagawa, 1996; Isard et al., 1998)
  - see paper for new generalized derivation
- Maximum *a posteriori* (MAP) smoother
  - (Godsill et al., 2001)

For brevity, I'll present details of only Forward-Backward smoothing.

Forward-Backward Particle Smoothing


$O(N)$
$O(N)$
$O(N)$
Filtered:
$\left\{w_0^{(i)}, \widetilde{x}_0^{(i)}\right\}$
$\left\{w_1^{(i)}, \widetilde{x}_1^{(i)}\right\}$
$\left\{w_2^{(i)}, \widetilde{x}_2^{(i)}\right\}$
$\left\{w_3^{(i)}, \widetilde{x}_3^{(i)}\right\}$
x_0
x_1
x_2
x_3
y_1
y_2
y_3
$O(1)$
Smoothed:
$\left\{w_{1|3}^{(i)}, \widetilde{x}_1^{(i)}\right\}$
$\left\{w_{1|3}^{(i)}, \widetilde{x}_2^{(i)}\right\}$
$\left\{w_{3|3}^{(i)}, \widetilde{x}_3^{(i)}\right\}$
$O(N^2)$
$O(N^2)$

Particles are re-weighted only: support doesn't change!

Forward-Backward Particle Smoothing

$$p(x_t|y_{1:T}) = \underbrace{p(x_t|y_{1:t})}_{\text{filtered}} \int \frac{\overbrace{p(x_{t+1}|y_{1:T})}^{\text{smoothed}}\overbrace{p(x_{t+1}|x_t)}^{\text{dynamics}}}{\underbrace{\int p(x_{t+1}|x_t)p(x_t|y_{1:t})dx_t}_{\text{state prediction}}} dx_{t+1}.$$

Hence, we can derive a particle smoother which performs forward filtering, then backward smoothing:

$$w_{t|T}^{(i)} = w_t^{(i)} \left[\sum_{j=1}^{N} w_{t+1|T}^{(j)} \frac{p(\widetilde{x}_{t+1}^{(j)}|\widetilde{x}_t^{(i)})}{\sum_{k=1}^{N} w_t^{(k)} p(\widetilde{x}_{t+1}^{(j)}|\widetilde{x}_t^{(k)})} \right]$$

The $O(N^2)$ bottleneck

Bayesian integrals $$p(x_t|y_{1:t}) \int \frac{p(x_{t+1}|y_{1:T})p(x_{t+1}|x_t)}{\int p(x_{t+1}|x_t)p(x_t|y_{1:t})dx_t} dx_{t+1}$$

↓

Monte Carlo estimator $$w_t^{(i)} \left[\sum_{j=1}^{N} w_{t+1|T}^{(j)} \frac{p(\widetilde{x}_{t+1}^{(j)}|\widetilde{x}_t^{(i)})}{\sum_{k=1}^{N} w_t^{(k)} p(\widetilde{x}_{t+1}^{(j)}|\widetilde{x}_t^{(k)})} \right]$$

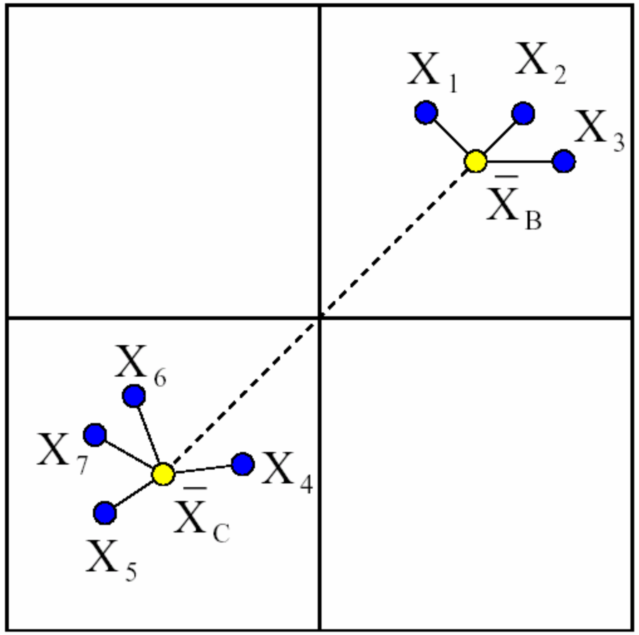
↓

Sum-kernel computation $$\forall_{j=1..N}, f_j = \sum_{i=1}^{N} w_i K(x_i, y_j)$$

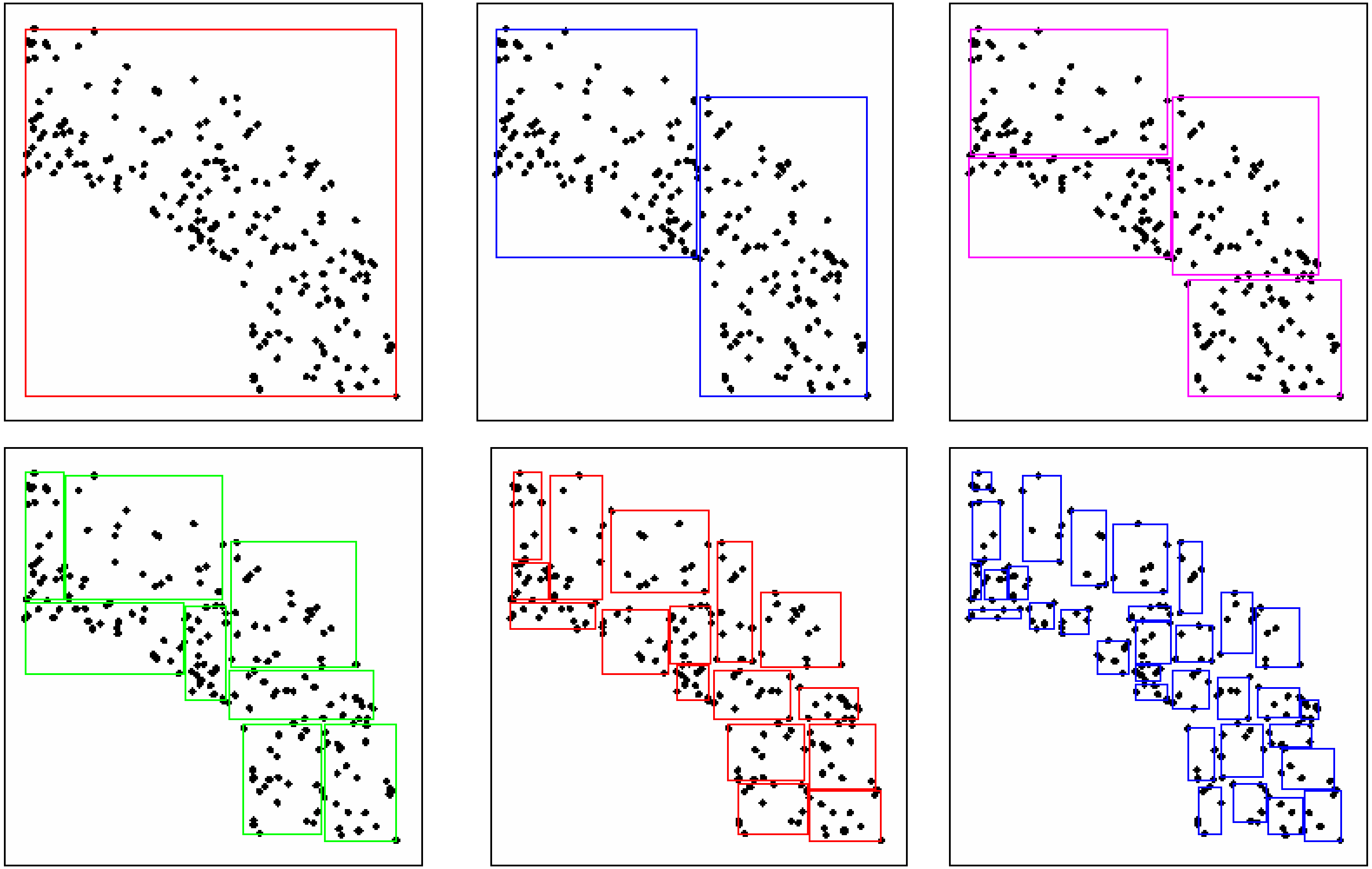
O(N^2)

Fast Multipole Methods use partitions and expansions of the kernel

$$
\begin{aligned}
f_j &= \sum_{i=1}^{N} w_i K(\mathbf{x}_i, \mathbf{y}_j) \\
&= \sum_{i=1}^{N} w_i K(\|\mathbf{x}_i - \mathbf{y}_j\|) \\
&= \sum_{i=1}^{N} w_i K(\|\mathbf{x}_i - c - (\mathbf{y}_j - c)\|) \\
&= \sum_{i=1}^{N} w_i \sum_{m=1}^{\infty} \Phi_m(\mathbf{x}_i - c)\Psi_m(\mathbf{y}_j - c) \\
&= \sum_{i=1}^{N} w_i \sum_{m=1}^{p} \Phi_m(\mathbf{x}_i - c)\Psi_m(\mathbf{y}_j - c) + e(p, \mathbf{x}, \mathbf{y}) \\
&= \sum_{m=1}^{p} \Psi_m(\mathbf{y}_j - c) \sum_{i=1}^{N} w_i \Phi_m(\mathbf{x}_i - c) + e(p, \mathbf{x}, \mathbf{y}) \\
&= \sum_{m=1}^{p} c_m \Psi_m(\mathbf{y}_j - c) + e(p, \mathbf{x}, \mathbf{y})
\end{aligned}
$$

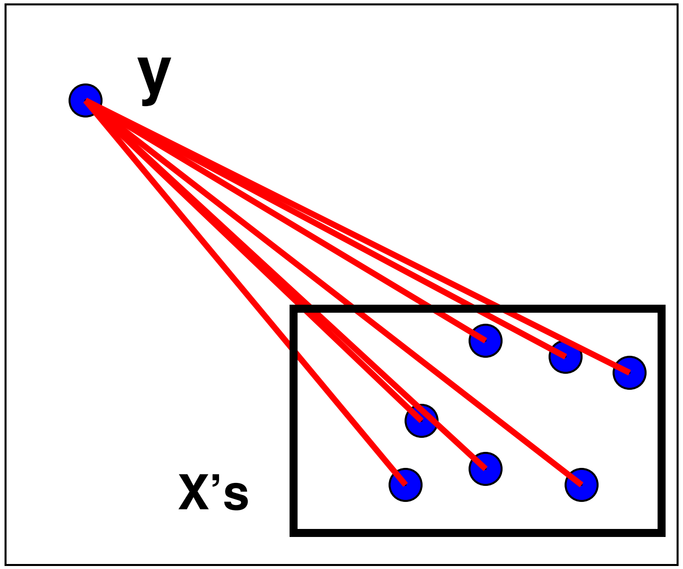


Tree recursions: We start by partitioning points using kd-trees or any metric trees



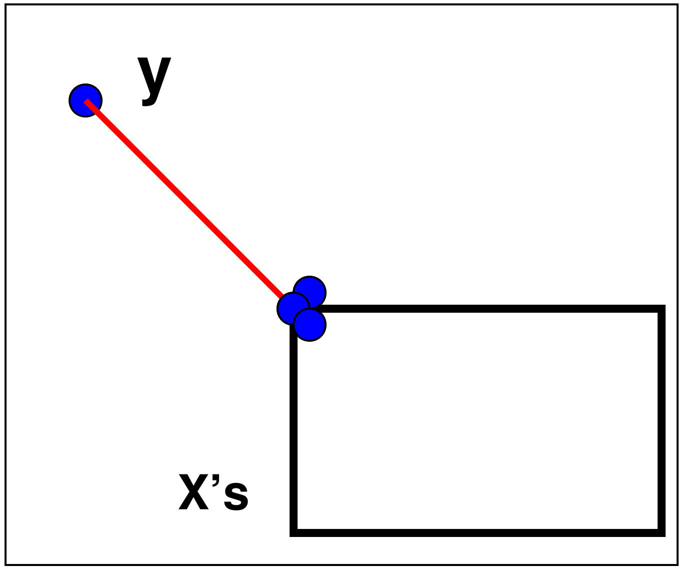
(Gray and Moore, 2000)

Far away groups of points are replaced by two single points (upper and lower bound)



$$f_j = \sum_{i=1}^{N} w_i \; K_i \left(\|x_i - y_j\| \right)$$

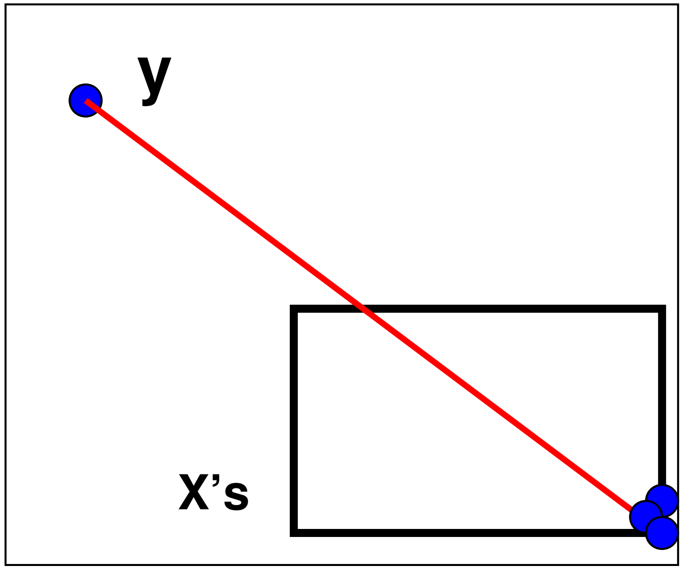
Far away groups of points are replaced by two single points (upper and lower bound)



$$f_j = \sum_{i=1}^{N} w_i \; K_i\left(\|x_i - y_j\|\right)$$

$$f^{(upper)}(X,Y) \;\; = \;\; K\left(d^{(lower)}(X,Y)\right) \sum_{i \in X} w_i$$

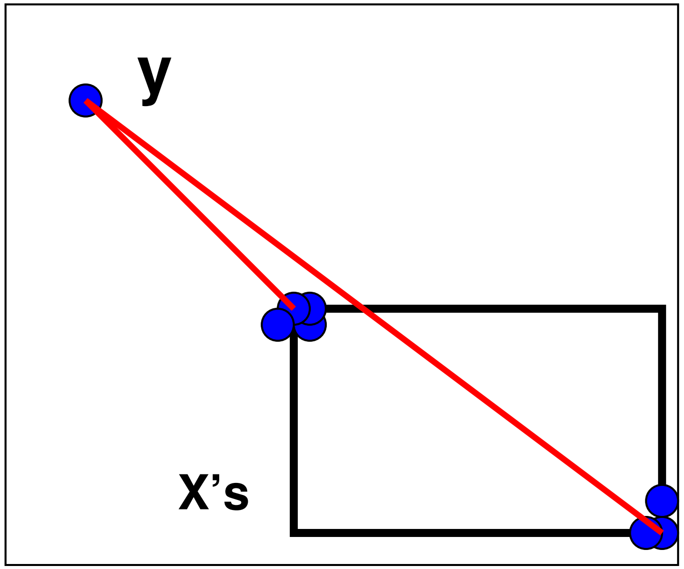
Far away groups of points are replaced by two single points (upper and lower bound)



$$f_j = \sum_{i=1}^{N} w_i \; K_i \left(\|x_i - y_j\| \right)$$

$$f^{(lower)}(X,Y) \;\; = \;\; K\left(d^{(upper)}(X,Y) \right) \sum_{i \in X} w_i$$

Far away groups of points are replaced by two single points (upper and lower bound)



$$f_j = \sum_{i=1}^{N} w_i \ K_i \left(\|x_i - y_j\| \right)$$

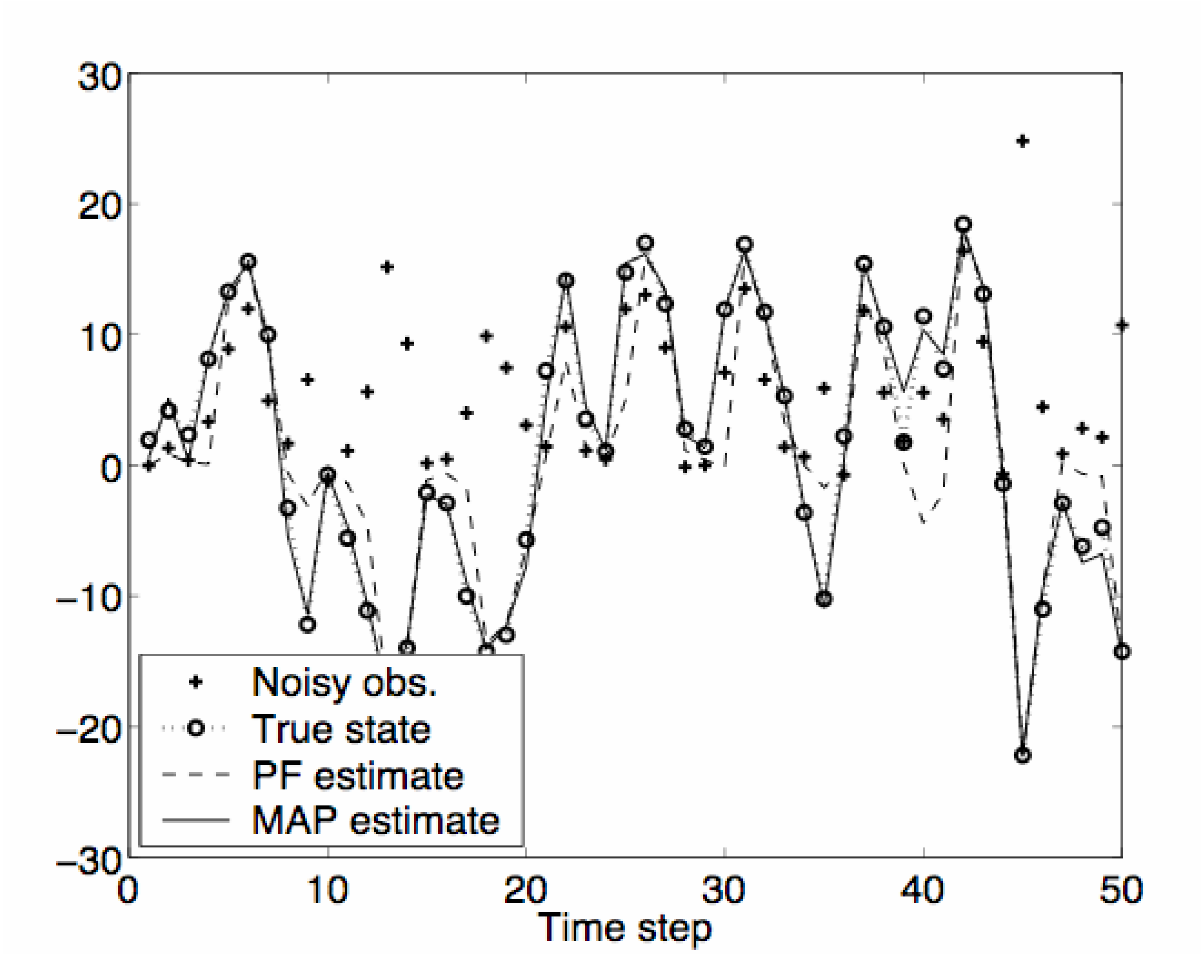
$$\tilde{f}(X,Y) \ = \ \frac{1}{2} \ \left(f^{(upper)} \left(X, Y \right) + f^{(lower)} \left(X, Y \right) \right)$$

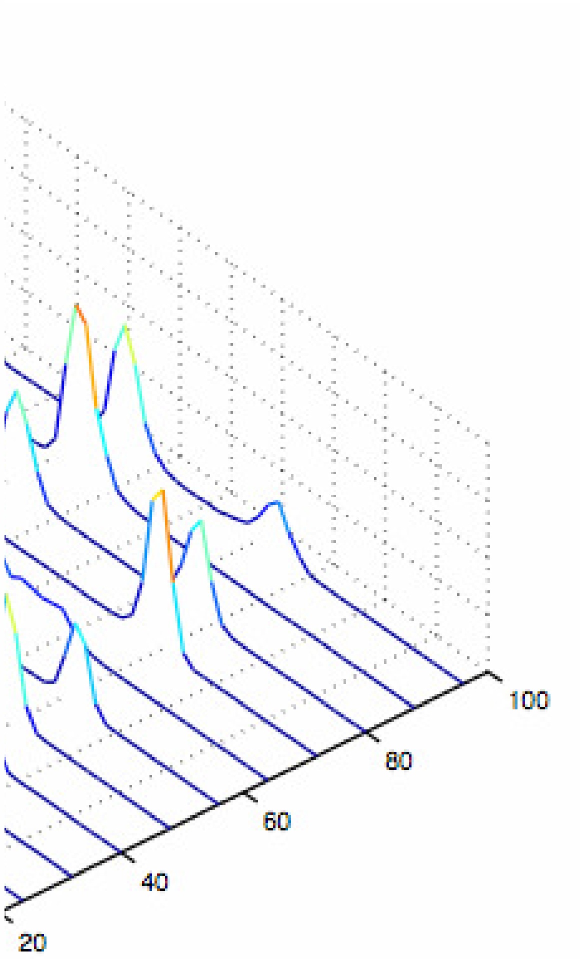
$$e(X,Y) \ = \ \frac{1}{2} \ \left(f^{(upper)} \left(X, Y \right) - f^{(lower)} \left(X, Y \right) \right)$$

Multi-modal non-Gaussian model - MAP smoother

$$y_t = x_t^2/20 + \mathcal{N}(0, \sigma_y)$$

$$x_t = x_{t-1}/2 + 25x_{t-1}/\left(1 + x_{t-1}^2)\right) + \cos(1.2t) + \mathcal{N}(0, \sigma_x)$$

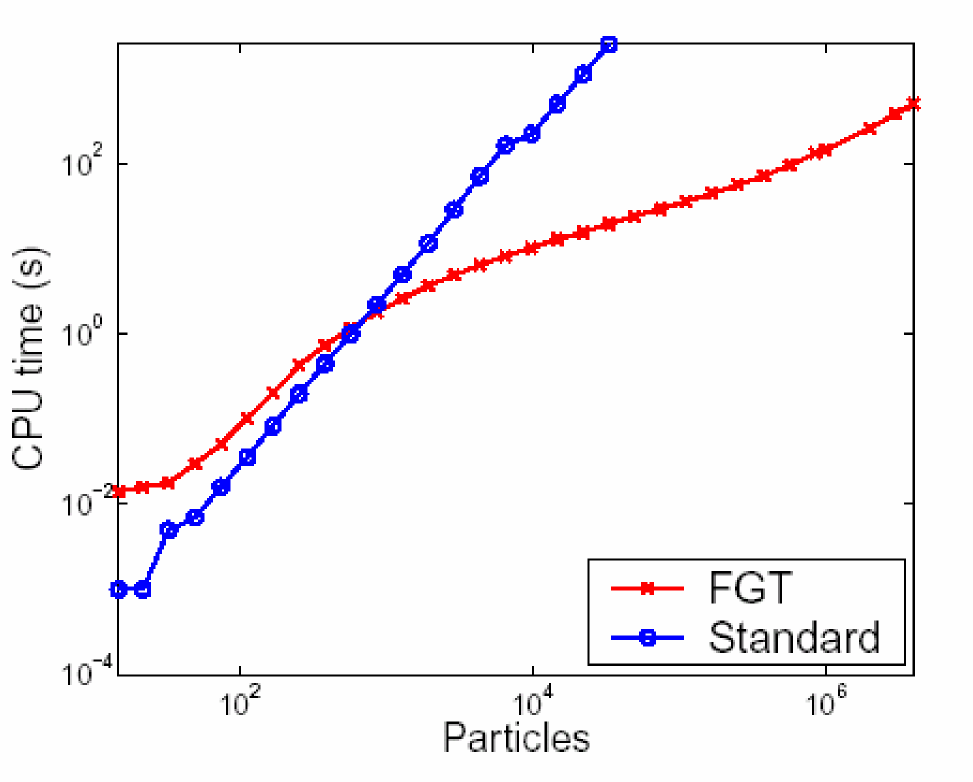




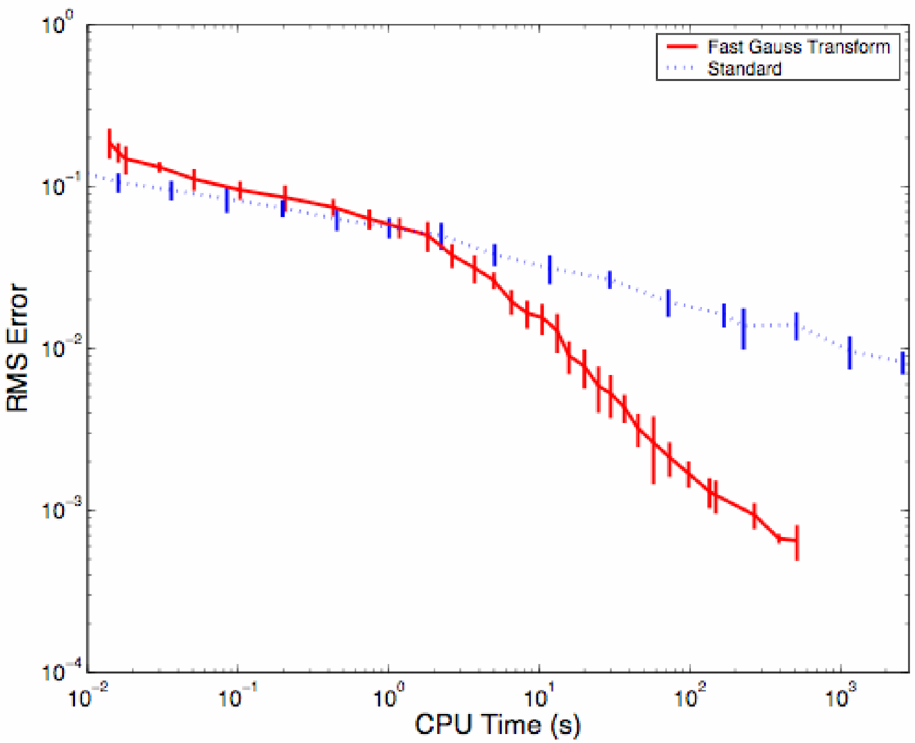
(Kitagawa et al., 1996)

…solved with MAP particle smoothing.

Particle smoothing with 4 million particles!

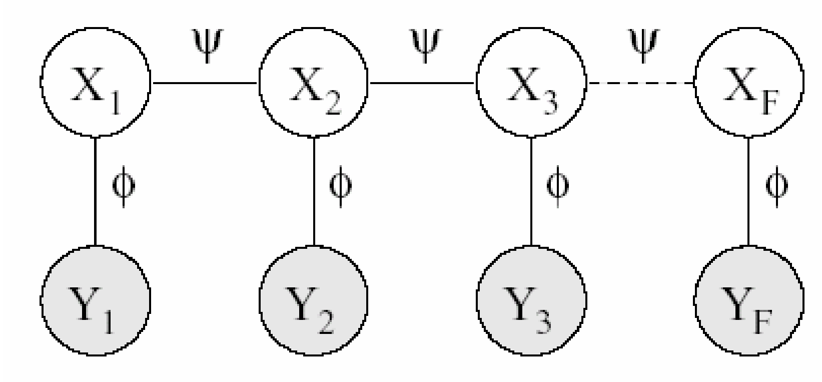


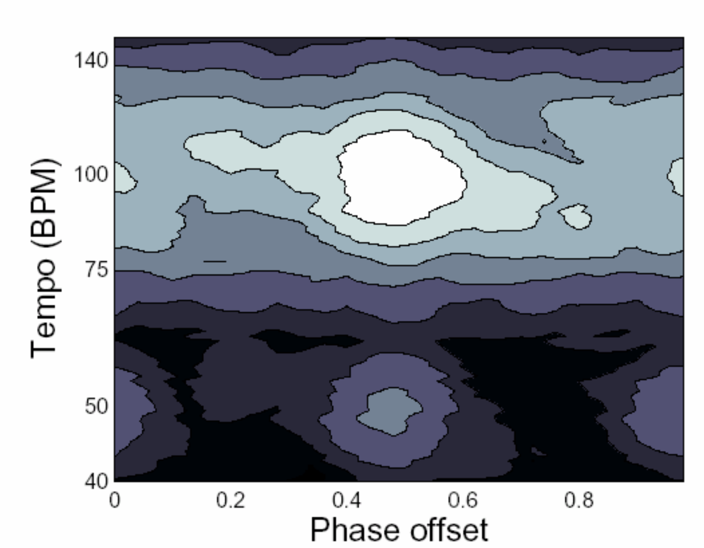
FBS on Linear-Gaussian chain of length 10 (3 dimensions)



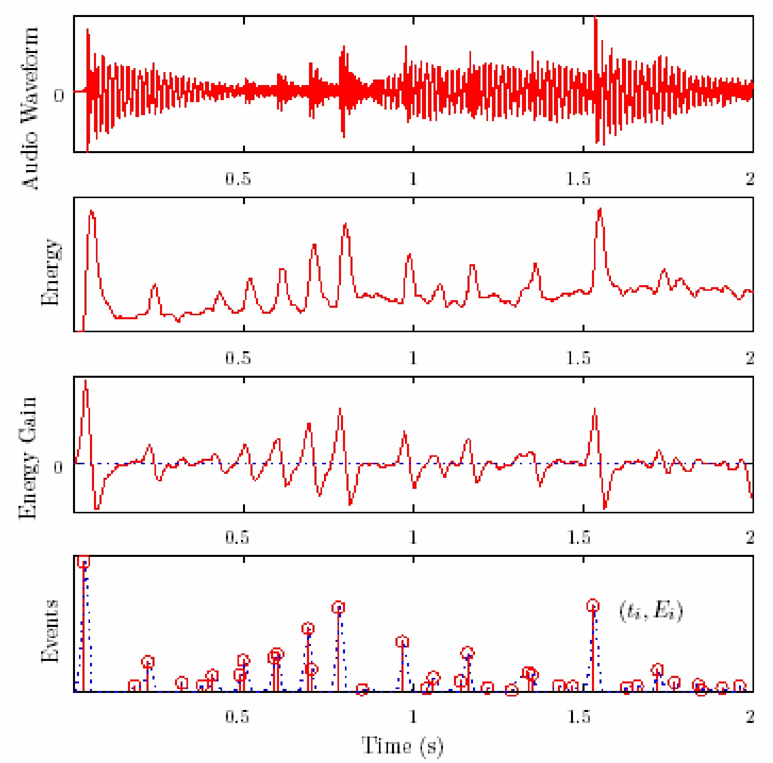
Better to use more particles, smoothed approximately, than fewer, smoothed exactly.

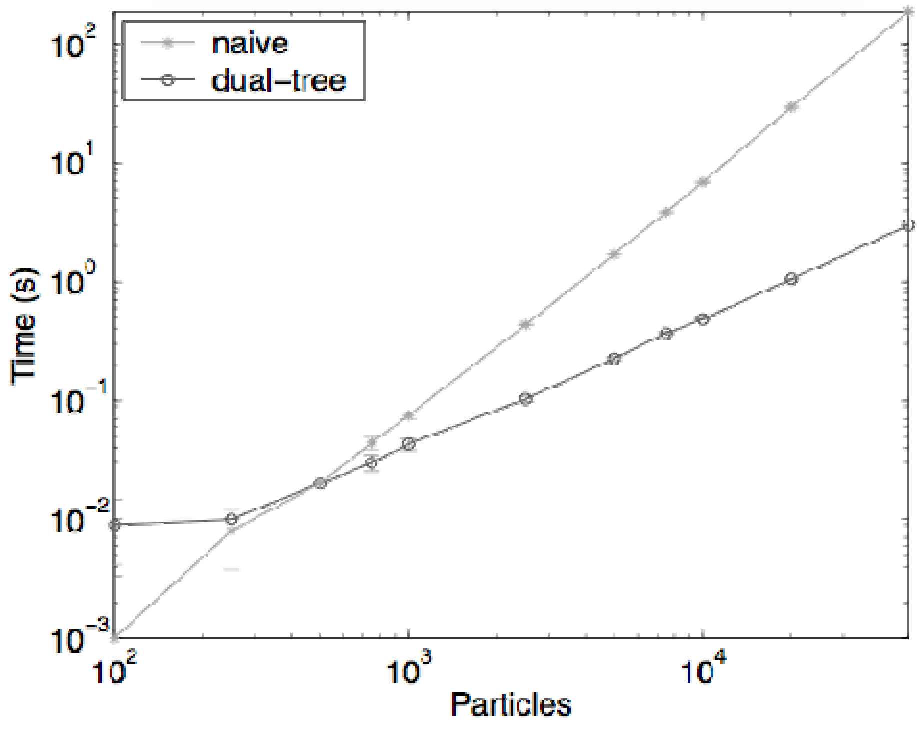
Application to Beat Tracking for Music Retrieval





2





Parameter estimation in dynamic state space models is still an open problem

$$\nabla_\theta p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) = \frac{\nabla_\theta p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})}{p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})} p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$$

$$\nabla_\theta p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t}) = \int_{\mathcal{X}^{t-1}} \frac{\nabla_\theta p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})}{p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})} p_\theta(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{1:t-1}$$

We’re doing importance sampling in a vast growing space.
This is very dangerous !

Parameter estimation is also **O(N^2)**

Particle methods and dual trees are a reasonable match for computing filter derivatives and parameter learning.

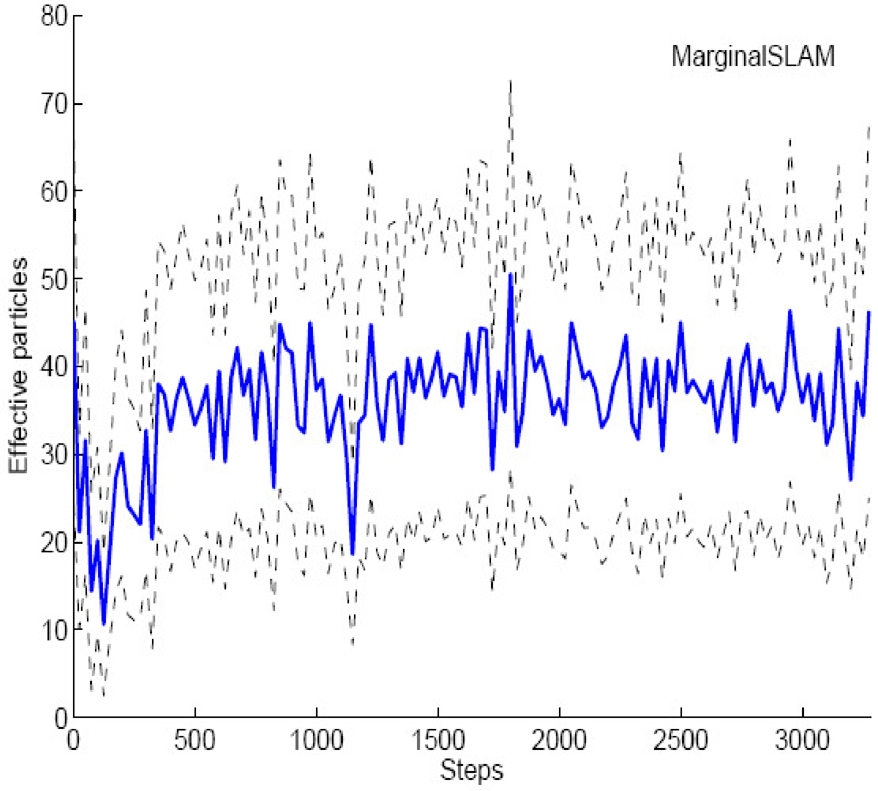
$$\widehat{p_\theta}(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \sum_{i=1}^{N} w_{t-1}^{(i)} \delta_{\mathbf{x}_{t-1}^{(i)}}(\mathbf{x}_{t-1})$$

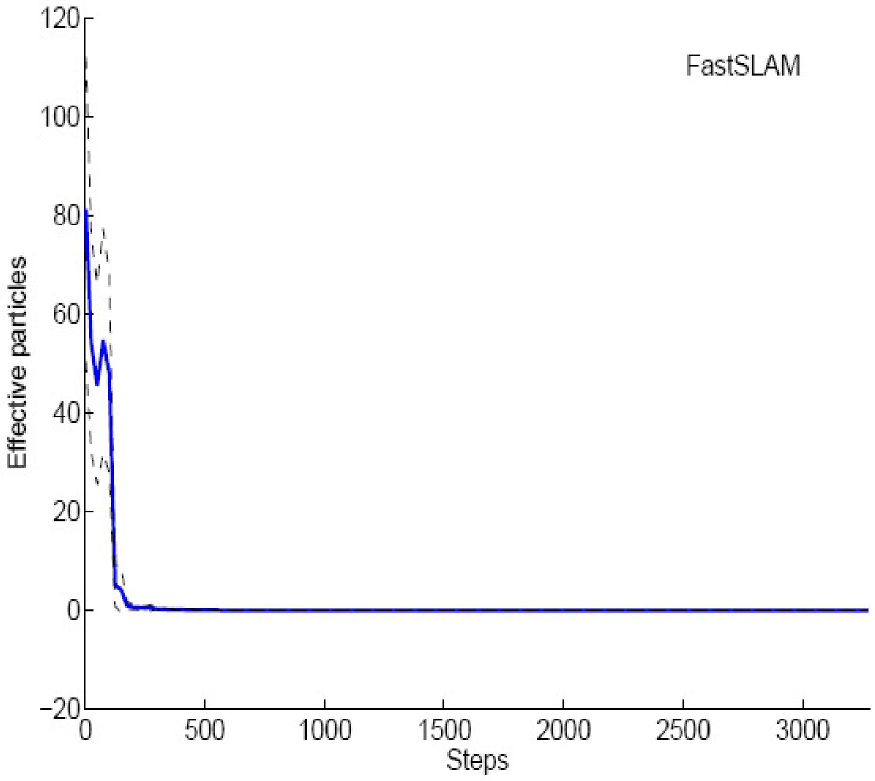
$$\widehat{\nabla_\theta p_\theta}(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \sum_{i=1}^{N} w_{t-1}^{(i)} \beta_{t-1}^{(i)} \delta_{\mathbf{x}_{t-1}^{(i)}}(\mathbf{x}_{t-1})$$

$$\widetilde{w}_t^{(i)} = \frac{p_\theta(\mathbf{y}_t|\mathbf{x}_t^{(i)}) \sum_{j=1}^{N} w_{t-1}^{(j)} p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(j)})}{q_\theta(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t})}$$

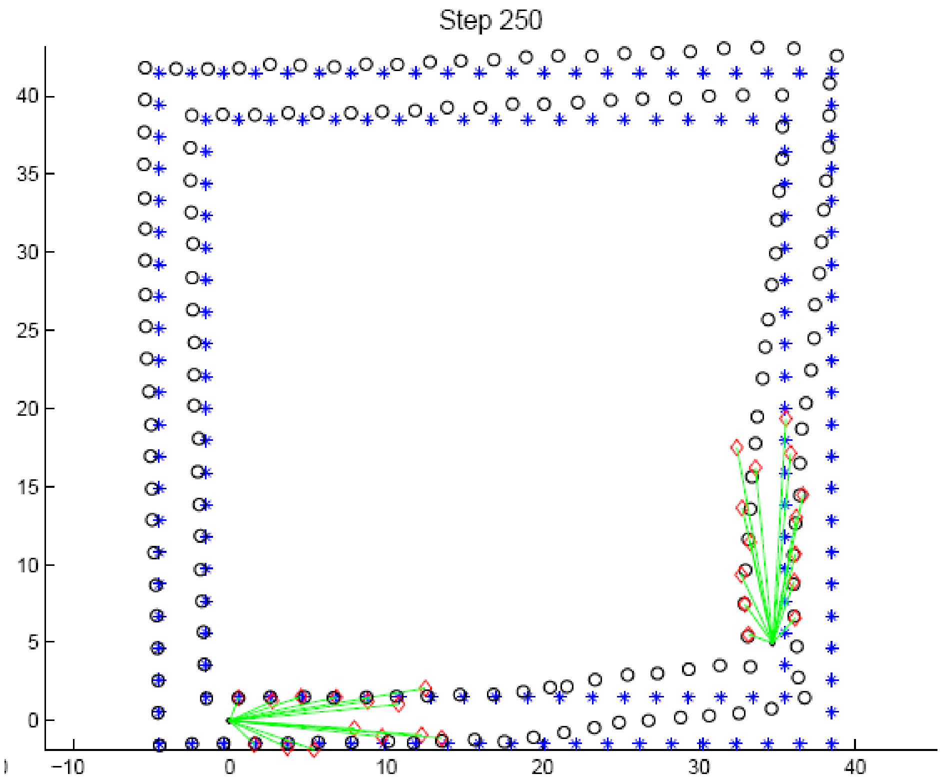
$$\widetilde{\rho}_t^{(i)} = \frac{p_\theta(\mathbf{y}_t|\mathbf{x}_t^{(i)}) \sum_j w_{t-1}^{(j)} p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(j)}) [\nabla_\theta \log p_\theta(\mathbf{y}_t|\mathbf{x}_t^{(i)}) + \beta_{t-1}^{(j)}]}{q_\theta(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t})}$$

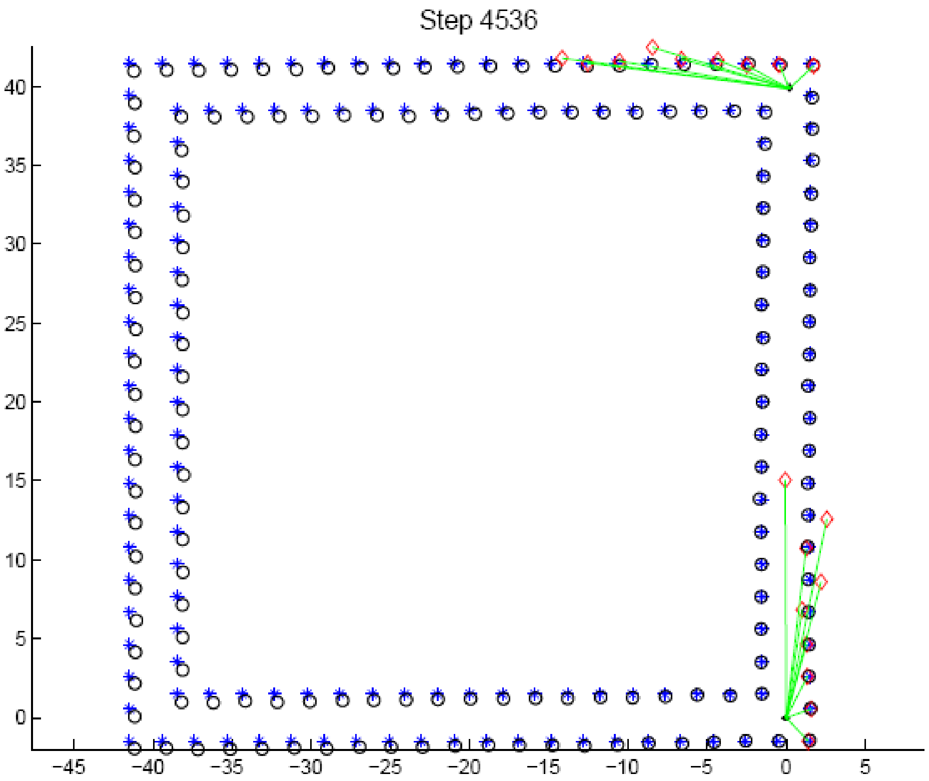
Static SLAM is an **O(N^2)** parameter estimation problem





Static SLAM is an O(N^2) parameter estimation problem





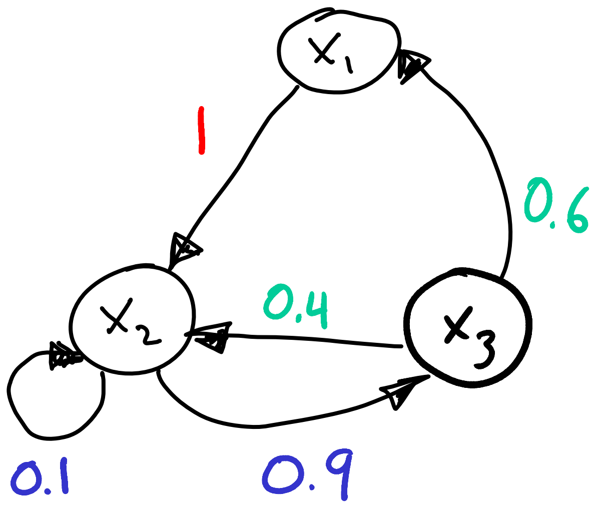
Overview

- History and applications
- Rejection sampling
- Importance sampling
- Particle filters
- Markov chain Monte Carlo
  - Metropolis-Hastings
  - Gibbs sampling
- Advanced Monte Carlo methods
  - Mixtures of kernels
  - Trans-dimensional Monte Carlo
  - Hybrid Monte Carlo
  - Monte Carlo EM
  - Particle methods with artificial dynamics

Markov Chain Monte Carlo

For simplicity, let's consider only 3 states:

$$x_t \in \mathcal{X} = \{x_1, x_2, x_3\}$$



$$T = P(x_t \mid x_{t-1}) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.1 & 0.9 \\ 0.6 & 0.4 & 0 \end{bmatrix}$$

Think of this as a webgraph. Our goal is to crawl it to find the "relevance" of each node.

Markov Chain Monte Carlo

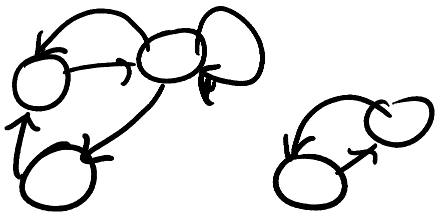
T is a stochastic matrix. As long as the graph (state space) is aperiodic and irreducible, we have that for any initial vector of probabilities ν:

$$\nu' T^t \to \pi' \quad \text{as} \quad t \to \infty$$

where π is the invariant or Stationary distribution of the Chain. It is unique.

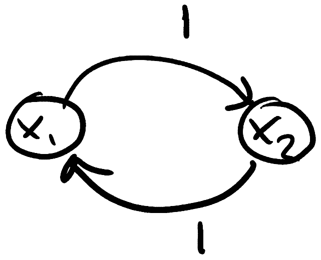
Markov Chain Monte Carlo

Need for irreducibility:



One cluster might never be visited!

Need for aperiodicity:



$$T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Let $\pi = \begin{bmatrix} 1/3 & 2/3 \end{bmatrix}$

$$\pi' T = \begin{bmatrix} 2/3 & 1/3 \end{bmatrix}$$

$$\pi' T^2 = \begin{bmatrix} 1/3 & 2/3 \end{bmatrix}$$

$$\vdots$$

Oscillation!

Markov Chain Monte Carlo

In the limit:

$$\pi' T = \pi'$$

π is the left eigenvector of T with corresponding eigenvalue 1. Componentwise, we have:

$$\sum_{i=1}^{3} \pi_i T_{ij} = \pi_j$$

As the state space grows:

$$\int \pi(x) \underbrace{p(y|x)}_{\text{Markov chain Kernel}} dx = \pi(y)$$

Markov Chain Monte Carlo

Detailed Balance:

If $$\pi(x_t)\,P(x_{t+1}|x_t) = \pi(x_{t+1})\,P(x_t|x_{t+1})$$

Integrating over x_t yields

$$\int \pi(x_t)\,P(x_{t+1}|x_t) = \pi(x_{t+1})$$

Which is the ergodic behaviour we want.
Now we have a sufficient condition for designing $P(x_{t+1}|x_t)$ so as to get samples from π

MCMC: Metropolis-Hastings

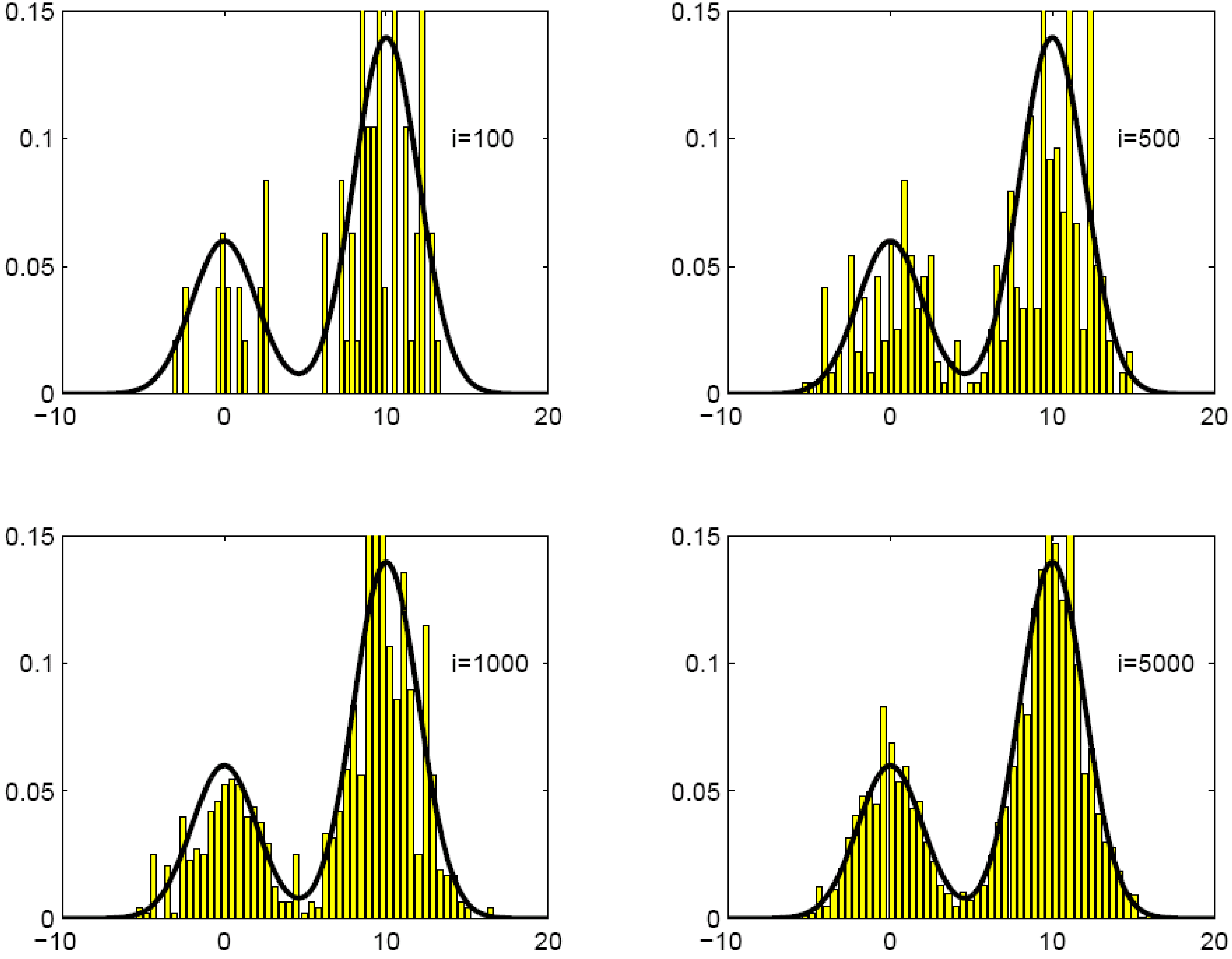
- Initialise $x^{(0)}$.
- For $i = 0$ to $N - 1$
  - Sample $u \sim U_{[0,1]}$.
  - Sample $x^{\star} \sim q(x^{\star}|x^{(i)})$.
  - If $u < A(x^{(i)}, x^{\star}) = \min\left\{1, \frac{p(x^{\star})q(x^{(i)}|x^{\star})}{p(x^{(i)})q(x^{\star}|x^{(i)})}\right\}$

    $$x^{(i+1)} = x^{\star}$$

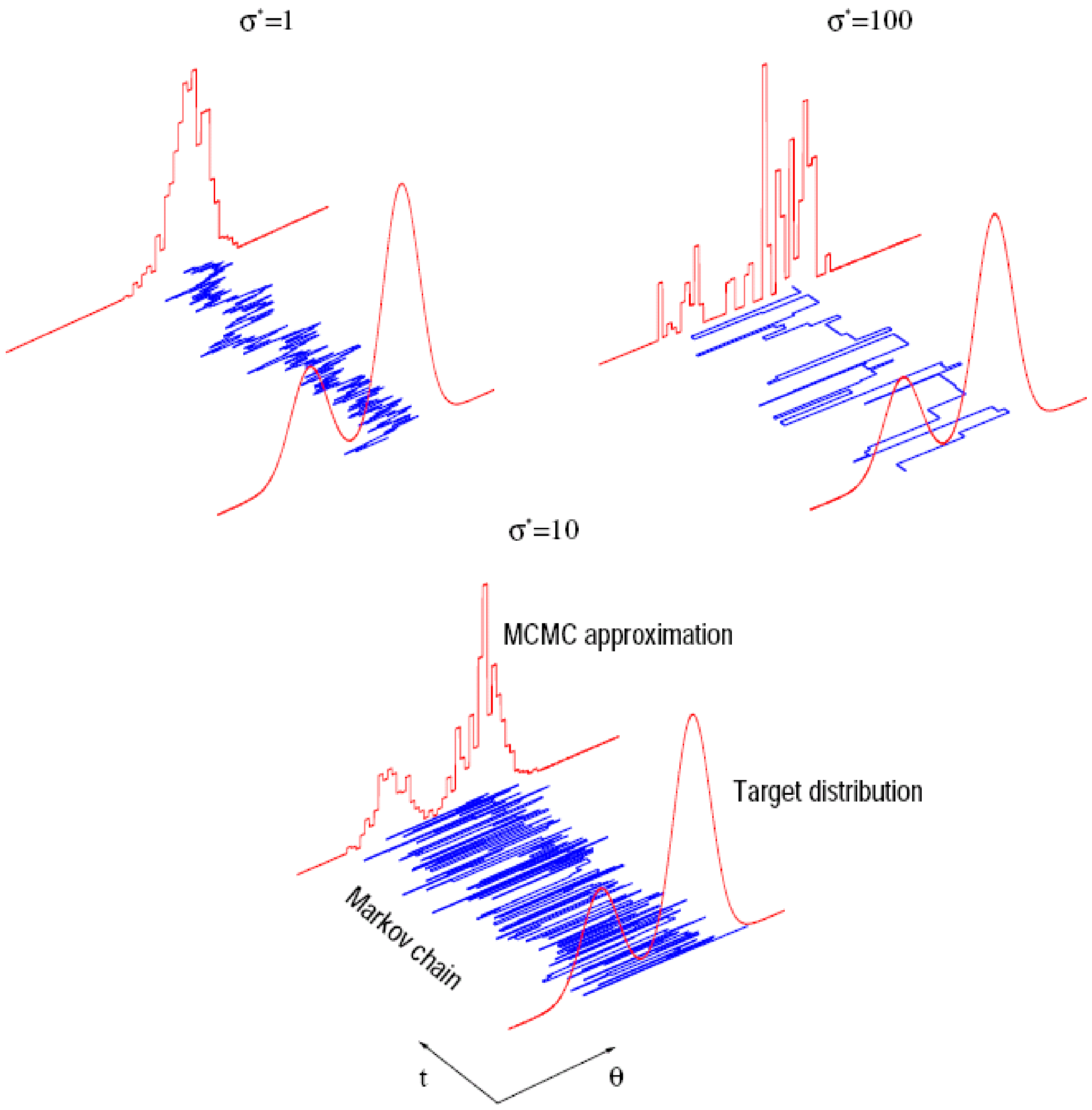
    else

    $$x^{(i+1)} = x^{(i)}$$

MCMC: Metropolis-Hastings



MCMC: Choosing the Right Proposal



MCMC: Theory

Kernel:

$$K(x,B) = \begin{cases} q(B|x)\,A(x,B) & x \notin B \\ 1 - \displaystyle\int_{x' \in \{\mathcal{X} \setminus B\}} q(x'|x)\,A(x,x') & x \in B \end{cases}$$

$$\therefore\; K(x,B) = q(B|x)\,A(x,B) + \mathbb{I}_{x \in B} \left\{ 1 - q(B|x)\,A(x,B) - \int_{x' \in \{\mathcal{X} \setminus B\}} q(x'|x)\,A(x,x') \right\}$$

$$K(x,B) = q(B|x)\,A(x,B) + \mathbb{I}_{x \in B} \left\{ 1 - \int_{x' \in \mathcal{X}} q(x'|x)\,A(x,x') \right\}$$

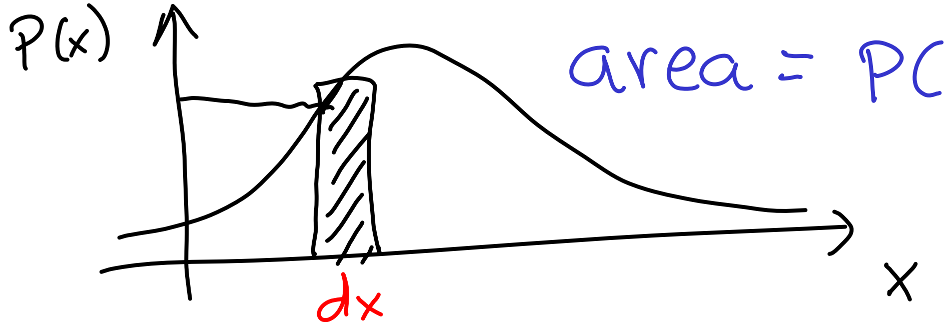
MCMC: Theory

Detailed balance:

$$\pi(A)K(A,B) = \pi(B)K(B,A)$$

$$\int_{x\in A} \pi(dx)\,K(x,B) = \int_{y\in B} \pi(dy)\,k(y,A)$$

Note: $\int f(x)\,p(x)dx \equiv \int f(x)\,p(dx)$



area $= P(dx) = P(x)\,dx$

MCMC: MH Annealed

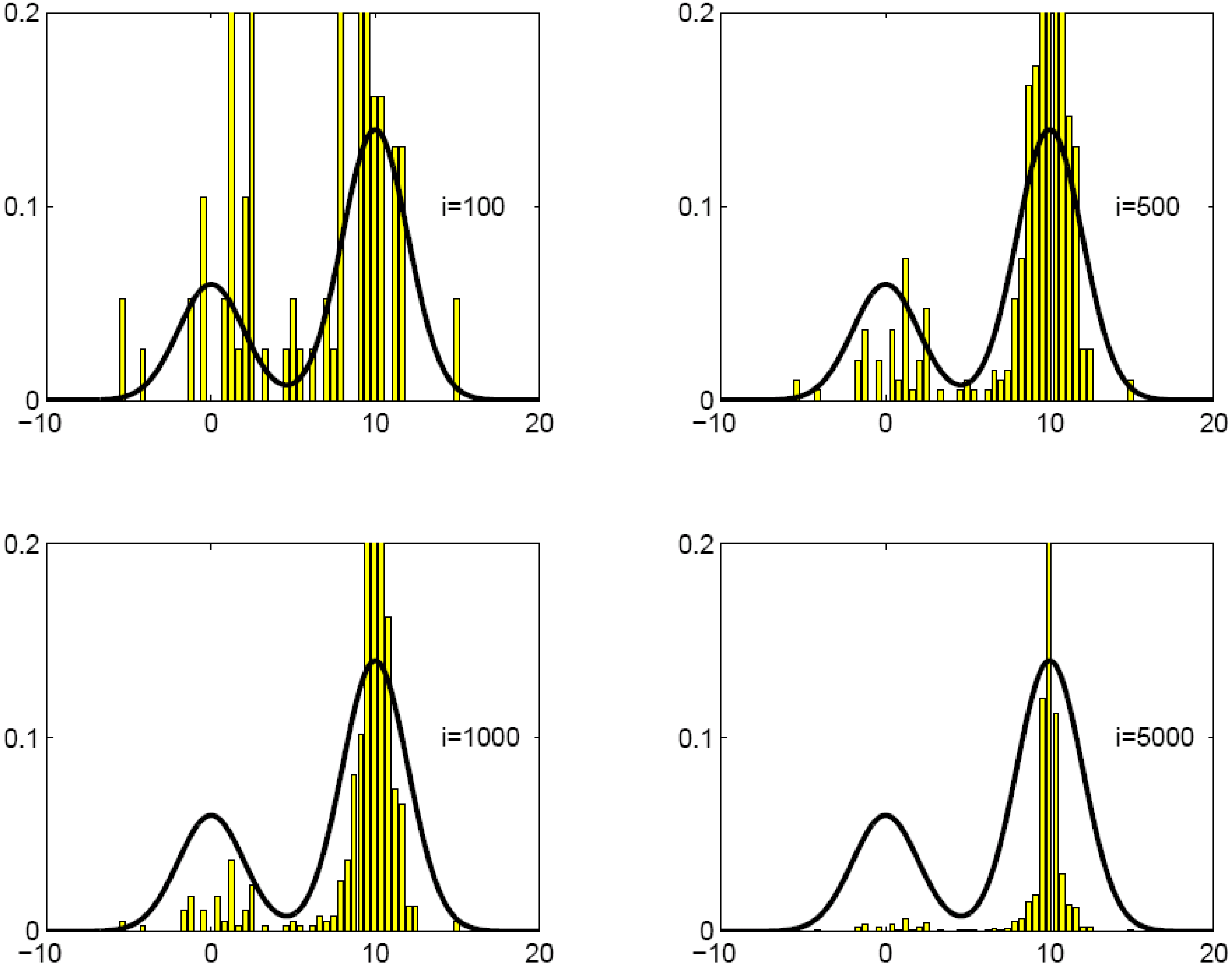
- Initialise $x^{(0)}$ and set $T_0 = 1$.
- For $i = 0$ to $N - 1$
  - Sample $u \sim U_{[0,1]}$.
  - Sample $x^\star \sim q(x^\star | x^{(i)})$.
  - If $u < A(x^{(i)}, x^\star) = \min\left\{1, \frac{p^{\frac{1}{T_i}}(x^\star) q(x^{(i)} | x^\star)}{p^{\frac{1}{T_i}}(x^{(i)}) q(x^\star | x^{(i)})}\right\}$

    $$x^{(i+1)} = x^\star$$

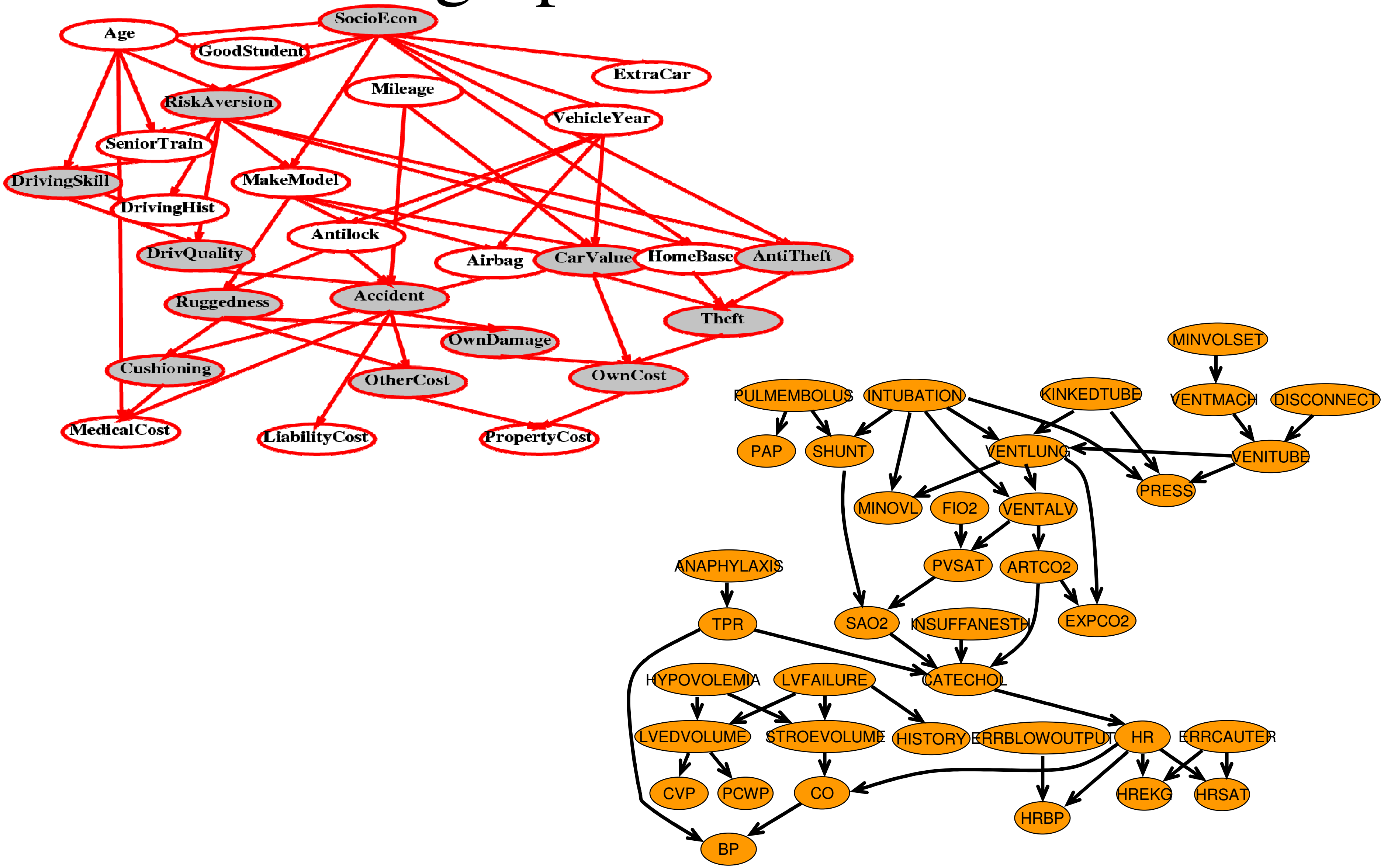
    else

    $$x^{(i+1)} = x^{(i)}$$
  - Set T_{i+1} according to a chosen cooling schedule.

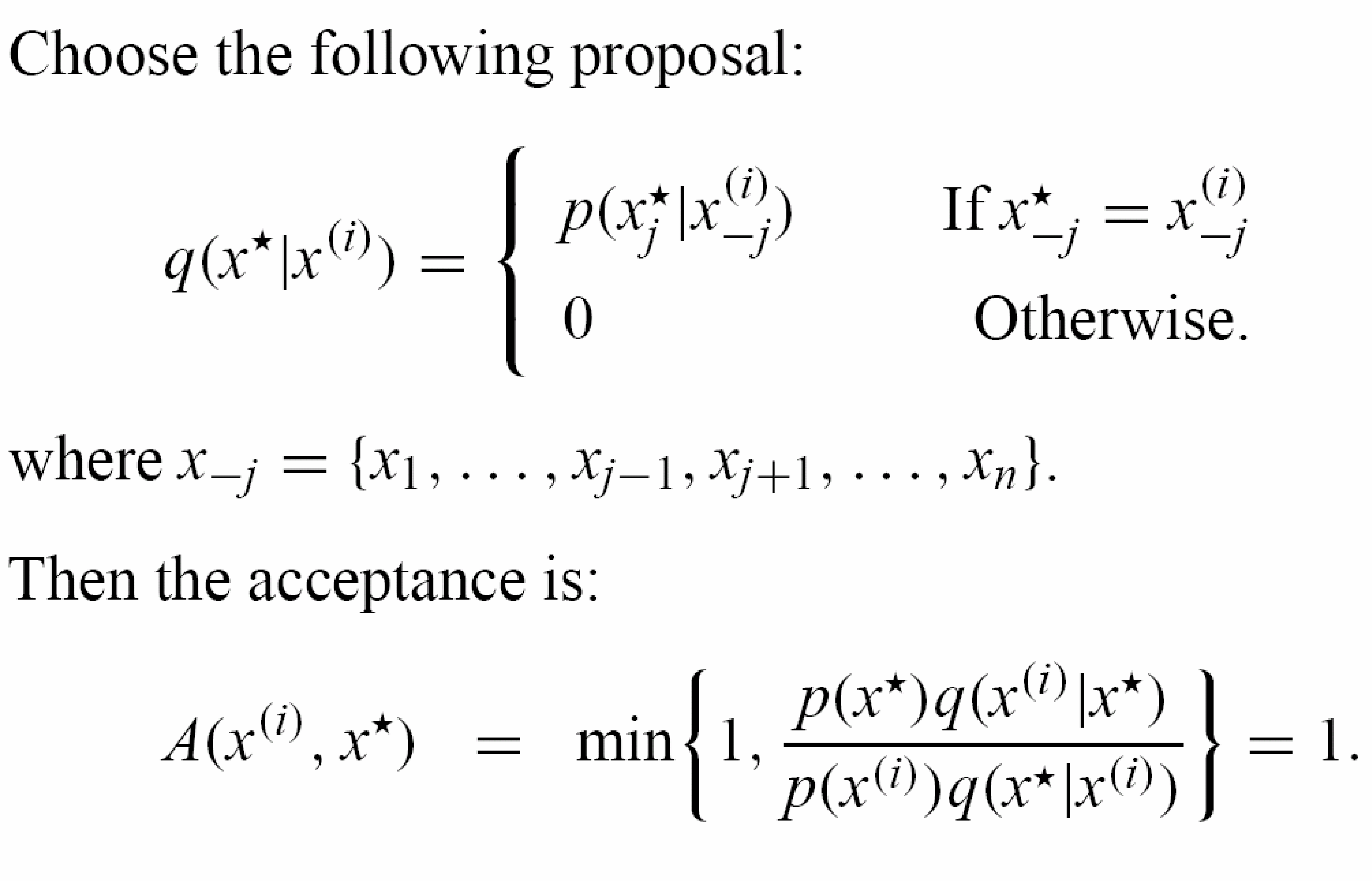
MCMC: MH Annealed



Extending MH to directed probabilistic graphical models



Gibbs Sampling



Choose the following proposal:

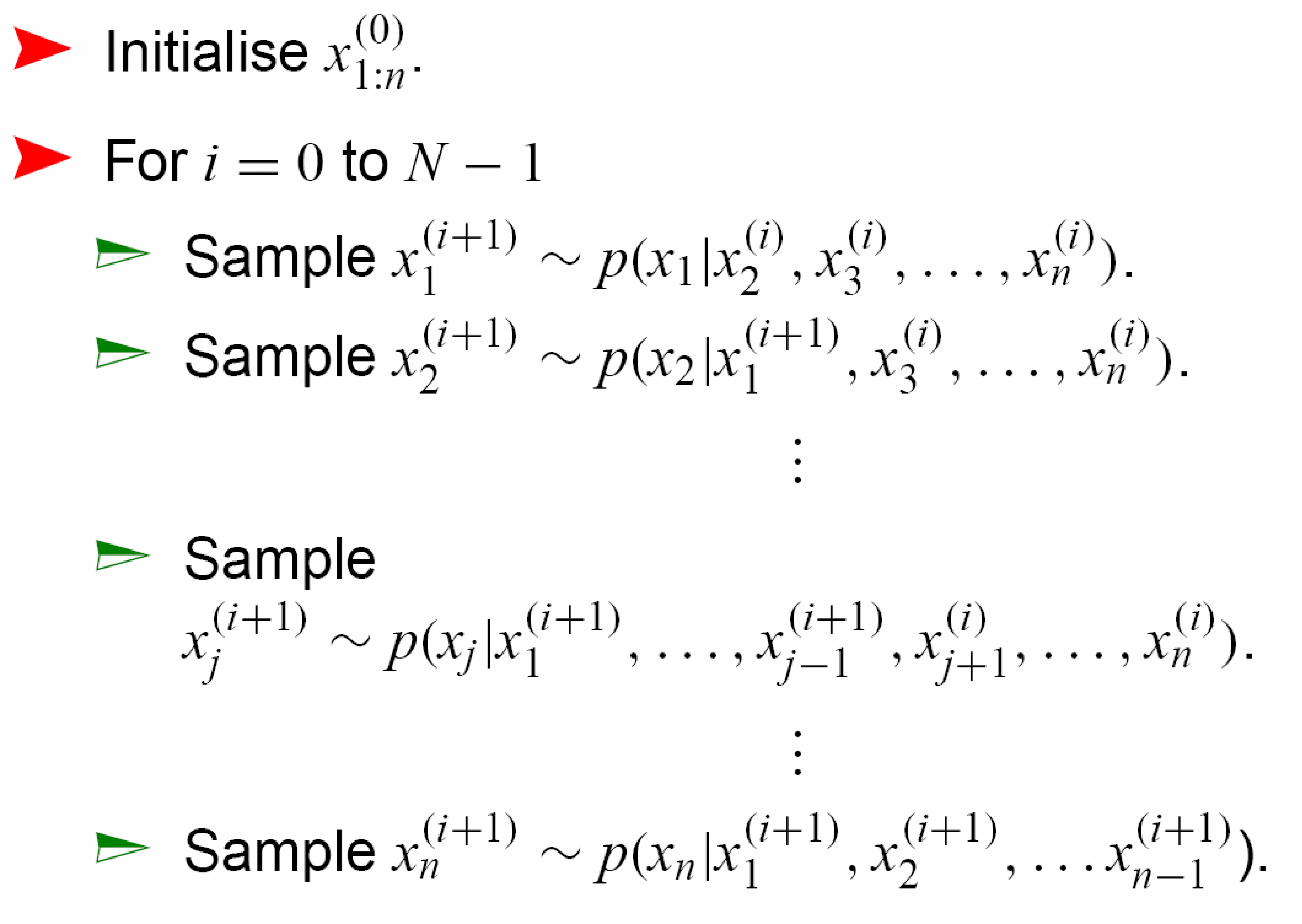
$$q(x^\star|x^{(i)}) = \begin{cases} p(x_j^\star|x_{-j}^{(i)}) & \text{If } x_{-j}^\star = x_{-j}^{(i)} \\ 0 & \text{Otherwise.} \end{cases}$$

where $x_{-j} = \{x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n\}$.

Then the acceptance is:

$$A(x^{(i)}, x^\star) \;=\; \min\left\{1, \frac{p(x^\star)q(x^{(i)}|x^\star)}{p(x^{(i)})q(x^\star|x^{(i)})}\right\} = 1.$$

Gibbs Sampling



- Initialise $x_{1:n}^{(0)}$.
- For $i = 0$ to $N-1$
  - Sample $x_1^{(i+1)} \sim p(x_1 | x_2^{(i)}, x_3^{(i)}, \ldots, x_n^{(i)})$.
  - Sample $x_2^{(i+1)} \sim p(x_2 | x_1^{(i+1)}, x_3^{(i)}, \ldots, x_n^{(i)})$.

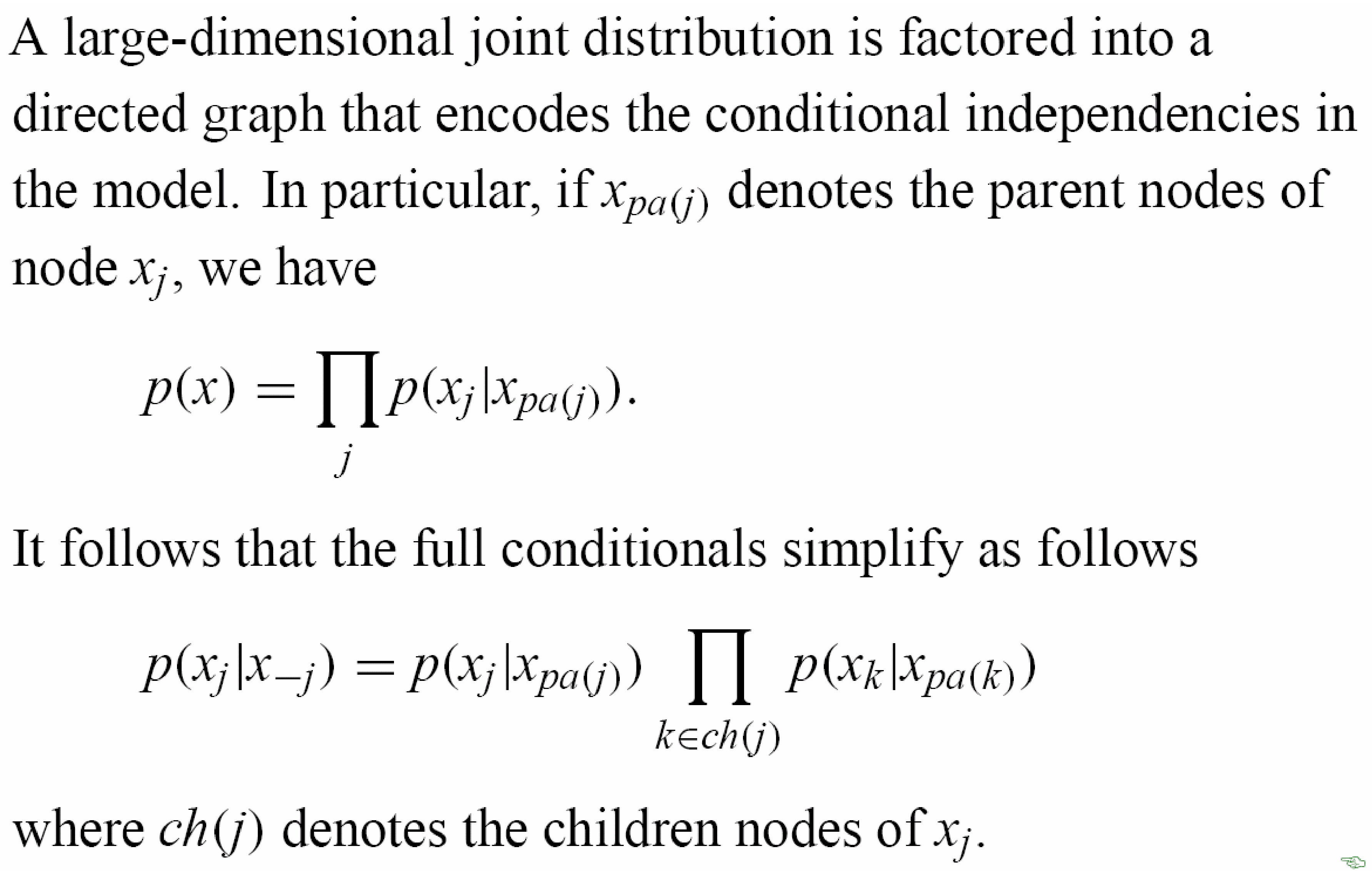
$$\vdots$$

  - Sample $x_j^{(i+1)} \sim p(x_j | x_1^{(i+1)}, \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, \ldots, x_n^{(i)})$.

$$\vdots$$

  - Sample $x_n^{(i+1)} \sim p(x_n | x_1^{(i+1)}, x_2^{(i+1)}, \ldots x_{n-1}^{(i+1)})$.

Gibbs Sampling For Graphical models



A large-dimensional joint distribution is factored into a directed graph that encodes the conditional independencies in the model. In particular, if $x_{pa(j)}$ denotes the parent nodes of node x_j, we have

$$p(x) = \prod_j p(x_j | x_{pa(j)}).$$

It follows that the full conditionals simplify as follows

$$p(x_j | x_{-j}) = p(x_j | x_{pa(j)}) \prod_{k \in ch(j)} p(x_k | x_{pa(k)})$$

where $ch(j)$ denotes the children nodes of x_j.

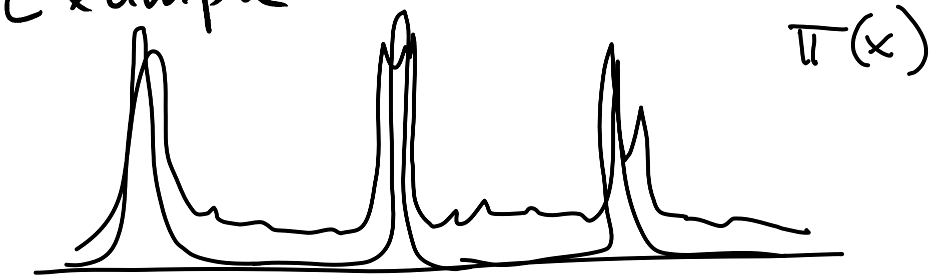
Overview

- History and applications
- Rejection sampling
- Importance sampling
- Particle filters
- Markov chain Monte Carlo
  - Metropolis-Hastings
  - Gibbs sampling
- Advanced Monte Carlo methods
  - Mixtures of kernels
  - Trans-dimensional Monte Carlo
  - Hybrid Monte Carlo
  - Monte Carlo EM
  - Particle methods with artificial dynamics

MH is a Building Block

- Mixtures of MCMC algorithms.
  - Global and local exploration.
  - Reversible jump MCMC.

Example



$q_1(x) \equiv$ FFT proposal

$q_2(x) \equiv$ random walk to explore local modes and discover detail

$$\pi T_1 = \pi$$

$$\pi T_2 = \pi$$

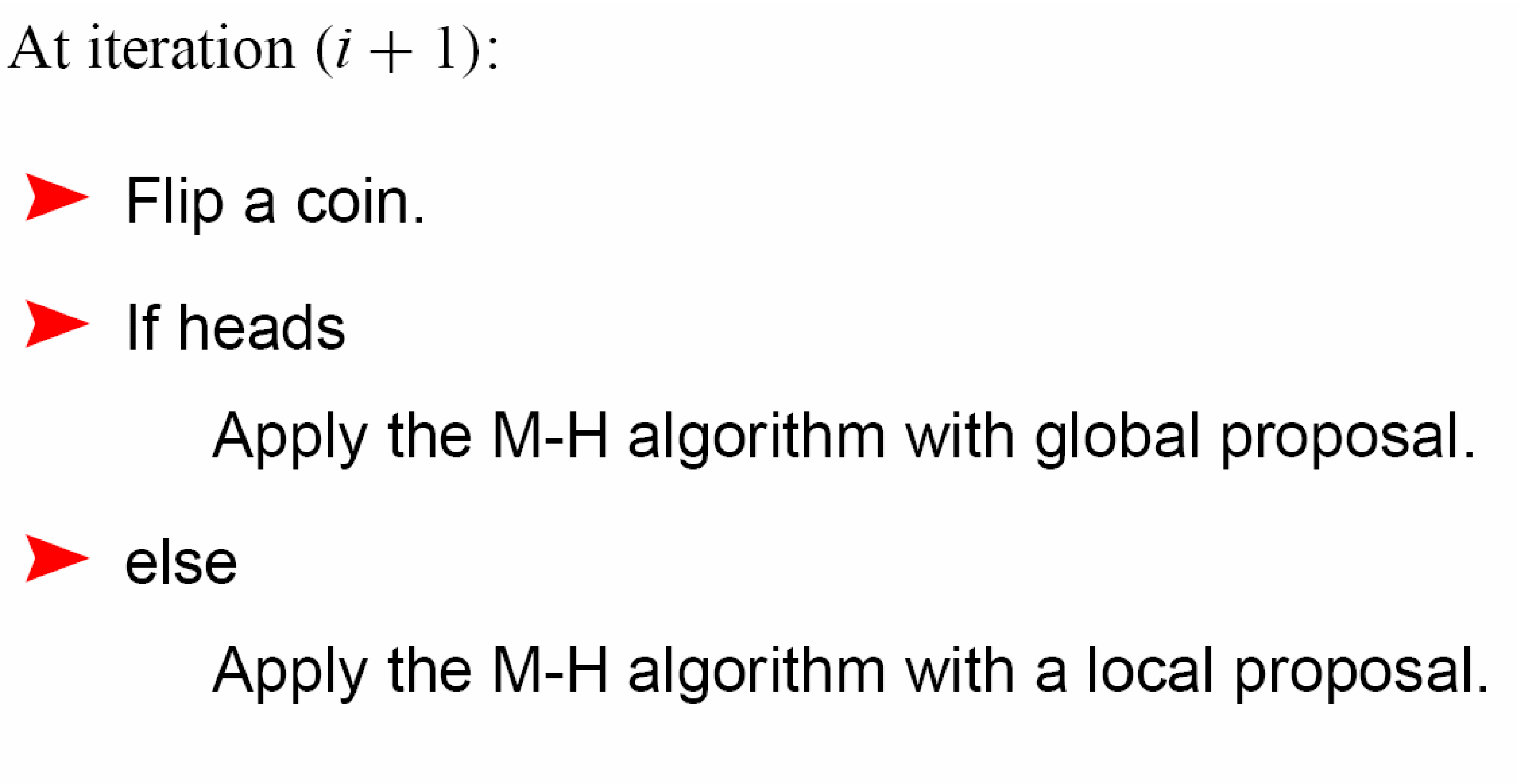
$$\Downarrow$$

$$\alpha \pi T_1 + (1-\alpha) \pi T_2 = \pi$$

$$\pi \left[\alpha T_1 + (1-\alpha) T_2 \right] = \pi$$

The mixture is also a kernel of π.

MH is a Building Block



At iteration $(i+1)$:

- Flip a coin.
- If heads

    Apply the M-H algorithm with global proposal.

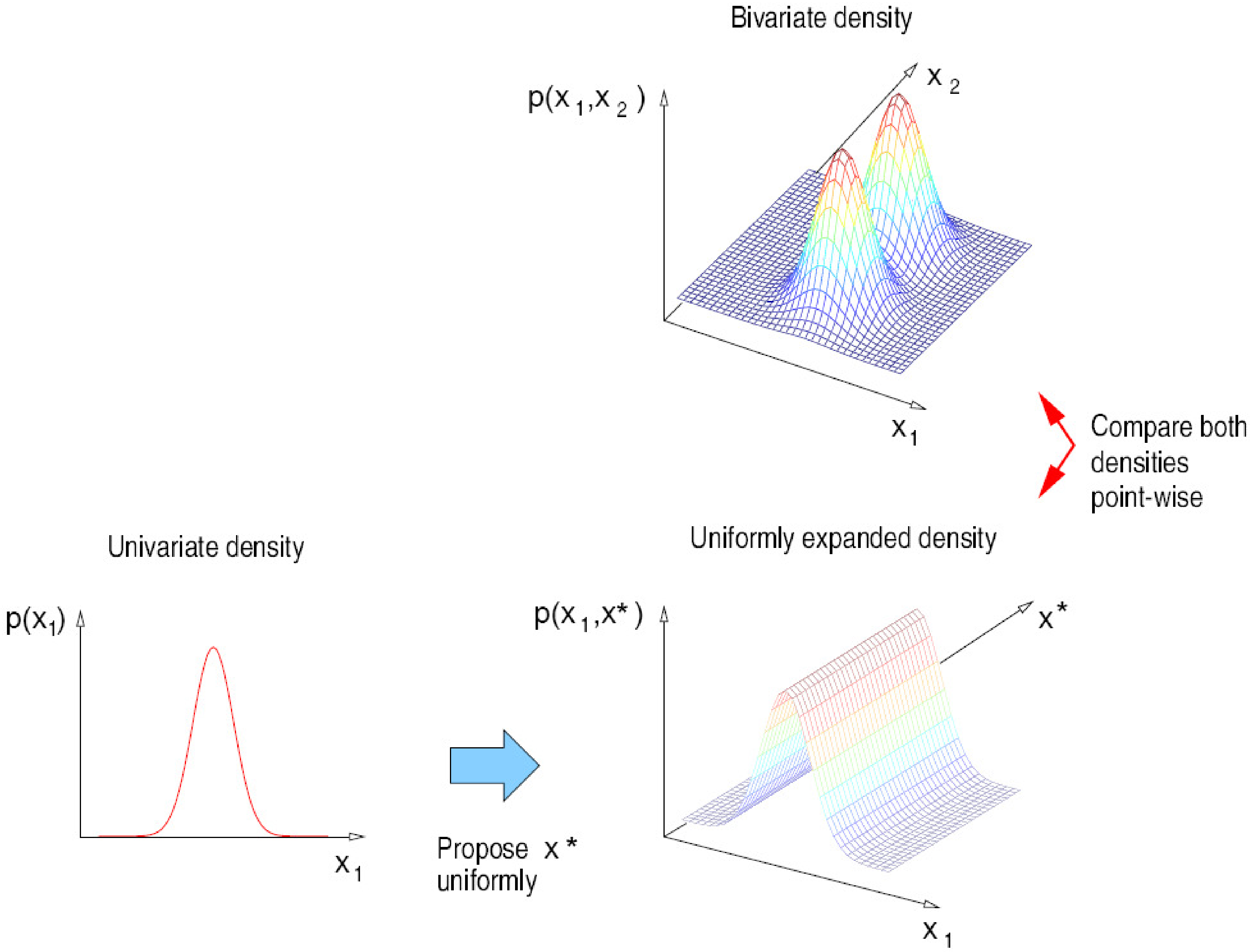
- else

    Apply the M-H algorithm with a local proposal.

Trans-Dimensional MCMC and the Reversible Jump Algorithm of Peter Green: Use a mixture of dimension jumping algorithms

```
1. Initialisation: set $(k^{(0)}, \mu^{(0)})$.
2. For $i = 0$ to $N - 1$
    –  Sample $u \sim \mathcal{U}_{[0,1]}$.
    –  If $(u \leq b_k)$
         •  then "birth" move.
         •  else if $(u \leq b_k + d_k)$ then "death" move.
         •  else if $(u \leq b_k + d_k + s_k)$ then "split" move.
         •  else if $(u \leq b_k + d_k + s_k + m_k)$ then "merge" move.
         •  else update.
       End If.
    –  Sample other parameters.
```

Must be Careful with Measures !



Compare both densities point-wise

Propose x * uniformly

Trans-Dimensional MCMC

$$K(x,B) = \sum_{\underset{\text{mixture index}}{\nearrow k}} \int_B \alpha_k \, q_k(dy|x) + \mathbb{I}_{x\in B}\left\{1 - \sum_k \int_{\mathcal{X}} \alpha_k \, q_k(dx'|x)\right\}$$

So we have a mixture of moves (or dimensionally jumping algorithms)

Next, we need to check that this is a valid Metropolis-Hastings.

Trans-Dimensional MCMC

Since we need to compare distributions, not densities, we need a commos space.

so, if

Θ^{k_1} is completed by $u_1 \sim g_1(\cdot)$

Θ^{k_2} " " " $u_2 \sim g_2(\cdot)$

we have a $\overset{\text{deterministic}}{\text{bijection}}$ $(\Theta^{k_2}, u_2) = T(\Theta^{k_1}, u_1)$

and the acceptance term becomes:

$$A = \min\left\{1, \frac{\pi(k_2, \Theta^{k_2})}{\pi(k_1, \Theta^{k_1})} \frac{\pi_{21}}{\pi_{12}} \frac{g_2(u_2)}{g_1(u_1)} \overbrace{\left| \frac{\partial T(\Theta^{k_1}, u_1)}{\partial(\Theta^{k_2}, u_2)} \right|}^{\text{Jacobian}} \right\}$$

Trans-Dimensional MCMC

Example: Mixture of Gaussians with unknown number of components

$$P(x|\mu) = \sum_{i=1}^{c} \pi_i \mathcal{N}(x_j; \mu_i, 1)$$

Merge Move

$$\mu = \frac{\mu_1 + \mu_2}{2}$$

Split move

$$\begin{cases} \mu_1 = \mu - u\beta \\ \mu_2 = \mu + u\beta \end{cases}$$

$u \sim N(0,1)$, $\beta \equiv$ parameter

For reversibility, $\|\mu_1 - \mu_2\| \leq 2\beta$ in merge.

Trans-Dimensional MCMC

SPLIT MOVE :

proposal to merge

$$A_{split} = \min\left\{1, \frac{P(\mu_{K+1}, K+1)}{P(\mu_K, K)} \frac{\frac{1}{K+1}\, m_{K+1}}{\frac{1}{K}\, s_K\, P(u)} J_{split}\right\}$$

Proposal to split

$$J_{split} = \left| \frac{\partial(\mu_1, \mu_2)}{\partial(\mu, u)} \right| = \begin{vmatrix} \partial\mu_1/\partial\mu & \partial\mu_2/\partial\mu \\ \partial\mu_1/\partial u & \partial\mu_2/\partial u \end{vmatrix} = \begin{vmatrix} 1 & 1 \\ -\beta & \beta \end{vmatrix} = 2\beta$$

$$\begin{cases} \mu_1 = \mu - u\beta \\ \mu_2 = \mu + u\beta \end{cases}$$

Trans-Dimensional MCMC

Merge Move

$$A_{merge} = \min\left\{1, \frac{P(k-1, \mu_{k-1})}{P(k, \mu_k)} \frac{s_{k-1}}{m_k} \frac{\frac{1}{k-1}}{\frac{1}{k}} \times J_{merge}\right\}$$

$$J_{merge} = \left| \frac{\partial(\mu, u)}{\partial(\mu_1, \mu_2)} \right| = \begin{vmatrix} \frac{\partial \mu}{\partial \mu_1} & \frac{\partial u}{\partial \mu_1} \\ \frac{\partial \mu}{\partial \mu_2} & \frac{\partial u}{\partial \mu_2} \end{vmatrix} = \frac{1}{2\beta}$$

Trans-Dimensional MCMC

$\boxed{\text{BIRTH}}$

$$\mu_{1:k+1} = \{\mu_{1:k}, \mu^*\}$$

$$A_{birth} = \min\left\{1, \frac{P(k+1, \mu_{k+1})}{P(k, \mu_k)} \frac{d_{k+1}}{b_k} \frac{\frac{1}{k+1}}{P(\mu^*)} J_{Birth}\right\}$$

↑ new component

$$J_{Birth} = \left|\frac{\partial \mu_{1:k+1}}{\partial(\mu_{1:k}, \mu^*)}\right| = \begin{vmatrix} \cancelto{1}{\frac{\partial \mu_{1:k}}{\partial \mu_{1:k}}} & 0 \\ \cancelto{0}{\partial} & 1 \end{vmatrix} = 1$$

$\boxed{\text{DEATH}}$

This is one of the μ in $\mu_{1:k}$!

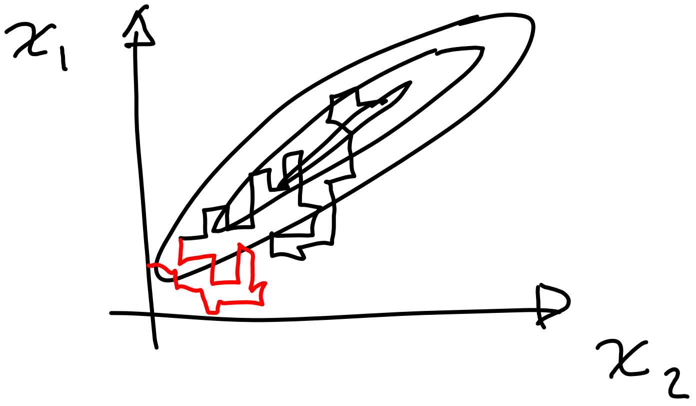
$$A_{death} = \min\left\{1, \frac{P(k-1, \mu_{k-1})}{P(k, \mu_k)} \frac{P(\mu^*)}{\frac{1}{k}} \times \frac{b_{k-1}}{d_k} \times \cancelto{1}{J_{death}}\right\}$$

MH is a Building Block

- Cycles of MCMC algorithms.
  - Large dimensional vectors.
  - Gibbs sampling.

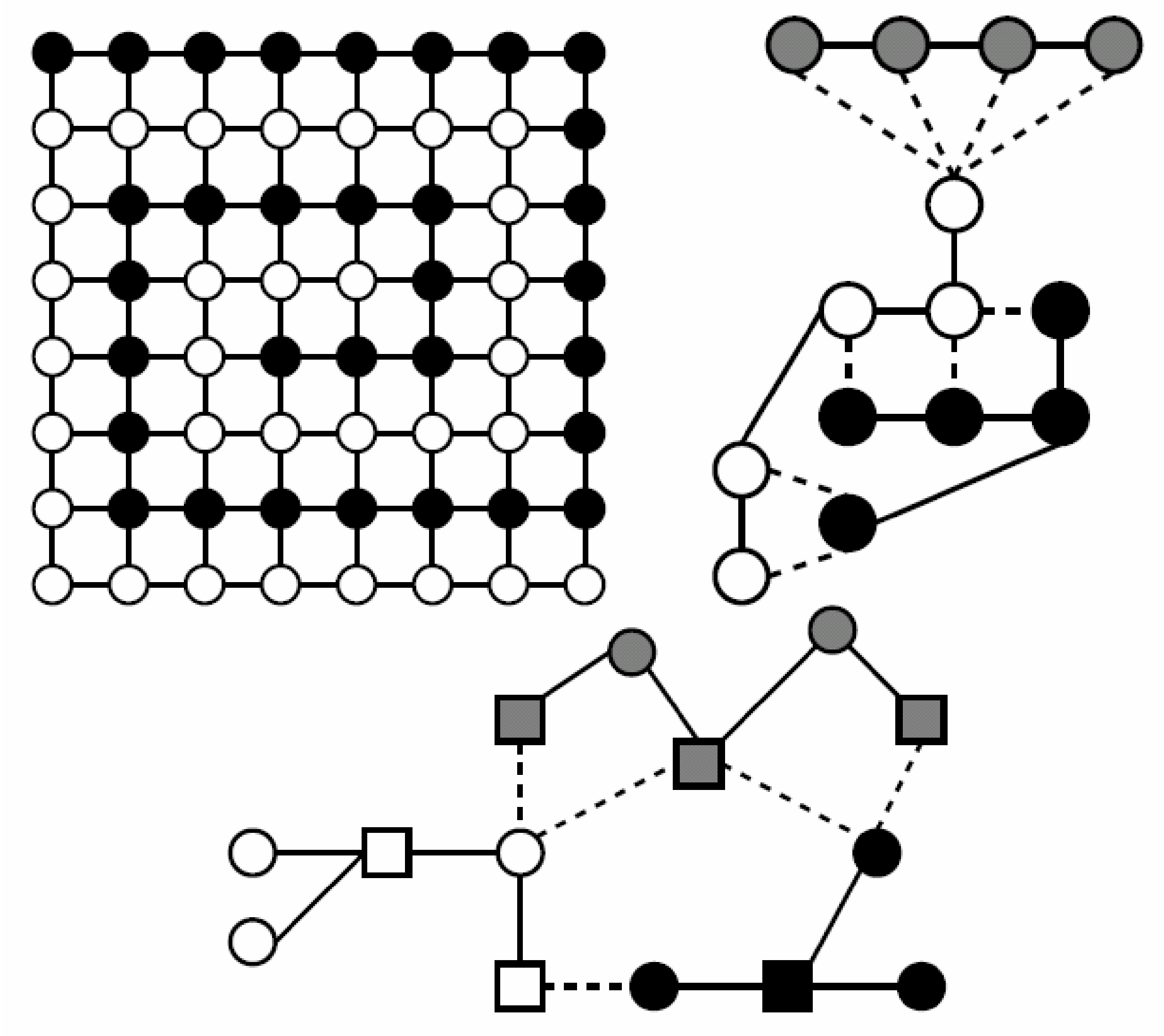
MH is a Building Block

- **Idea**: Split the high dimensional vector x into blocks $\{x_{b1}, \ldots, x_{bn}\}$.
- **Cycle**: sample each block using an MH algorithm with invariant distribution $p(x_{bi}|x_{-bi})$ and proposal distribution $q(x_{bi})$, where $x_{-bi} = \{\text{All blocks except } x_{bi}\}$.
- Block highly correlated variables.



Chain can take a long time to mix when variables are correlated.

Collapsing and Blocking



Monte Carlo EM

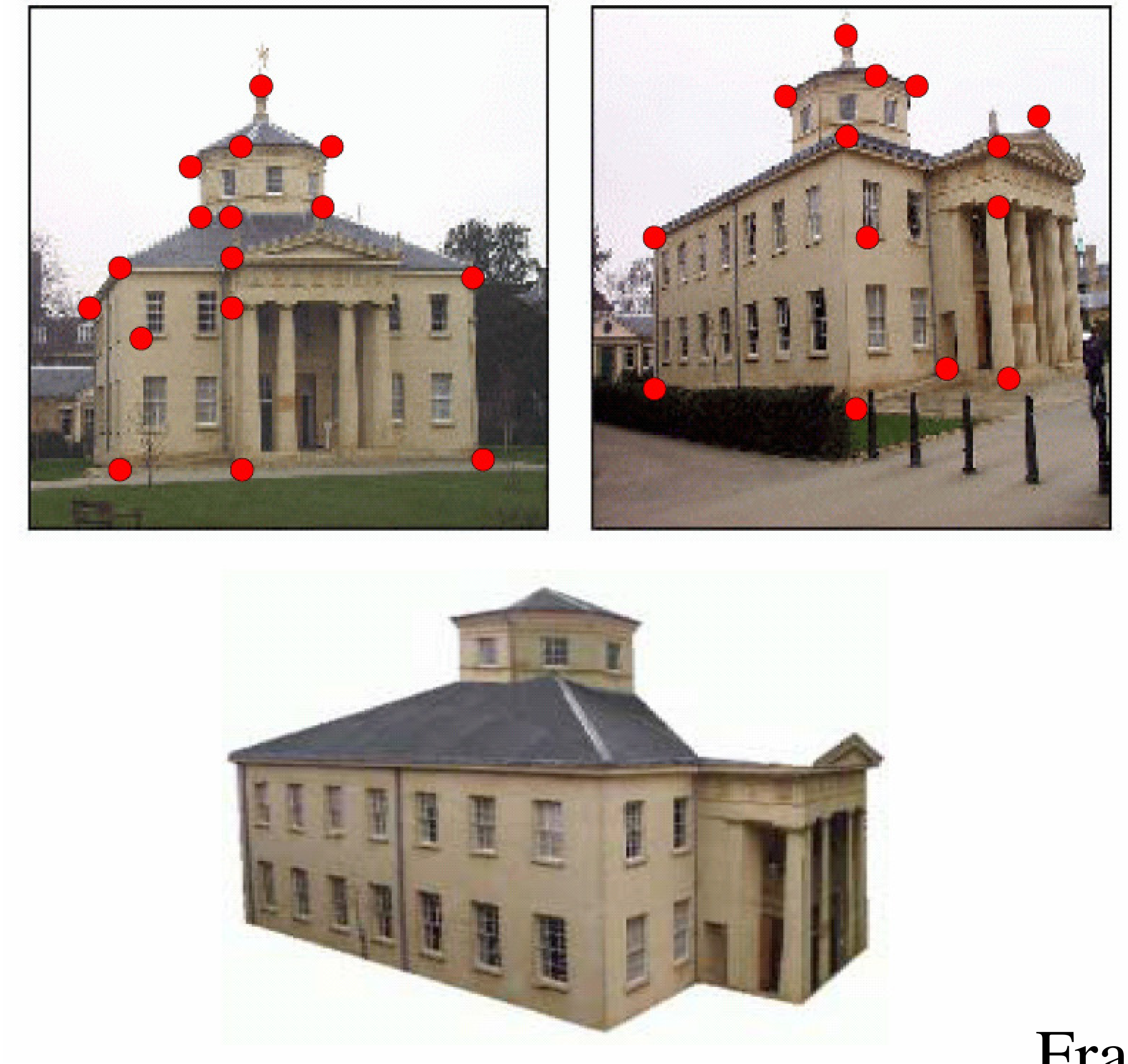
1. **E step:** Compute the expected value of the complete log-likelihood function with respect to the distribution of the hidden variables

$$Q(\theta) = \int_{X_h} \log\left(p(x_h, x_v|\theta)\right) p(x_h|x_v, \theta^{(\text{old})}) dx_h,$$

where $\theta^{(\text{old})}$ refers to the value of the parameters at the previous time step.

2. **M step:** Perform the following maximisation $\theta^{(\text{new})} = \arg\max_{\theta} Q(\theta)$.

Application of MCEM



Frank Dellaert

Auxiliary Variable Samplers

- It is often easier to sample from an augmented distribution $p(x, u)$, where u is an auxiliary variable, than from $p(x)$.

- It is possible to obtain marginal samples $x^{(i)}$ by sampling $(x^{(i)}, u^{(i)})$ according to $p(x, u)$ and, then, ignoring the samples $u^{(i)}$.

- This very useful idea was proposed in the physics literature (Swendsen and Wang, 1987).

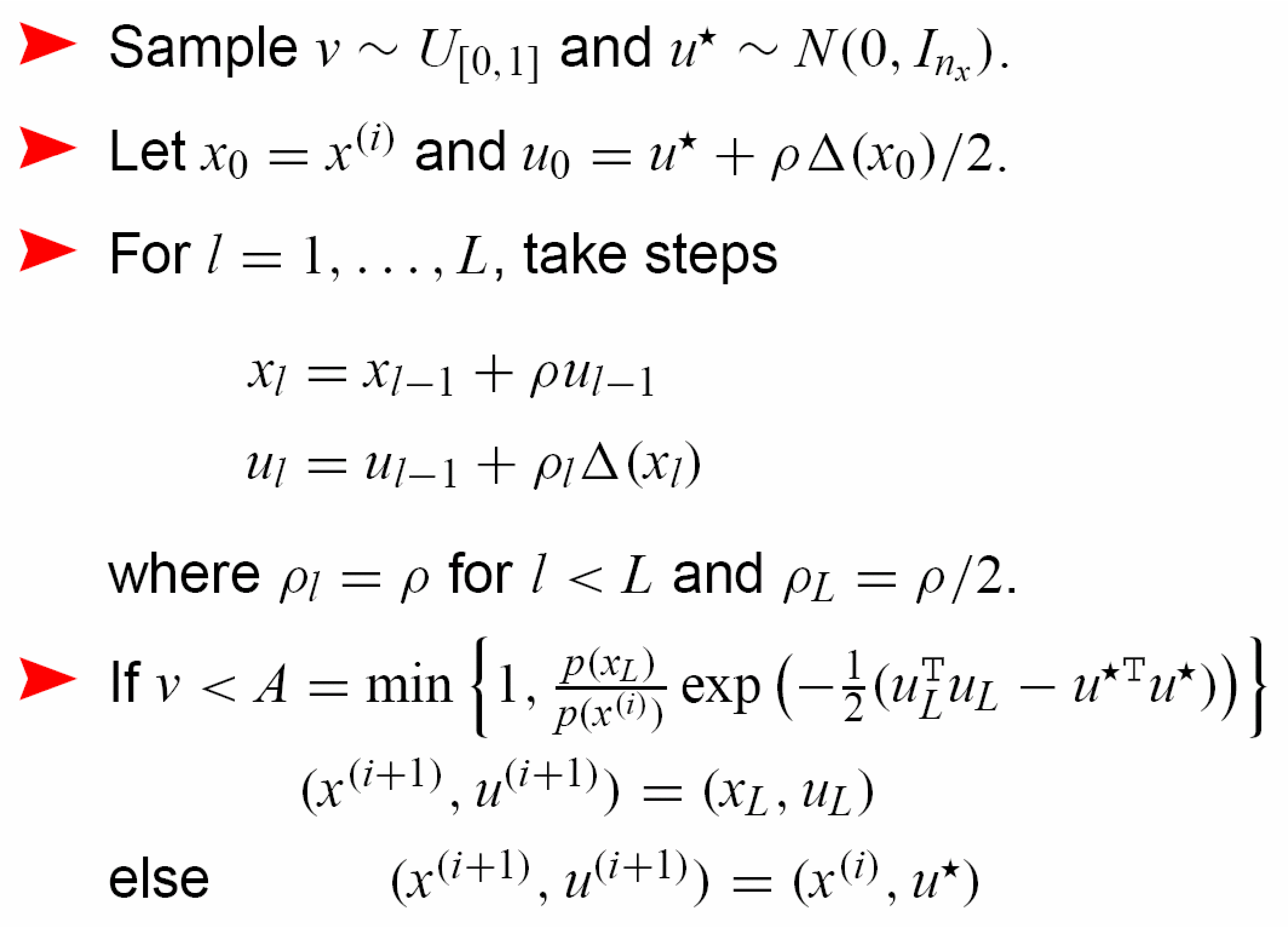
Hybrid Monte Carlo

- The idea is to exploit gradient information.
- Define the extended target distribution:

$$p(x, u) = p(x) N(u; 0, I_{n_x}).$$

- Introduce the gradient vector: $\Delta(x) = \partial \log p(x) / \partial x$
- Introduce the parameters ρ and L.
- Next we “leapfrog”.

Hybrid Monte Carlo



- Sample $v \sim U_{[0,1]}$ and $u^{\star} \sim N(0, I_{n_x})$.
- Let $x_0 = x^{(i)}$ and $u_0 = u^{\star} + \rho\Delta(x_0)/2$.
- For $l = 1, \ldots, L$, take steps

$$x_l = x_{l-1} + \rho u_{l-1}$$

$$u_l = u_{l-1} + \rho_l \Delta(x_l)$$

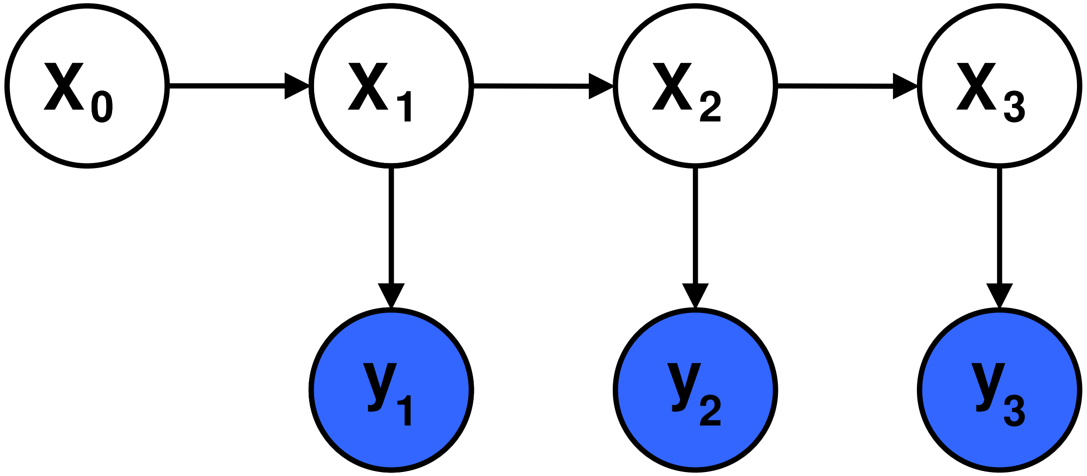
where $\rho_l = \rho$ for $l < L$ and $\rho_L = \rho/2$.

- If $v < A = \min\left\{1, \frac{p(x_L)}{p(x^{(i)})} \exp\left(-\frac{1}{2}(u_L^{\mathrm{T}} u_L - u^{\star\mathrm{T}} u^{\star})\right)\right\}$

$$(x^{(i+1)}, u^{(i+1)}) = (x_L, u_L)$$

else $$(x^{(i+1)}, u^{(i+1)}) = (x^{(i)}, u^{\star})$$

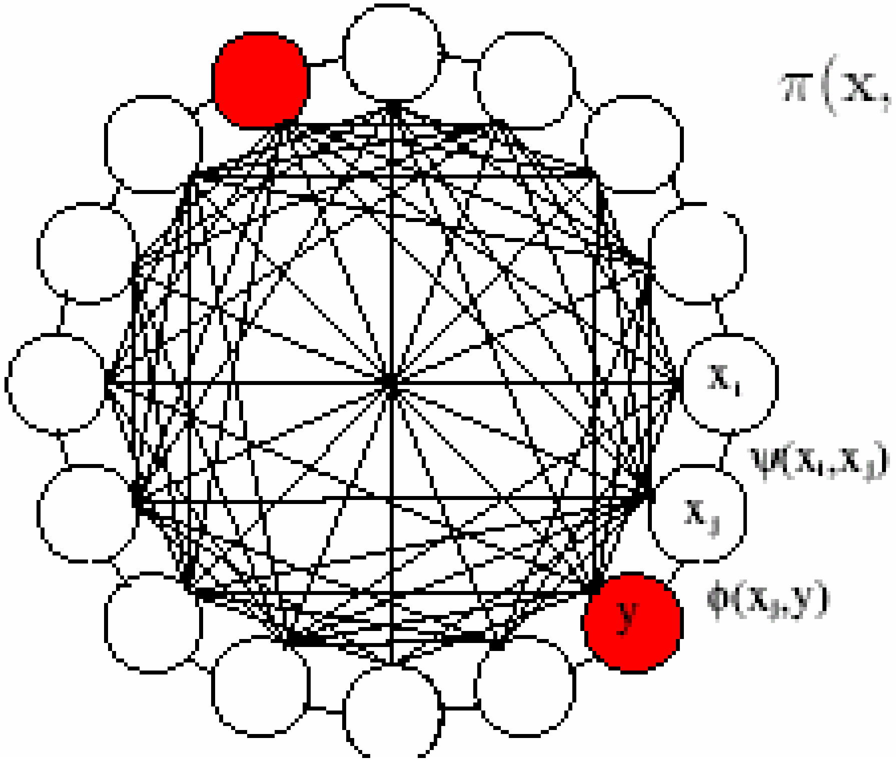
Dynamic models are growing probabilistic models



$$p(x_0)\,p(x_1|x_0)\,p(y_1|x_1)\,p(x_2|x_1)\,p(y_2|x_2)\,p(x_3|x_2)\,p(y_3|x_3)$$

The new extension: Hot coupling

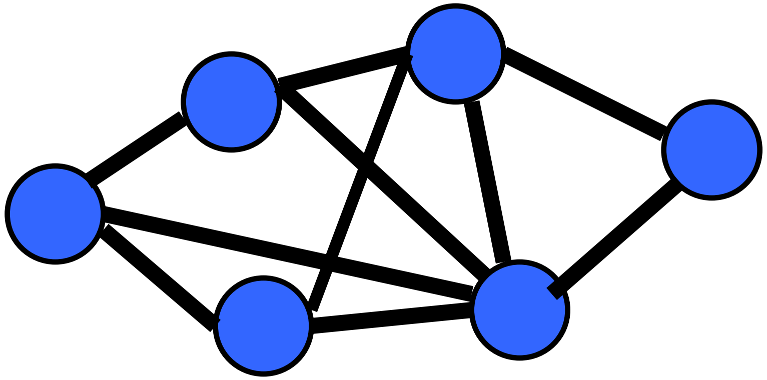
We want to sample from a static distribution, e.g. a fully connected Boltzmann machine, CRF or MRF. We do this by constructing an online sequence of distributions that converges to the right distribution.



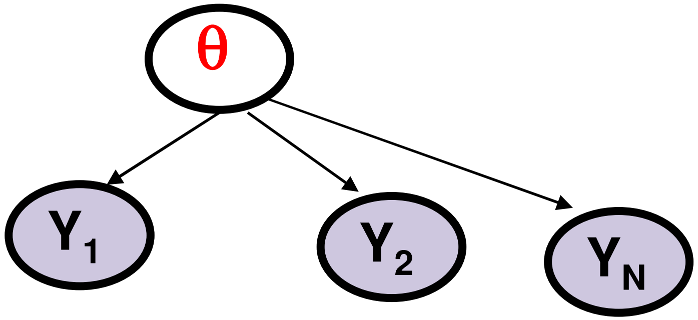
$$\pi(\mathrm{x},\mathrm{y}) = \frac{1}{Z}\prod_{i\in\mathcal{V}}\phi(x_i,y_i)\prod_{(i,j)\in\mathcal{E}}\psi(x_i,x_j)$$

Idea: start with a tractable model and grow it with succesive particle corrections until we reach the final static model.

The new extension: Hot coupling



Online data processing is equivalent to adding constraints sequentially



The same idea is adopted to carry out Bayesian experimental design, rare event simulation and stochastic control.

New particle methods

This is how the artificial sequence is constructed (del Moral, Doucet and Jasra), where K(.) is an MCMC proposal kernel:

$$\widetilde{\pi}_n(\mathbf{x}_{1:n}) = \pi_n(\mathbf{x}_n)\widetilde{\pi}_{n-1}(\mathbf{x}_{1:n-1}|\mathbf{x}_n)$$

$$\widetilde{\pi}_{n-1}(\mathbf{x}_{1:n-1}|\mathbf{x}_n) = \prod_{t=1}^{n-1} L_t(\mathbf{x}_t|\mathbf{x}_{t+1})$$

$$w_n \propto \frac{\pi_n(\mathbf{x}_n)}{\pi_{n-1}(\mathbf{x}_{n-1})}\frac{L_{n-1}(\mathbf{x}_{n-1}|\mathbf{x}_n)}{K_n(\mathbf{x}_n|\mathbf{x}_{n-1})}$$

New particle methods

A simple choice of the L(.) leads to annealed importance sampling (Jarzynski, Neal).

$$L_{n-1}(\mathbf{x}_{n-1}|\mathbf{x}_n) = \frac{\pi_{n-1}(\mathbf{x}_{n-1})K_n(\mathbf{x}_n|\mathbf{x}_{n-1})}{\pi_{n-1}(\mathbf{x}_n)}$$

$$w_n \propto \frac{\pi_n(\mathbf{x}_n)}{\pi_{n-1}(\mathbf{x}_n)}$$

But we can be more general (and **optimal**):

$$w_n \propto \frac{f_n(\mathbf{x}_n)}{\int f_{n-1}(\mathbf{x}_{n-1})K_n(\mathbf{x}_{n-1},\mathbf{x}_n)d\mathbf{x}_{n-1}}$$

Particle solution for experimental design

We construct an artificial sequence (Peter Muller) where simulations are “data”, while searching over good policies

$$u(a) = \int u(a,x)p(x|a)dx \qquad x = \{y,\theta\}$$

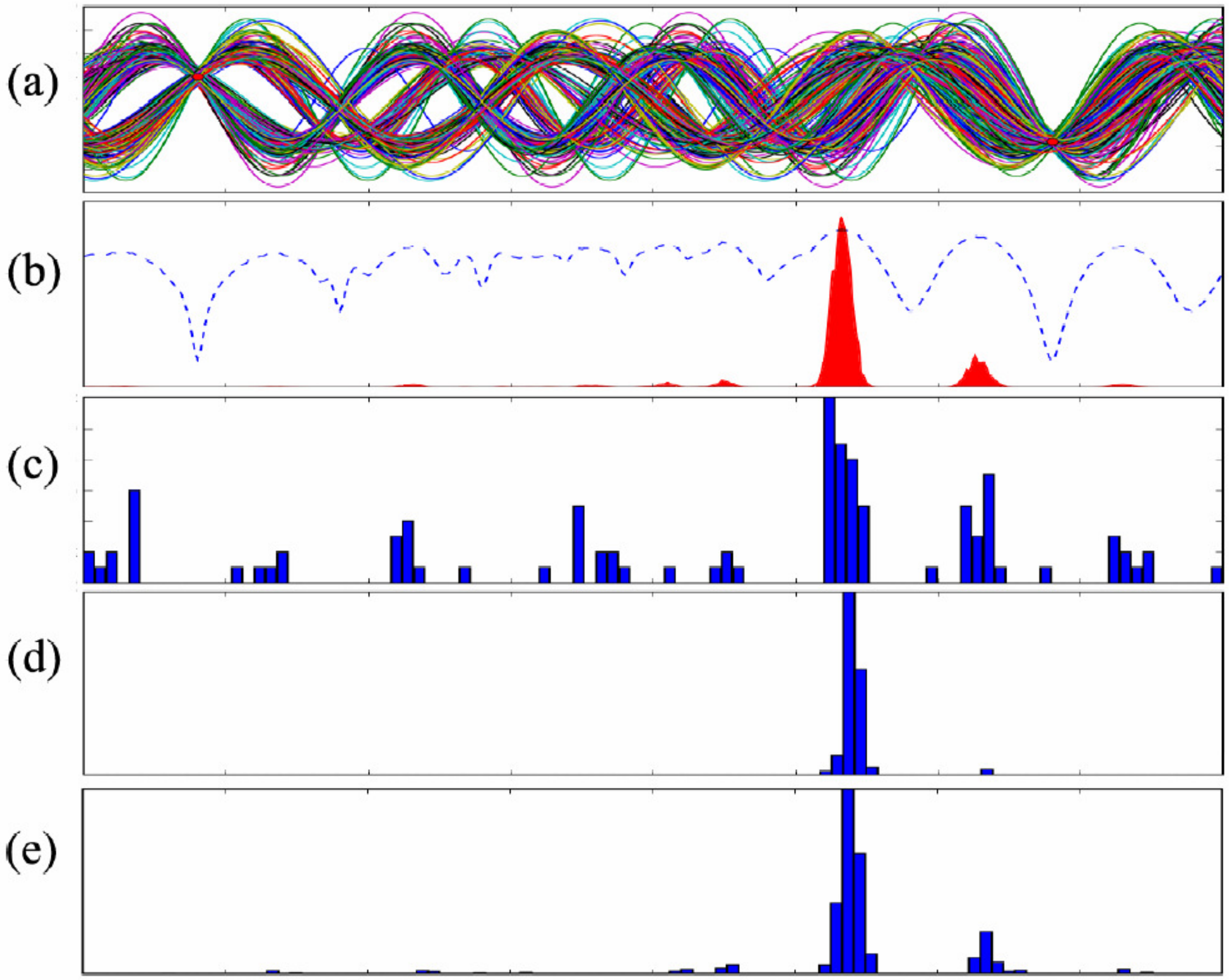
$$u^2(a) = \int\int u(a,x_1)u(a,x_2)p(x_1|a)p(x_2|a)dx_{1:2}$$

$$u^m(a) = \int\cdots\int \prod_{i=1}^{m} u(a,x_i)p(x_i|a)dx_{1:m}$$

$$w_n \propto \frac{u(a_t,x_t)p(x_t|a_t)}{u(a_{t-1}x_{t-1})p(x_{t-1}|a_{t-1})}\frac{L(\cdot)}{K(\cdot)}$$

Choosing input data to learn parameters

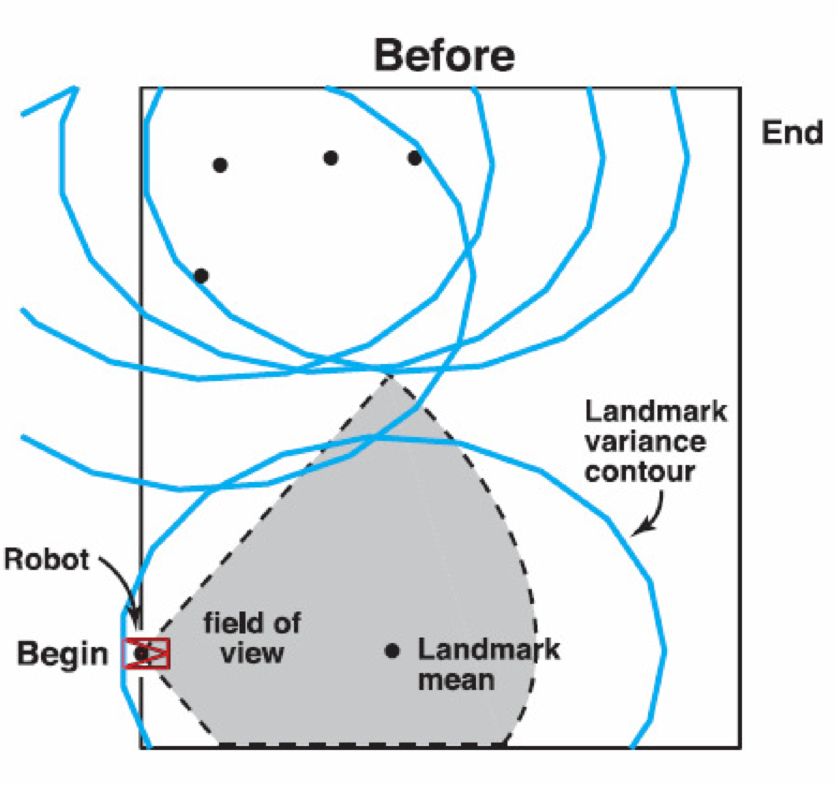
$$y = f(x; A, \omega, \rho) = A \sin\left(2\pi[(d\omega) + \rho]\right)$$



Hendrik
Kueck

Monte Carlo Planning: Problem statement

- The robot, with a rough prior map, has to accomplish a series of tasks in an environment (e.g. reaching End).
- It chooses a parameterized path $\pi^*(\boldsymbol{\theta})$ so as to learn the most about its own pose and the location of navigation landmarks (posterior map), while accomplishing tasks.



POMDP Formulation

- State (robot and landmark locations) $\mathbf{x}_t$
- Observations $\mathbf{y}_t$
- Actions (PID regulation about policy path) $\mathbf{u}_t \sim \pi(\boldsymbol{\theta})$
- Transition model $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$
- Observation model $p(\mathbf{y}_t|\mathbf{x}_t)$
- Cost function: Average Mean Square Error

$$C^{\pi}_{AMSE} = \mathbb{E}_{p(\mathbf{x}_T, \mathbf{y}_{1:T}|\boldsymbol{\pi})} \left[\left(\widehat{\mathbf{x}}_T - \mathbf{x}_T\right)\left(\widehat{\mathbf{x}}_T - \mathbf{x}_T\right)^T \right]$$

$$\widehat{x} = \mathbb{E}_{p(\mathbf{x}_T|\mathbf{y}_{1:T}, \boldsymbol{\pi})} [\mathbf{x}_t]$$

- This cost function is very nasty!
  - It is a function of the **belief state, nonlinear, non-Gaussian and typically not differentiable**.
  - Classical reinforcement learning, LQG, policy gradients and dynamic programming do not apply. A new method is needed.

Algorithm

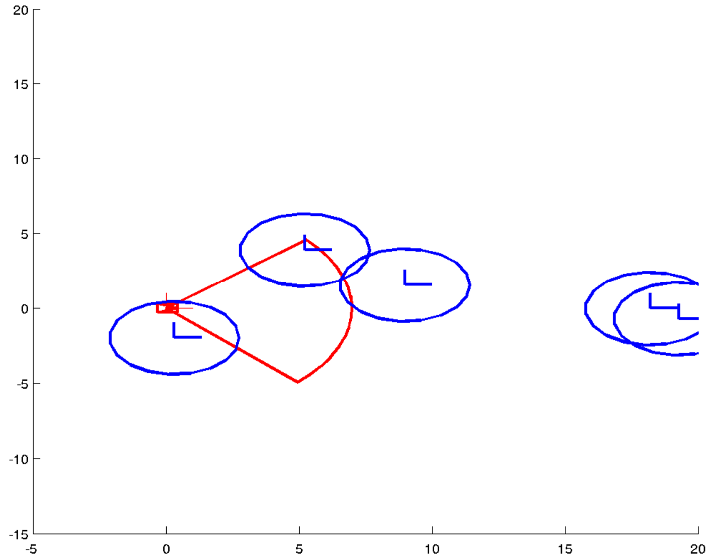
1. Given the current policy parameters, simulate future states, actions and observations.

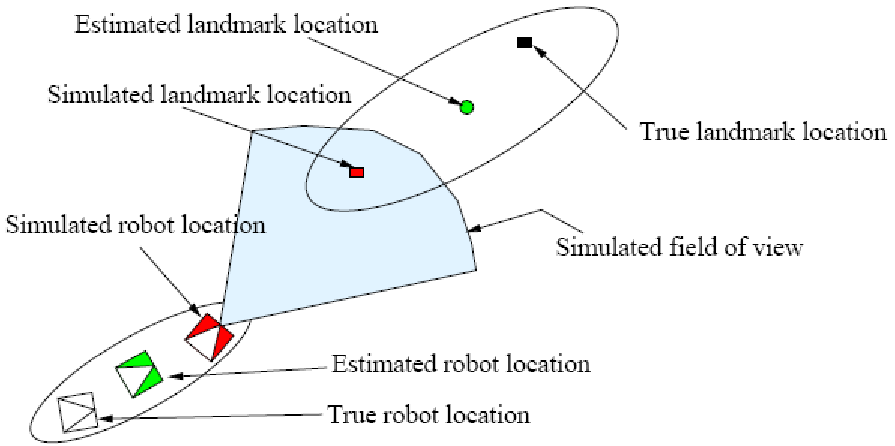
2. Approximate the AMSE cost with a Monte Carlo estimate.

3. Given the new AMSE estimate, update the policy parameters using an active learning approach.

Monte Carlo Simulation

1. Sample states from the prior.
2. Follow the policy path by generating actions with a PID controller.
3. While doing this, sample states from the transition prior and “fictitious” measurements from the observation model.
4. For each of these samples, use a Bayesian SLAM filter to approximate the posterior mean. Yes, this is very expensive.





Evaluation Component

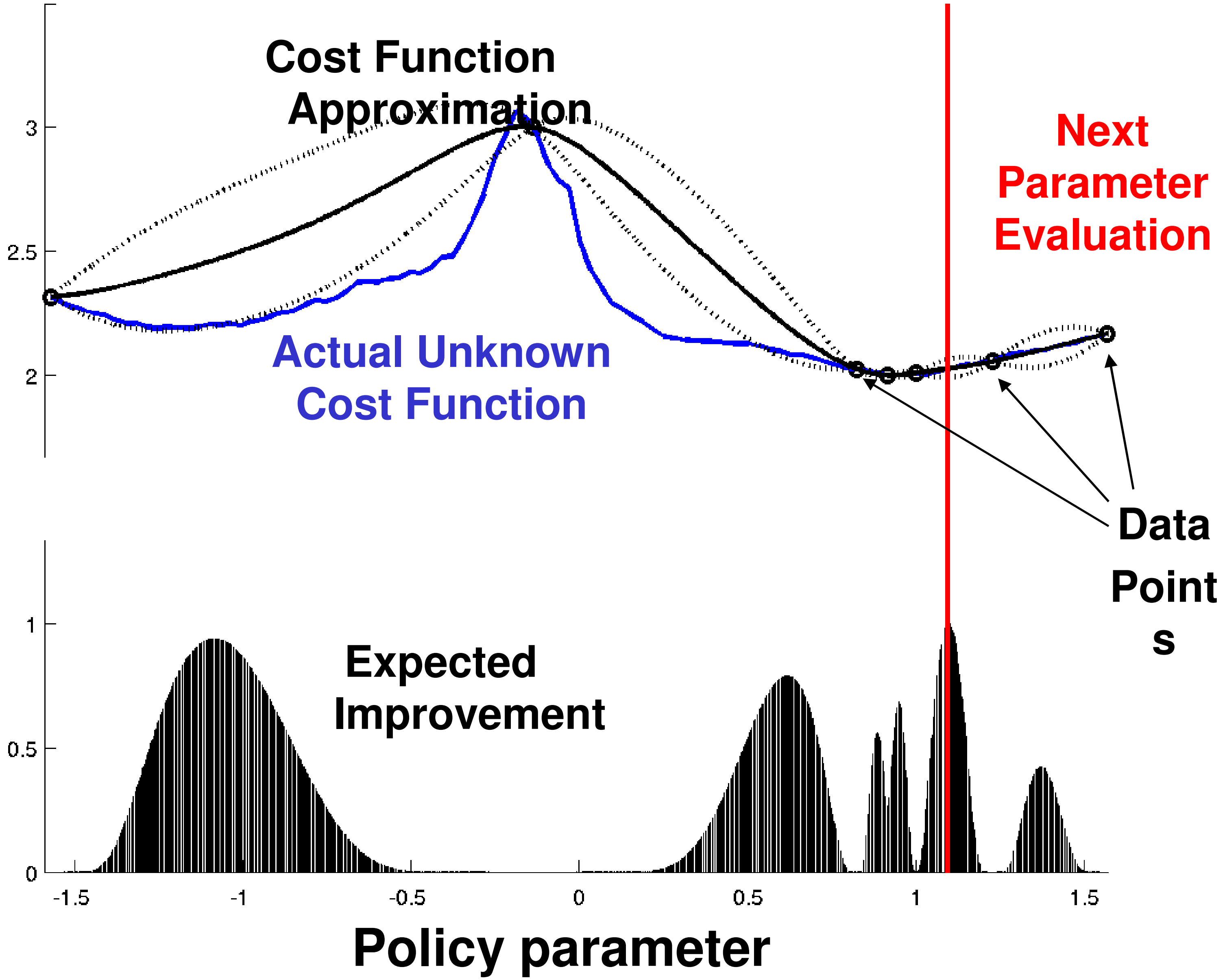
- Adopt a simple Monte Carlo estimate:

$$\widehat{C}^{\pi}_{AMSE} = \frac{1}{N}\sum_{i=1}^{N}[(\widehat{\mathbf{x}}_T^{(i)} - \mathbf{x}_T^{(i)})(\widehat{\mathbf{x}}_T^{(i)} - \mathbf{x}_T^{(i)})^T]$$

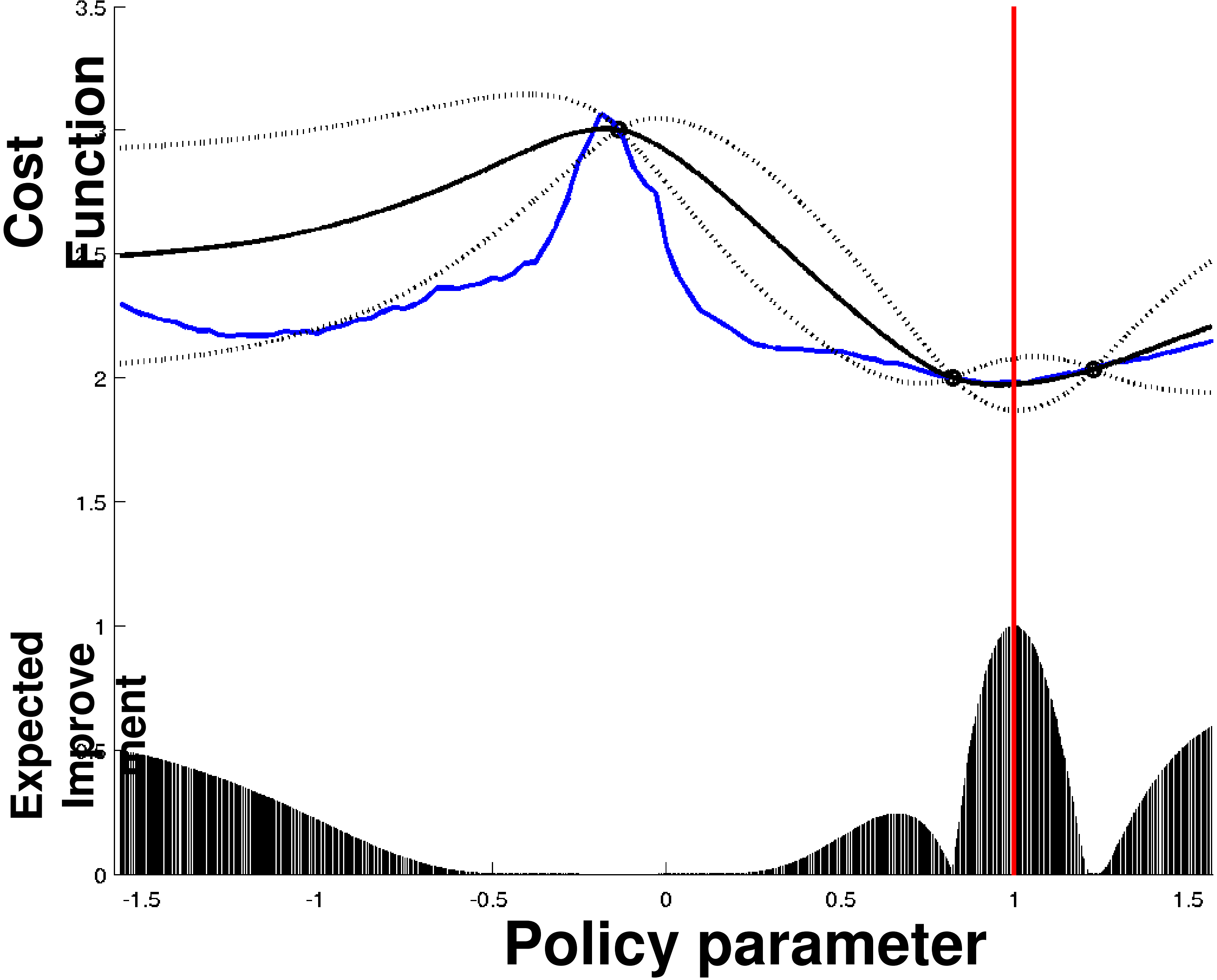
- Reduce variance of the Monte Carlo estimate of the cost function. e.g. using common random numbers as in PEGASUS

[Ng and Jordan 2000]

Active Policy Learning Component

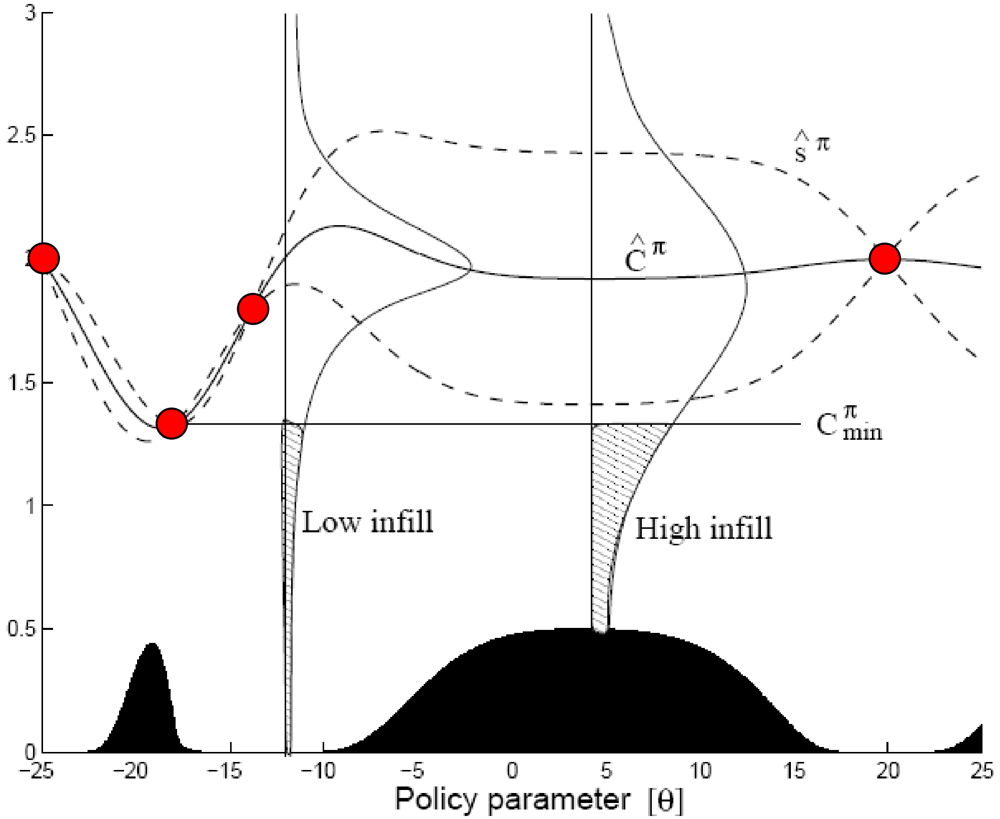


The Expected Improvement Function: Trades-off exploration and exploitation



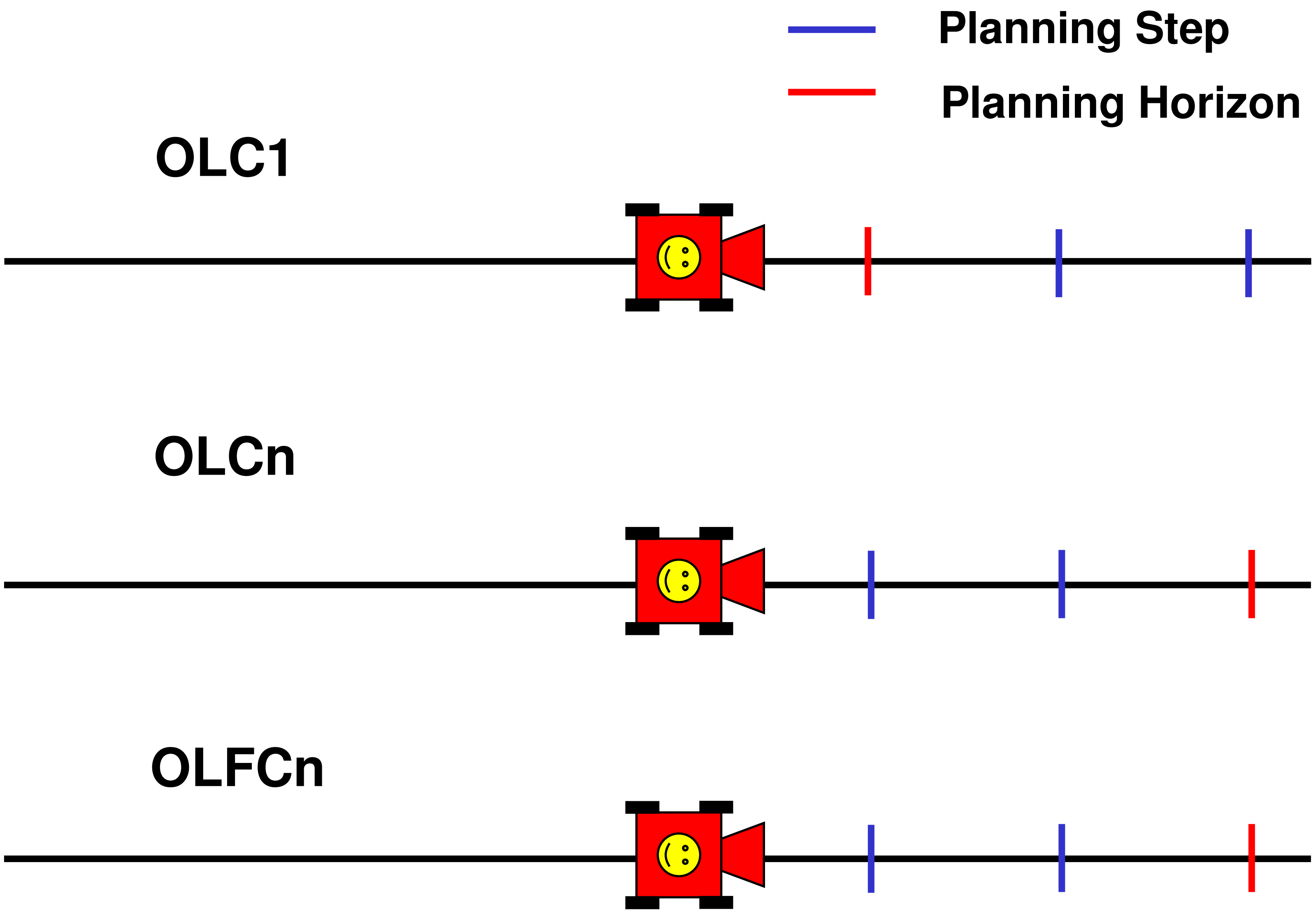
The Expected Improvement Function: Trades-off exploration and exploitation

- The cost function is approximated as a **Gaussian process**
- New parameters are selected using an **expected improvement** function.



$$EI(\boldsymbol{\theta}) = \begin{cases} (C^{\pi}_{\text{min}} - \widehat{C}^{\pi}(\boldsymbol{\theta}))\Phi(d) + \widehat{s}(\boldsymbol{\theta})\phi(d) & \text{if } \widehat{s} > 0 \\ 0 & \text{if } \widehat{s} = 0 \end{cases} \qquad d = \frac{(C^{\pi}_{\text{min}} - \widehat{C}^{\pi}(\boldsymbol{\theta}))}{\widehat{s}(\boldsymbol{\theta})}$$

Receding Horizon Control Strategies



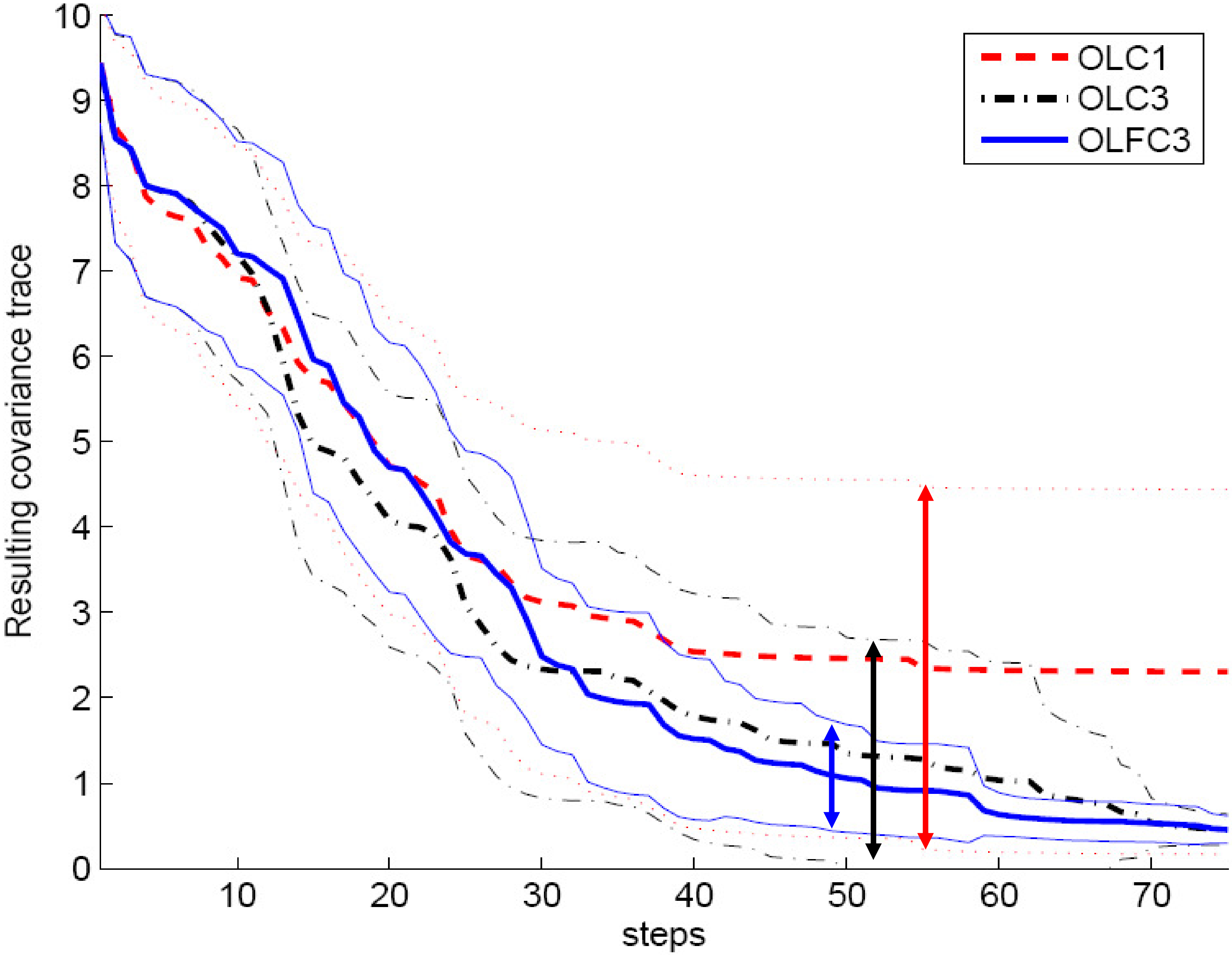
Planning Step

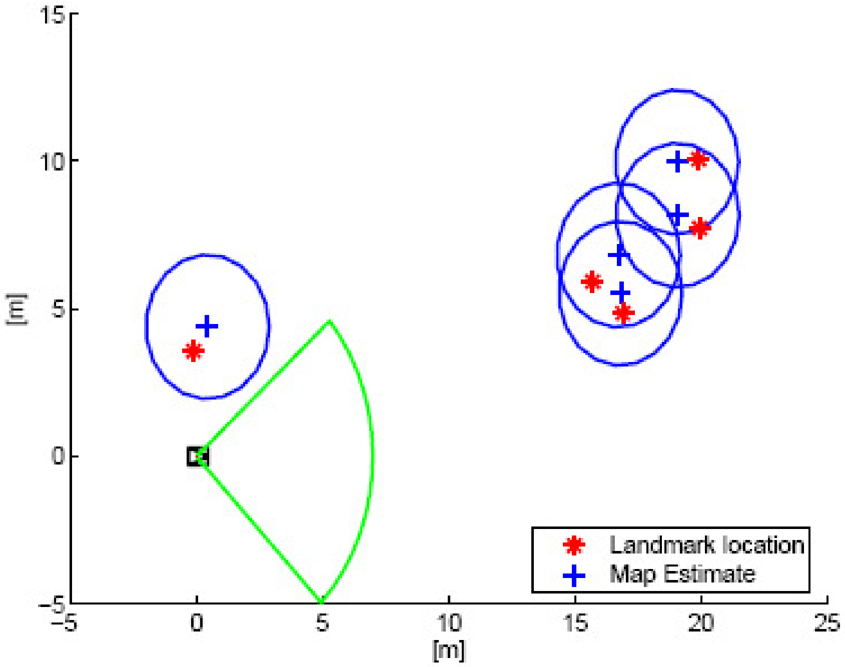
Planning Horizon

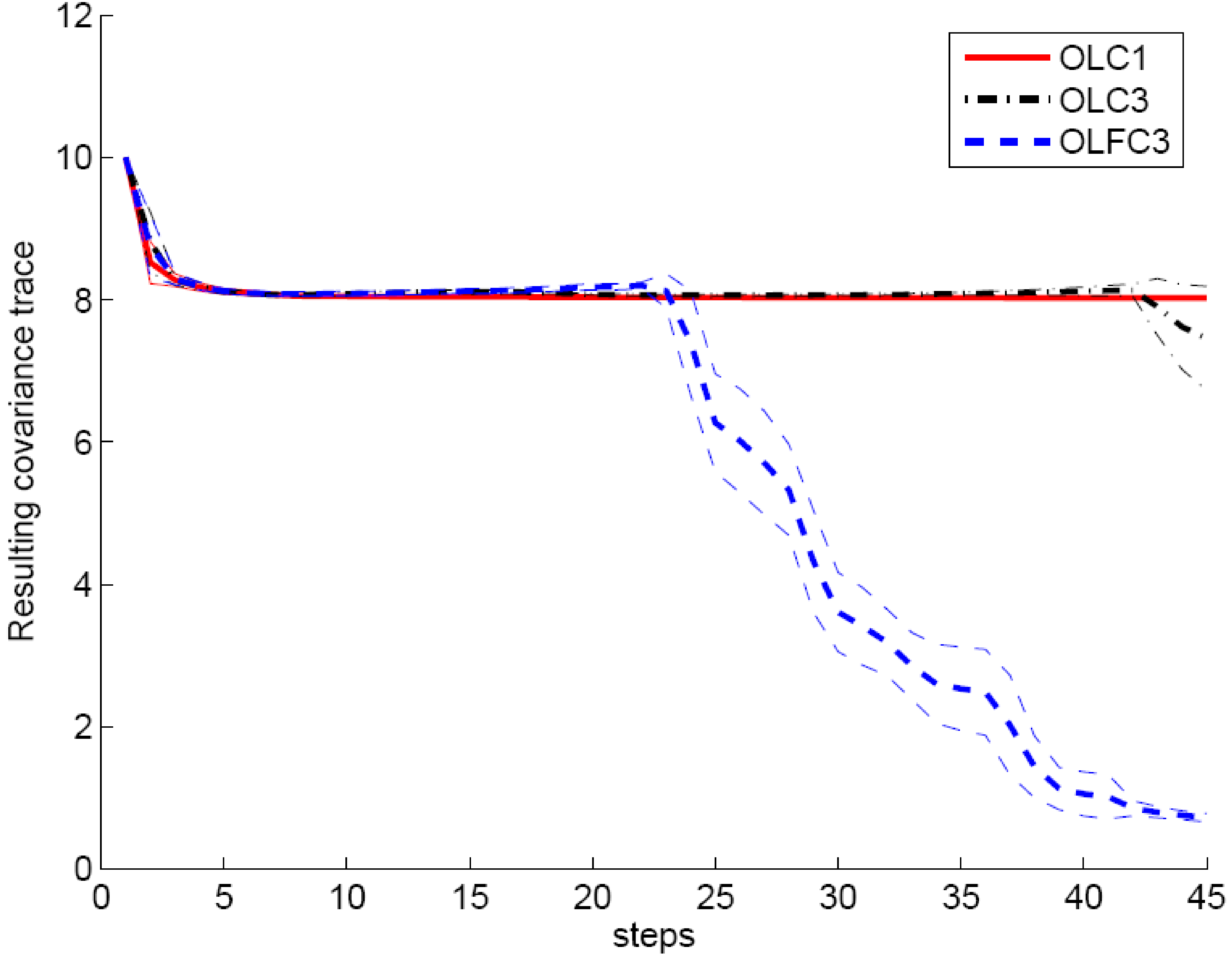
OLC1

OLCn

OLFCn


Resulting covariance trace
steps
0
1
2
3
4
5
6
7
8
9
10
10
20
30
40
50
60
70
OLC1
OLC3
OLFC3


15
10
5
0
−5
[m]
−5
0
5
10
15
20
25
[m]
Landmark location
Map Estimate

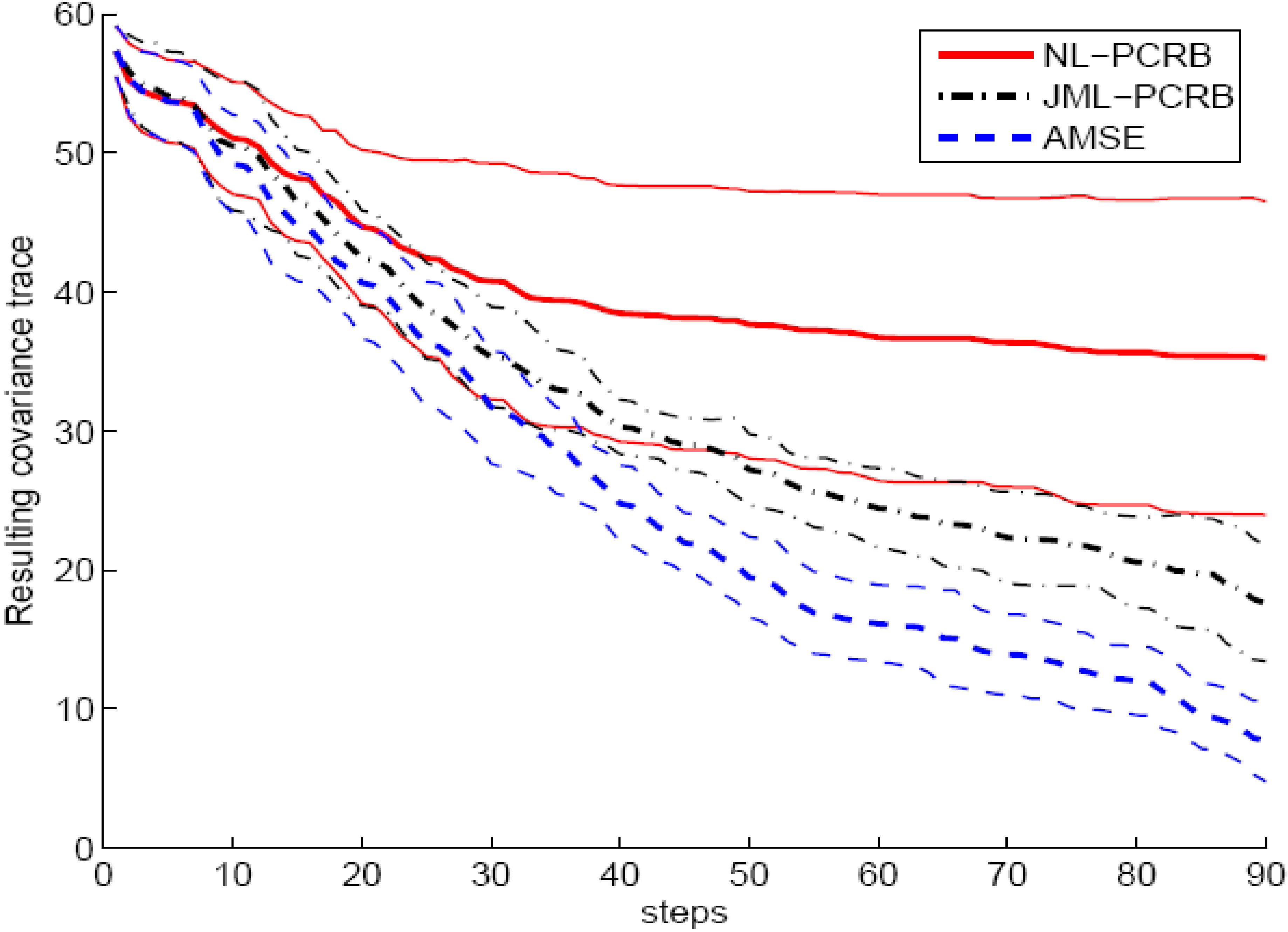

Resulting covariance trace
steps
OLC1
OLC3
OLFC3
0
2
4
6
8
10
12
0
5
10
15
20
25
30
35
40
45

A cheaper cost function approximation

- PCRB approach

$$C^{\pi}_{AMSE} \geq C^{\pi}_{PCRB}$$

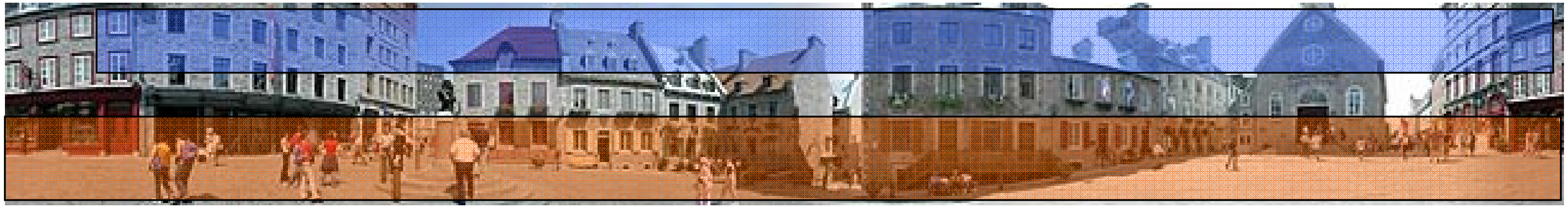
  - PCRB allow us to compute a bound for the cost function with one single Ricatti-like recursion for all the samples.
  - Therefore, the computational cost can be dramatically reduced (~20-30% of total computational cost for 30 samples).
  - However, the bound can be loose, thus we have to test it empirically.
  - We have tested two different PCRBs [Tichavsky et al. 98] [Bergman et al. 01]


NL−PCRB
JML−PCRB
AMSE
Resulting covariance trace
steps
0
10
20
30
40
50
60
0
10
20
30
40
50
60
70
80
90

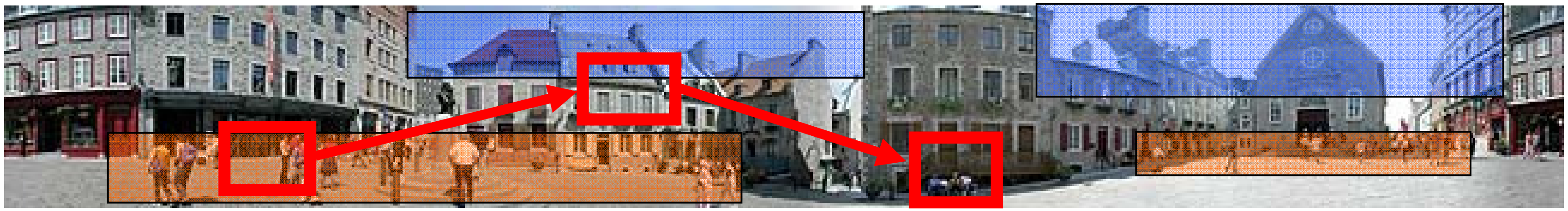
Planning in the visual field

Target-directed attention

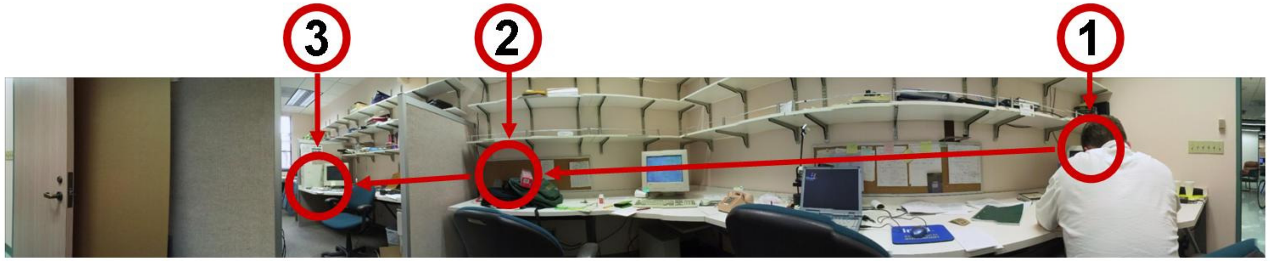
Assume, e.g. you have prior over where to find people and sky



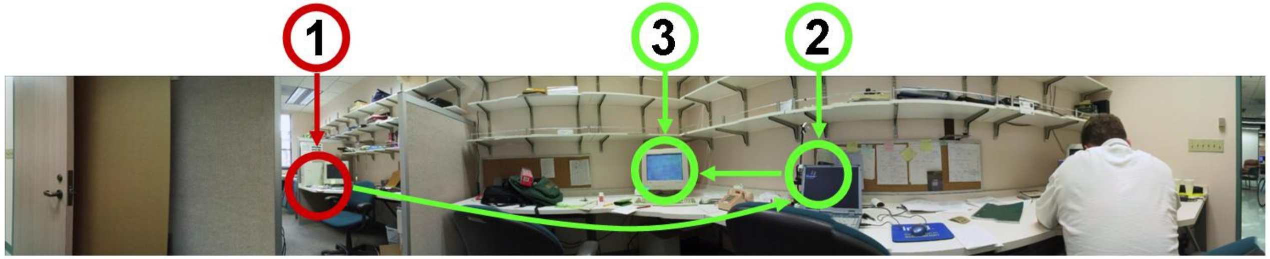
Use Bayesian theory and maximum expected utility principle to combine this prior with bottom-up saliency maps and object likelihood models to obtain a more informative posterior.



Qualitative results

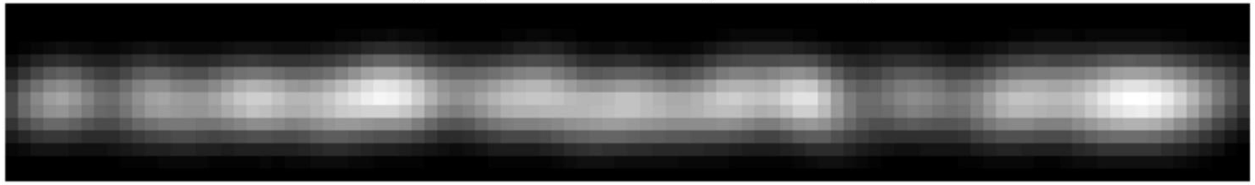


Gaze sequence when using only bottom-up saliency

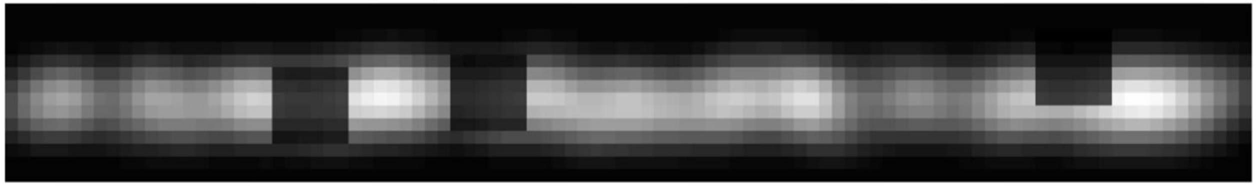


Gaze sequence after gaze planning with bottom-up and top-down information

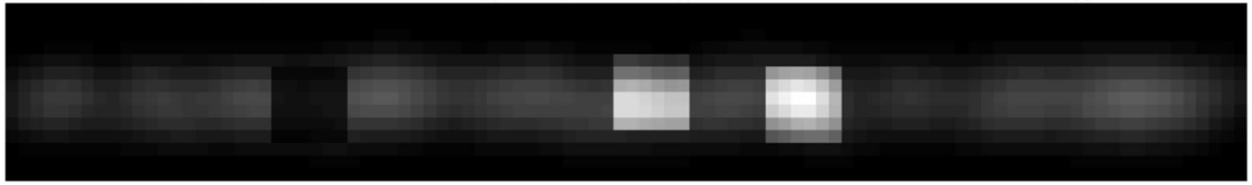
Object prior before any reasoning



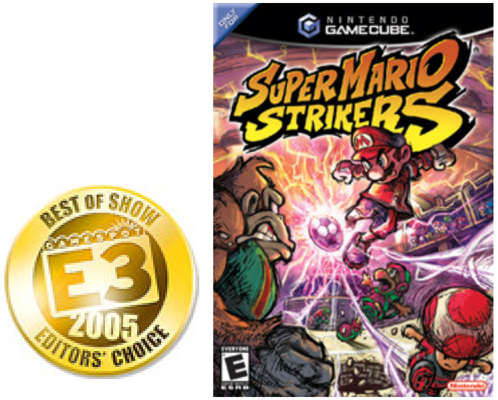
Object posterior after BU reasoning only

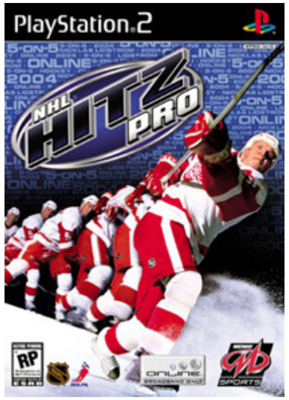


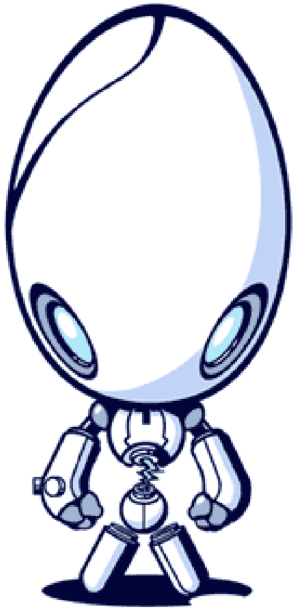
Object posterior after gaze planning with BU and TD reasoning



Planning with Next Level Games / Mike Cora







Motivation

- There exist stable and mature middleware solutions for:
  - graphics, animation, physics, sound, online

- AI and gameplay middleware is the remaining wild-west frontier, and should:
  - Automate parameter tuning
  - Automate quality analysis
  - Simplify the process of creating smart and adaptive agents
  - Generalize to other applications and reduce the amount of game-specific re-implementation

- In the long term, it should reduce the cost and improve the quality of game AI.

- Simulated worlds also have real-world applicability for various strategy analysis and planning domains.

Project desiderata

- Investigate learning algorithms that reduce the need for manual parameter tuning.
- Algorithms should allow for easy integration with game AI development.
- It should be easy for non-programmers to design and tune behaviors, and enable learning where applicable. Tuning for fun is much different than tuning for optimality.
- Algorithms should learn quickly, since there are tight computational and memory constraints on consoles.

Experimental setup

- Taxi Domain:
  - Learn map navigation to pickup and dropoff passengers.
  - Well known benchmark application for *Hierarchical Reinforcement Learning*.

- Expanded on existing algorithms with:
  - Low-level active policy optimizer to learn control of continuous non-linear vehicle dynamics.
  - Mid-level active path learning for navigating a topological map.
  - High-level model-based learning for deciding when to navigate, park, pickup and dropoff passengers.

- Applied to The Open Racing Car Simulator.

Hierarchical Policies

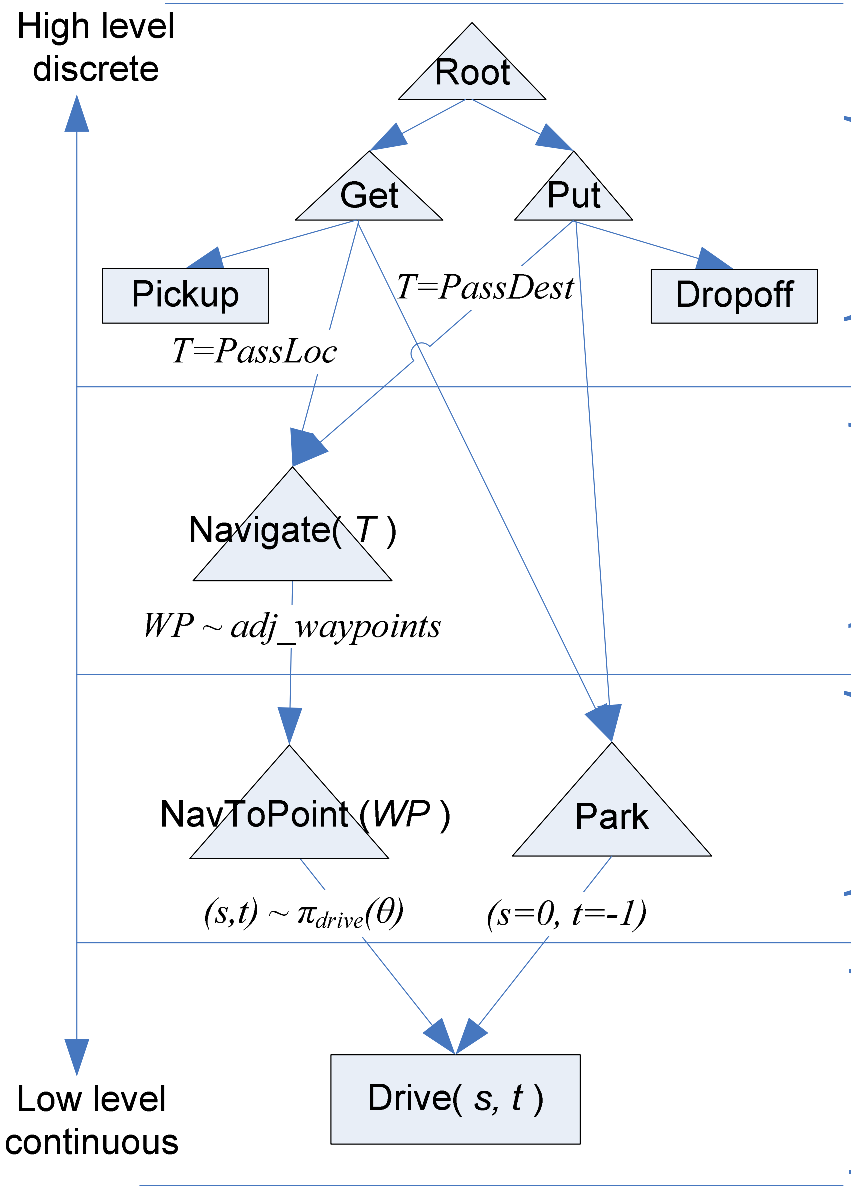
- Break the problem into sub-problems: actions/tasks decomposed into subtasks that
  - extend over time (temporal abstraction)
  - depend on subsets of the state (state abstraction)
  - are easy for non-programmers to create

- Each task is a separate decision process, with termination criteria and reward functions.

- Enables different algorithms and policy representations at different tasks.

- Follows the most common pattern of game AI development: hierarchical state machines.

Hybrid Task Hierarchy



Sparse and discrete states and actions, learn *Q(s,a)* table with:

- RAR
- MAXQ
- MB-MAXQ

Discrete topological map, with continuous waypoint coordinates. The actions are drawn from adjacent waypoints:

- Learn *Q(s,a)* with RAR, MB-MAXQ.
- Learn $V_{TM}(\theta)$ with episodic updates and active EGO exploration.

Continuous state and action spaces:

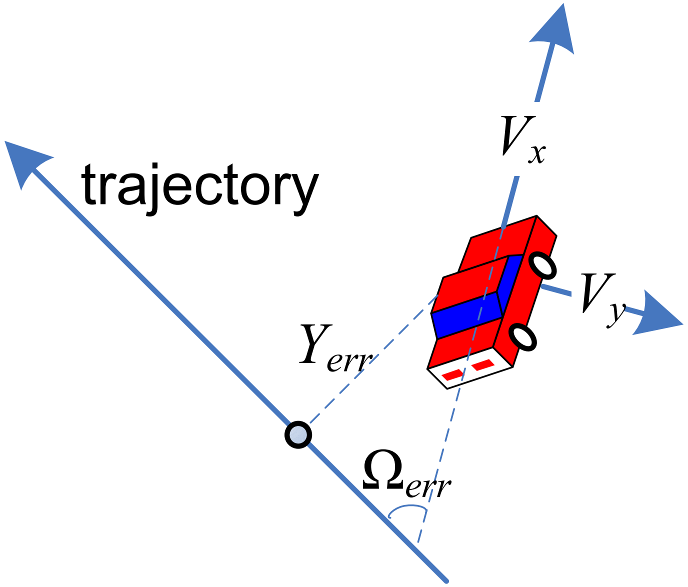
- Active policy search using EGO to optimize the 15 parameters of a non-linear function.
- Hardcoded Park task.

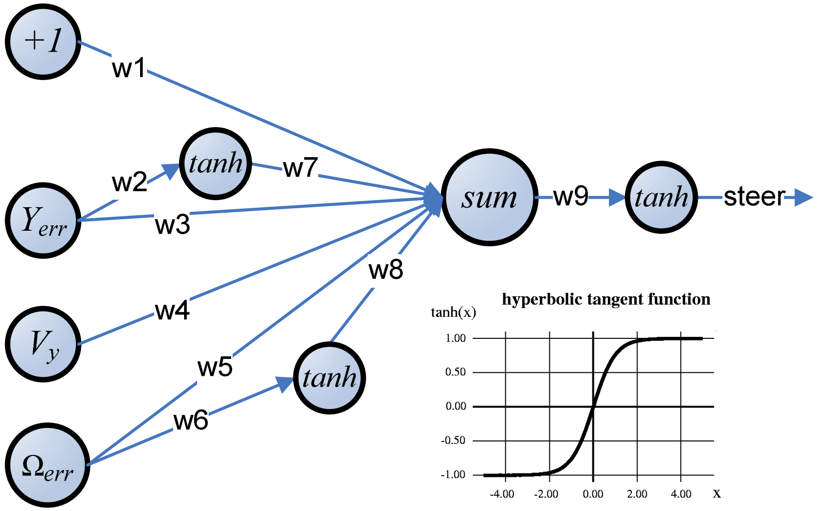
Continuous steering and throttle values between [-1,1] are passed to the simulator:

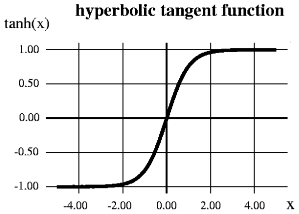
- Simple 2D vehicle model in Matlab
- Complex 3D non-linear vehicle model using the TORCS simulator.

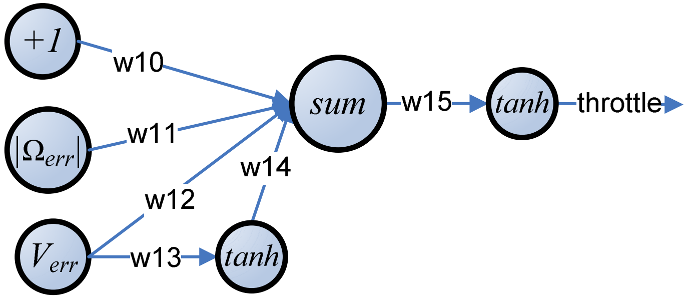
Low-Level: Trajectory following

TORCS: 3D game engine that implements complex vehicle dynamics complete with manual and automatic transmission, engine, clutch, tire, and suspension models.









Active Path Finding in Middle Level

- Mid-level *Navigate* policy generates sequence of waypoints on a topological map to navigate from a location to a destination.

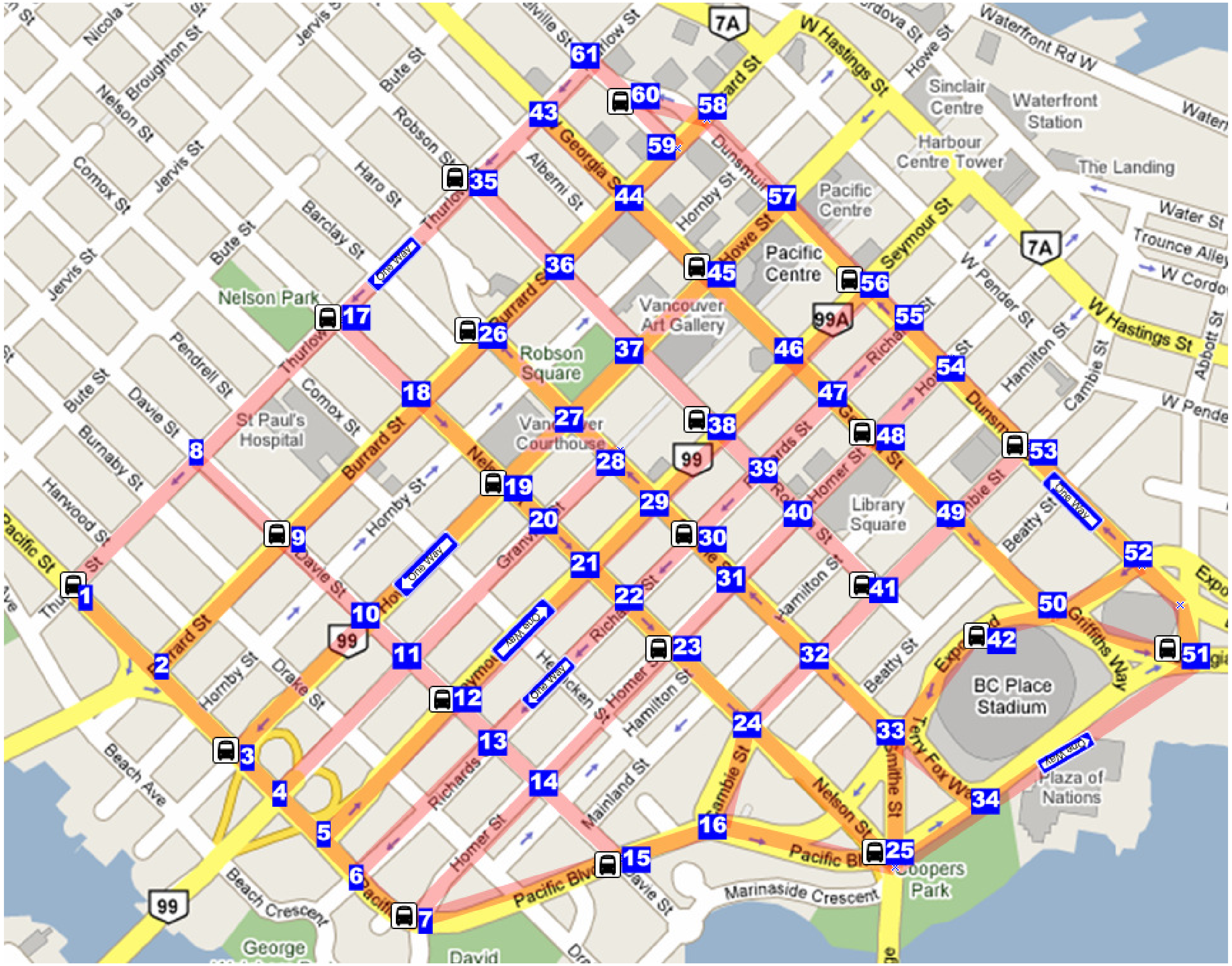
- *V(✪)* value function represents the path length from the current node, to the target.

- Parameterized by the current taxi location, and the target intersection coordinates: ✪$=\{X_C, Y_C, X_T, Y_T\}$

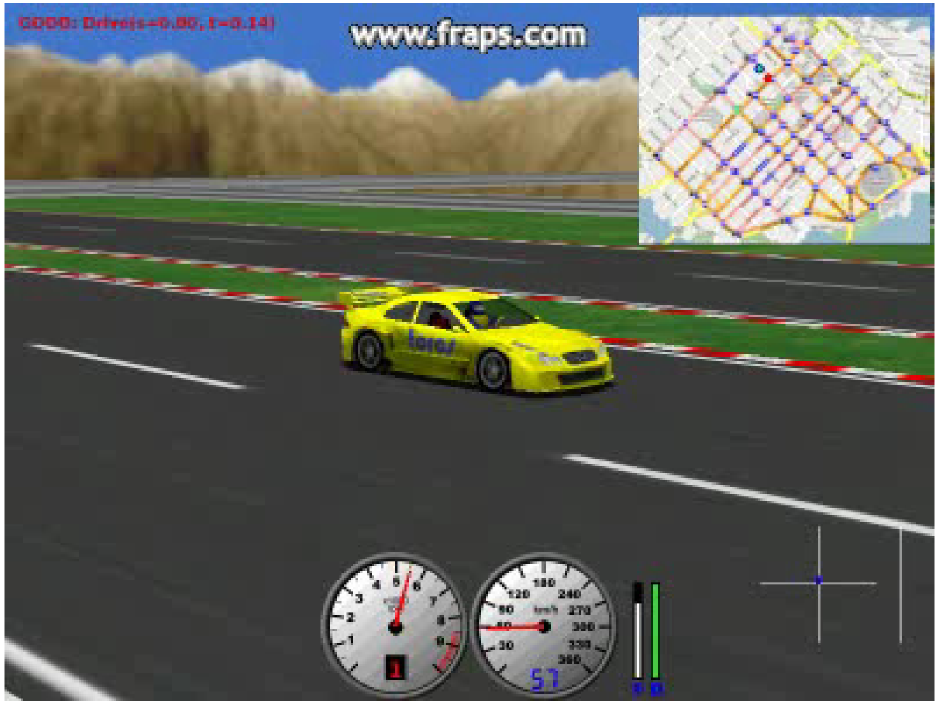
- Adjacent waypoints that maximize the expected *V(✪)*, and have the highest uncertainty (unexplored) are sampled from the Gaussian Process model $GP_{V(\text{✪})}$

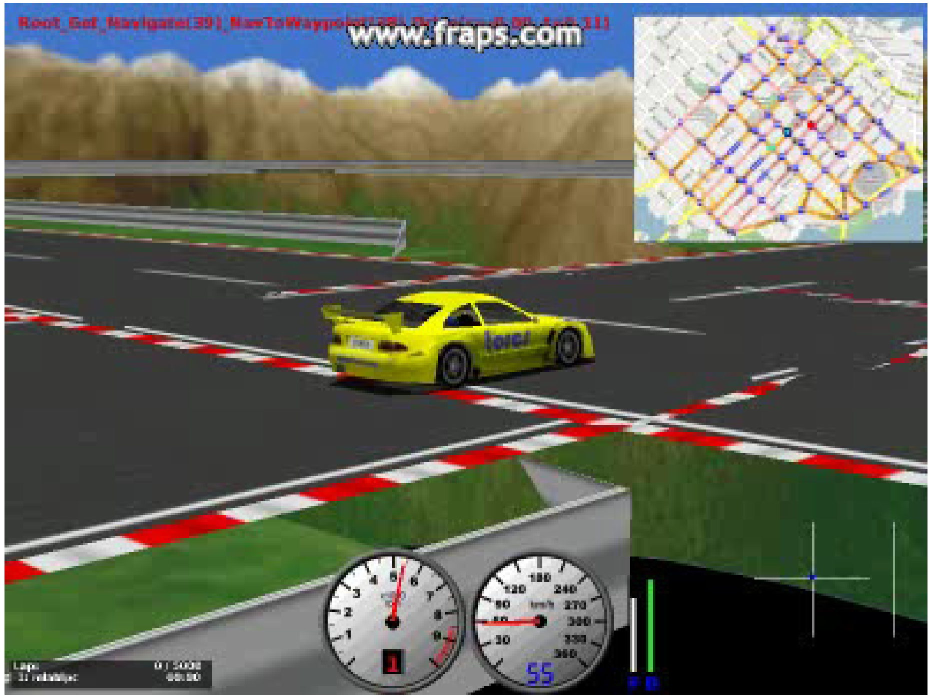
- After every episode, every state along the path is updated with new value estimates.

Mid-level: Topological map



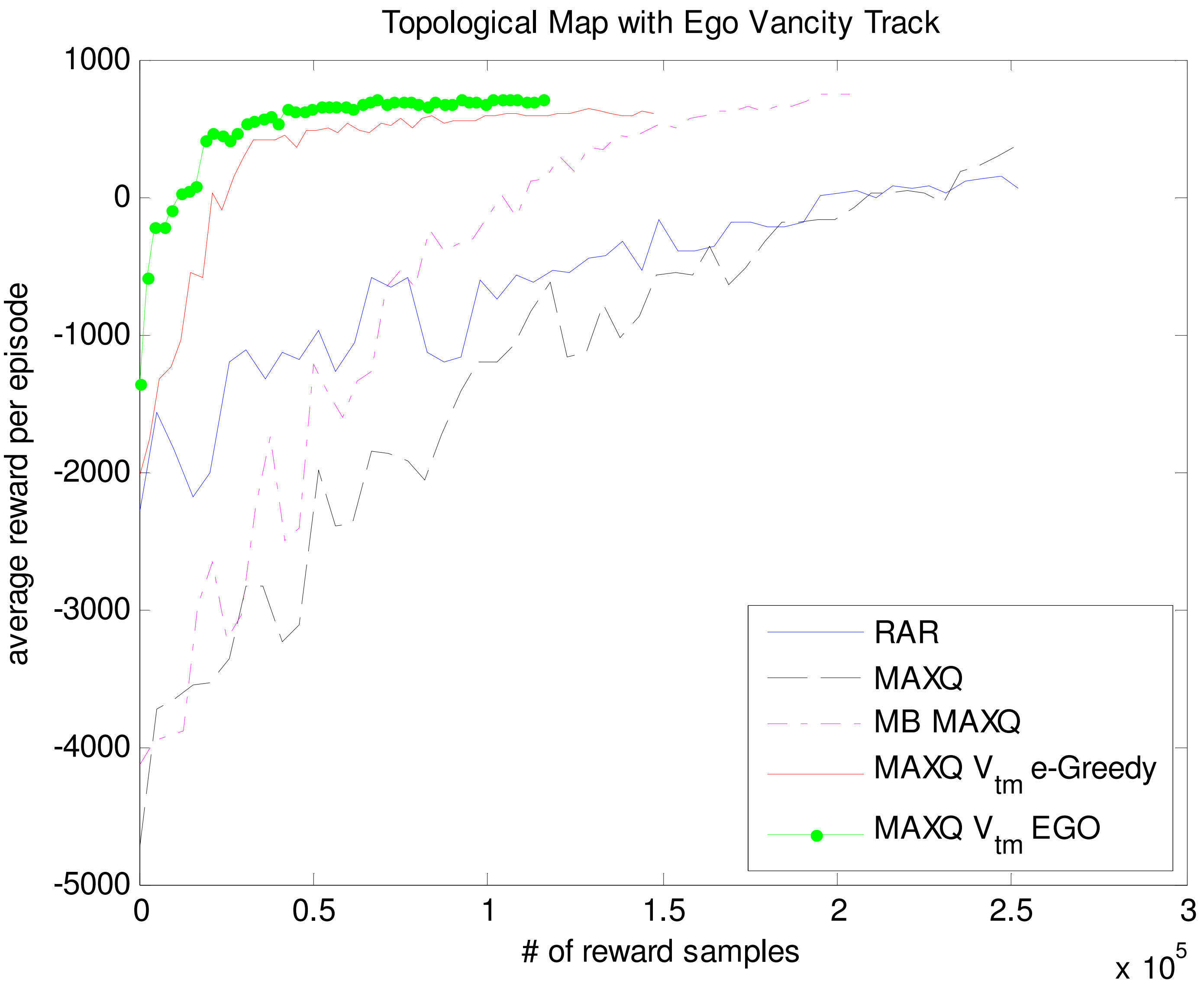
Hierarchical systems apply to many robot tasks – key to build large systems





We used TORCS: A 3D game engine that implements complex vehicle dynamics complete with manual and automatic transmission, engine, clutch, tire, and suspension models.

Mike’s results



Beyond simple Monte Carlo for POMDPs

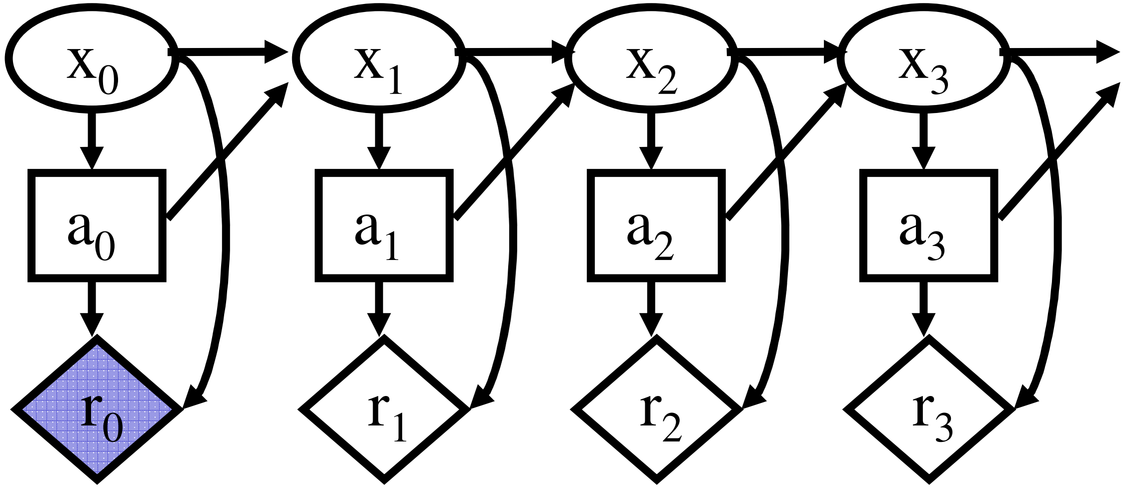
- Realistic POMDPs are very hard to solve as they are often nonlinear, non-Gaussian, high-dimensional, hybrid (discrete and continuous variables) and non-differentiable.

- Fortunately, a few researchers, including Dayan & Hinton (1997), Attias (2003), Toussaint & Storkey (2006), Verma and Rao (2006), and Peters and Schaal (2006) have been able to **map these probabilistic decision problems to inference and learning tasks**.

- These new mappings enable us to develop a vast set of machine learning tools (algorithms that exploit structure and relations, EM, MCMC, GPs, variational approximations, …) for solving POMDPs.

The decision making model: MDP

Initial state distribution:	$\mu(x_0)$
Transition model:	$f(x_n \mid x_{n-1}, a_{n-1})$
Reward model:	$g(r_n \mid x_n, a_n)$
Parameterized policy:	$\pi_\theta(a_n \mid x_n)$



Objective:

$$J_\mu^\pi(\theta) = \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma^n r_n\right]$$

Can also easily parameterize the reward and transition models

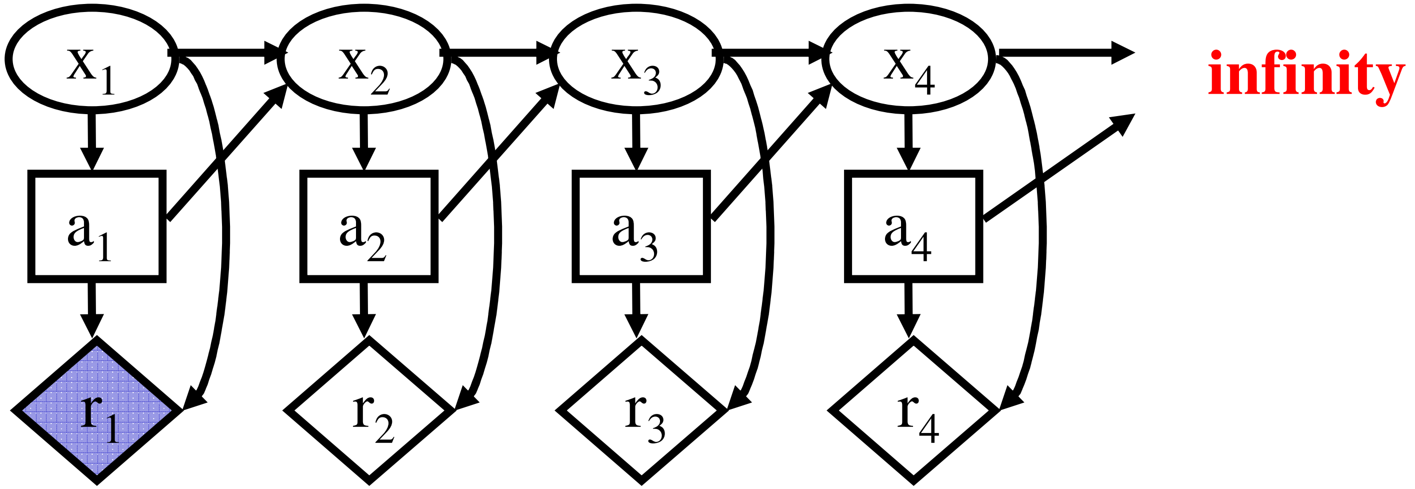
The key infinite mixture insight

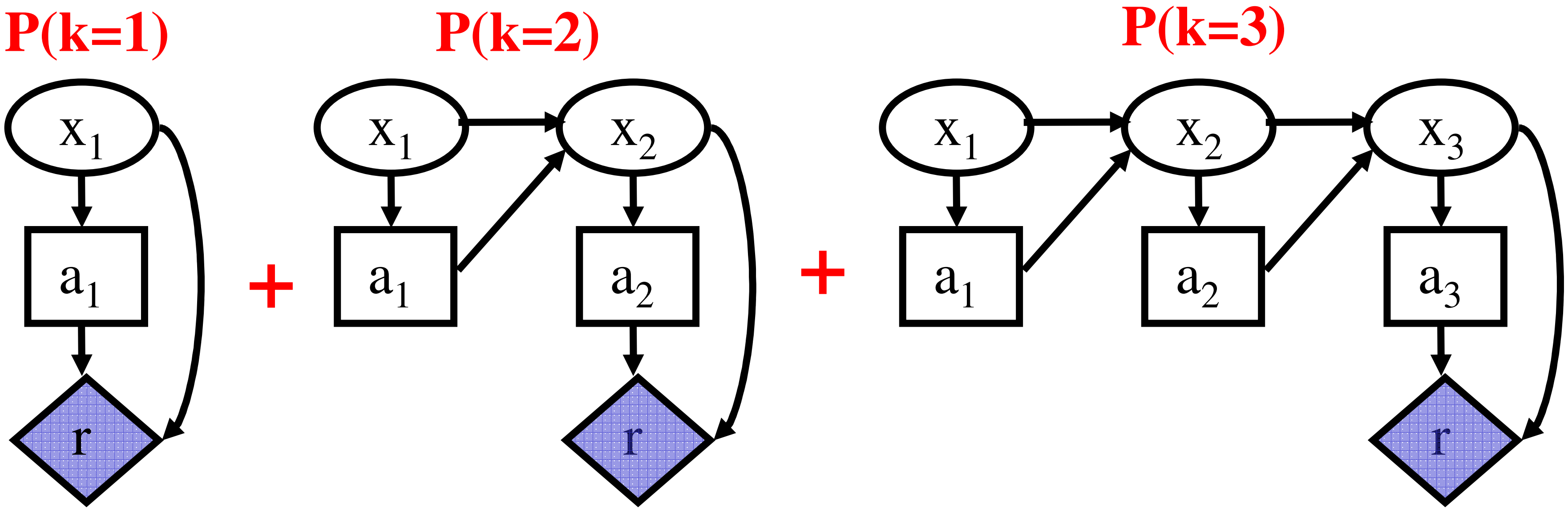
$$
\begin{aligned}
J_{\mu}^{\pi}(\theta) &= \mathbb{E}_{\mu(x_0)\pi_\theta(a_0|x_0)g(r_0|x_0,a_0)\prod_{n=1}^{\infty} g(r_n|x_n,a_n)\pi_\theta(a_n|x_n)f(x_n|x_{n-1},a_{n-1})}\left[\sum_{n=0}^{\infty}\gamma^n r_n\right] \\
&= \sum_{a_0}\sum_{x_0}\sum_{r_0}\gamma^0 r_0 g(r_0|x_0)\mu(x_0)\pi_\theta(a_0|x_0) \\
&\quad +\sum_{a_{0:1}}\sum_{x_{0:1}}\sum_{r_1}\gamma^1 r_1 g(r_1|x_1,a_1)\mu(x_0)\pi_\theta(a_0|x_0)\pi_\theta(a_1|x_1)f(x_1|x_1,a_1)+\cdots
\end{aligned}
$$

$$
= (1-\gamma)\sum_{k=0}^{\infty}\sum_{a_{1:k}}\sum_{x_{1:k}}\sum_{r_k}\underbrace{(1-\gamma)^{-1}\gamma^k}_{\text{prior } p(k)} r_k \underbrace{g(r_k|x_k,a_k)\mu(x_0)\pi_\theta(a_0|x_0)\prod_{n=1}^{k}\pi_\theta(a_n|x_n)f(x_n|x_{n-1},a_{n-1})}_{\text{prior } p_\theta(x_{0:k},a_{0:k},r_k|k)}
$$

$$
= (1-\gamma)\,\mathbb{E}_{p_\theta}\left[r_k\right]
$$

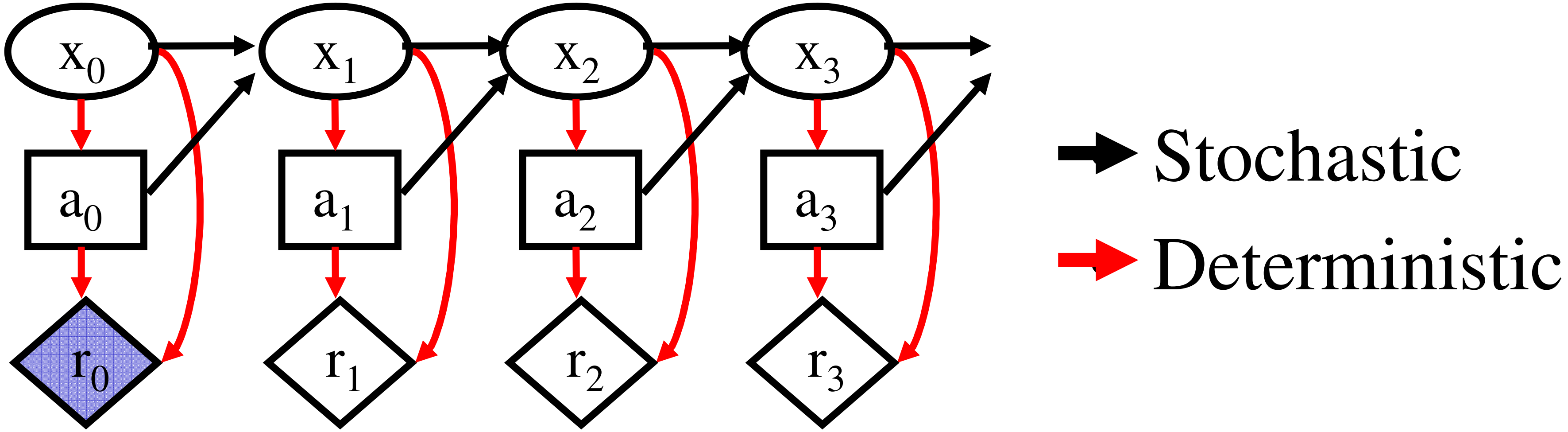
The key infinite mixture insight





The key infinite mixture insight: Deterministic policies and rewards

$$J_\mu^\pi(\theta) = (1-\gamma)\sum_{k=0}^{\infty}\sum_{x_{1:k}} \underbrace{(1-\gamma)^{-1}\gamma^k}_{\text{prior } p(k)} r_k \underbrace{\mu(x_0)\prod_{n=1}^{k} f_\theta(x_n|x_{n-1},a_{n-1})}_{\text{prior } p_\theta(x_{0:k}|k)}$$



The key infinite mixture insight: appears implicitly in some RL works

(Wang, Bowling & Schuurmans 2007, Lagoudakis &Parr 2003)

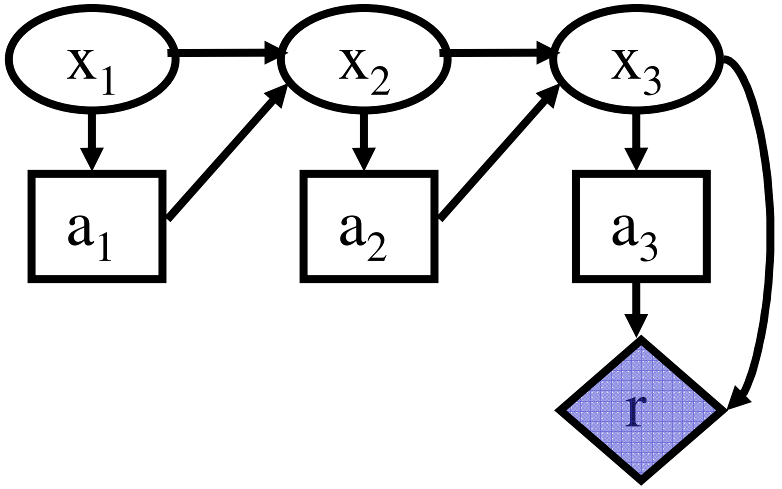
$$\begin{aligned} V &= \sum_{k=0}^{\infty} \gamma^k \left(\Pi P\right)^k \Pi r \\ V &= \Pi r + \gamma \Pi P V \end{aligned}$$

$$J_\mu^\pi = \mu^t V = (1-\gamma) \sum_{k=0}^{\infty} (1-\gamma)^{-1} \gamma^k \mu^t \left(\Pi P\right)^k \Pi r$$

$$V = \left(1 - \gamma \Pi P\right)^{-1} \Pi r$$

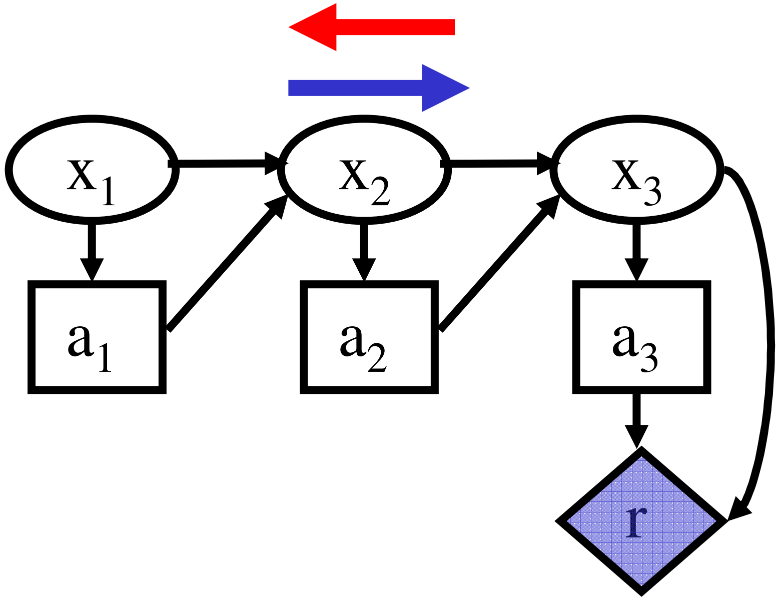
Doina Precup

The EM approach



$$\overset{\textbf{MDP posterior}}{\widetilde{p}_\theta\left(x_{0:k}, a_{0:k} \middle| k, r_k\right)} = \frac{\overset{\textbf{Likelihood}}{\boxed{r\left(x_k, a_k\right)}}\,\overset{\textbf{Prior}}{\boxed{p_\theta\left(x_{0:k}, a_{0:k} \middle| k\right)}}}{\underset{\textbf{Marginal likelihood}}{\boxed{\widetilde{p}_\theta\left(r_k \middle| k\right)}}}$$

The E step



$$
\begin{aligned}
\widetilde{p}(x_n, a_n | k, r_k) &= \frac{\overbrace{\widetilde{p}(x_n, a_n | k)}^{\boxed{\alpha_k(x_n,a_n)}}\overbrace{\widetilde{p}(r_k | x_n, a_n, k)}^{\boxed{\beta_k(x_n,a_n)}}}{\widetilde{p}(r_k | k)}, \\
\widetilde{p}\,(r_k | k) &= \sum_{a_n} \sum_{x_n} \alpha_k(x_n, a_n) \beta_k(x_n, a_n).
\end{aligned}
$$

The M step

As in standard EM for HMMs, the likelihood (expected reward) is maximized by optimizing the Q function:

$$Q\left(\theta_{\text{old}}, \theta\right) = \mathbb{E}_{\tilde{p}_{\theta_{\text{old}}}(k, x_{0:k}, a_{0:k} | r_k)} \overset{\textbf{Complete data likelihood}}{\boxed{\left[\log\left(r_k\, p_\theta\left(k, x_{0:k}, a_{0:k}\right)\right)\right]}}$$

Assuming a conditional probability table representation for the policy:

$$\pi_\theta(a_n = a | x_n = i) = \theta_{ia}$$

$$\approx \quad \frac{\sum_{k=0}^{k_{\max}} p\left(k\right) \sum_{n=1}^{k} \alpha_k(x_n = i, a_n = a) \beta_k(x_n = i, a_n = a)}{\sum_{k=0}^{k_{\max}} p\left(k\right) \sum_{n=1}^{k} \sum_a \alpha_k(x_n = i, a_n = a) \beta_k(x_n = i, a_n = a)}$$

But one can use many other representations.

Improvements (Toussaint *et al*)

- Alpha and beta messages are computed only once for all MDPs. Define betas in terms of time-to-go.
- We can optimize the Q function easily for a large class of policies.
- The technique also applies to linear Gaussian MDPs.
- It can be easily extended to POMDPs
- For structured models, one can apply the junction tree algorithm.

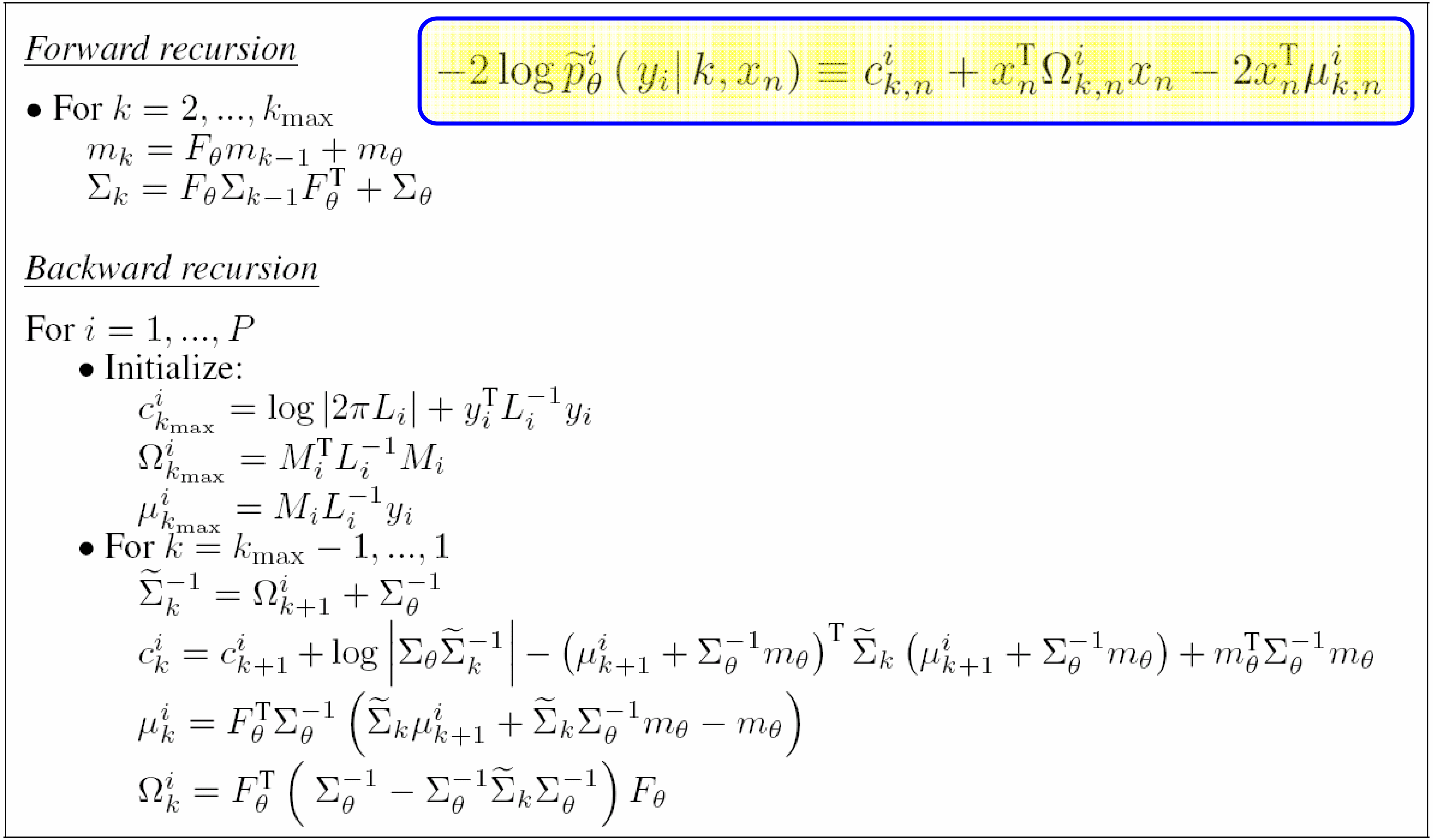
Mixture reward functions: Relaxing the Q in LQG optimal control

$$\begin{aligned}\mu\left(x_1\right) &= \mathcal{N}\left(x_1;\mu_1,\Sigma_1\right)\\ f_\theta\left(x_n\middle|x_{n-1}\right) &= \mathcal{N}\left(x_n;F_\theta x_{n-1}+m_\theta,\Sigma_\theta\right)\end{aligned}$$

$$r\left(x\right)=\sum_{i=1}^{P}\pi_i\mathcal{N}\left(y_i;M_i x,L_i\right)$$

That is, although the dynamics are linear-Gaussian, the reward can be an arbitrary positive function. This has nice applications in robotics (Jan Peters).

Mixture reward functions: E step



$$-2\log \widetilde{p}_\theta^i\left(y_i \middle| k, x_n\right) \equiv c_{k,n}^i + x_n^{\mathsf{T}}\Omega_{k,n}^i x_n - 2x_n^{\mathsf{T}}\mu_{k,n}^i$$

Forward recursion

- For $k = 2, \ldots, k_{\max}$

$$m_k = F_\theta m_{k-1} + m_\theta$$
$$\Sigma_k = F_\theta \Sigma_{k-1} F_\theta^{\mathsf{T}} + \Sigma_\theta$$

Backward recursion

For $i = 1, \ldots, P$

- Initialize:

$$c_{k_{\max}}^i = \log\left|2\pi L_i\right| + y_i^{\mathsf{T}} L_i^{-1} y_i$$
$$\Omega_{k_{\max}}^i = M_i^{\mathsf{T}} L_i^{-1} M_i$$
$$\mu_{k_{\max}}^i = M_i L_i^{-1} y_i$$

- For $k = k_{\max} - 1, \ldots, 1$

$$\widetilde{\Sigma}_k^{-1} = \Omega_{k+1}^i + \Sigma_\theta^{-1}$$
$$c_k^i = c_{k+1}^i + \log\left|\Sigma_\theta \widetilde{\Sigma}_k^{-1}\right| - \left(\mu_{k+1}^i + \Sigma_\theta^{-1} m_\theta\right)^{\mathsf{T}} \widetilde{\Sigma}_k \left(\mu_{k+1}^i + \Sigma_\theta^{-1} m_\theta\right) + m_\theta^{\mathsf{T}} \Sigma_\theta^{-1} m_\theta$$
$$\mu_k^i = F_\theta^{\mathsf{T}} \Sigma_\theta^{-1}\left(\widetilde{\Sigma}_k \mu_{k+1}^i + \widetilde{\Sigma}_k \Sigma_\theta^{-1} m_\theta - m_\theta\right)$$
$$\Omega_k^i = F_\theta^{\mathsf{T}}\left(\Sigma_\theta^{-1} - \Sigma_\theta^{-1} \widetilde{\Sigma}_k \Sigma_\theta^{-1}\right) F_\theta$$

Mixture reward functions: M step

Differentiating the Q function:

$$\sum_{k=1}^{k_{\max}} \widetilde{p}_\theta(k) \sum_{n=1}^{k} \mathbb{E}\left[-\log|\Sigma_\theta| - (x_n - F_\theta x_{n-1} - m_\theta)^T \Sigma_\theta (x_n - F_\theta x_{n-1} - m_\theta)\right]$$

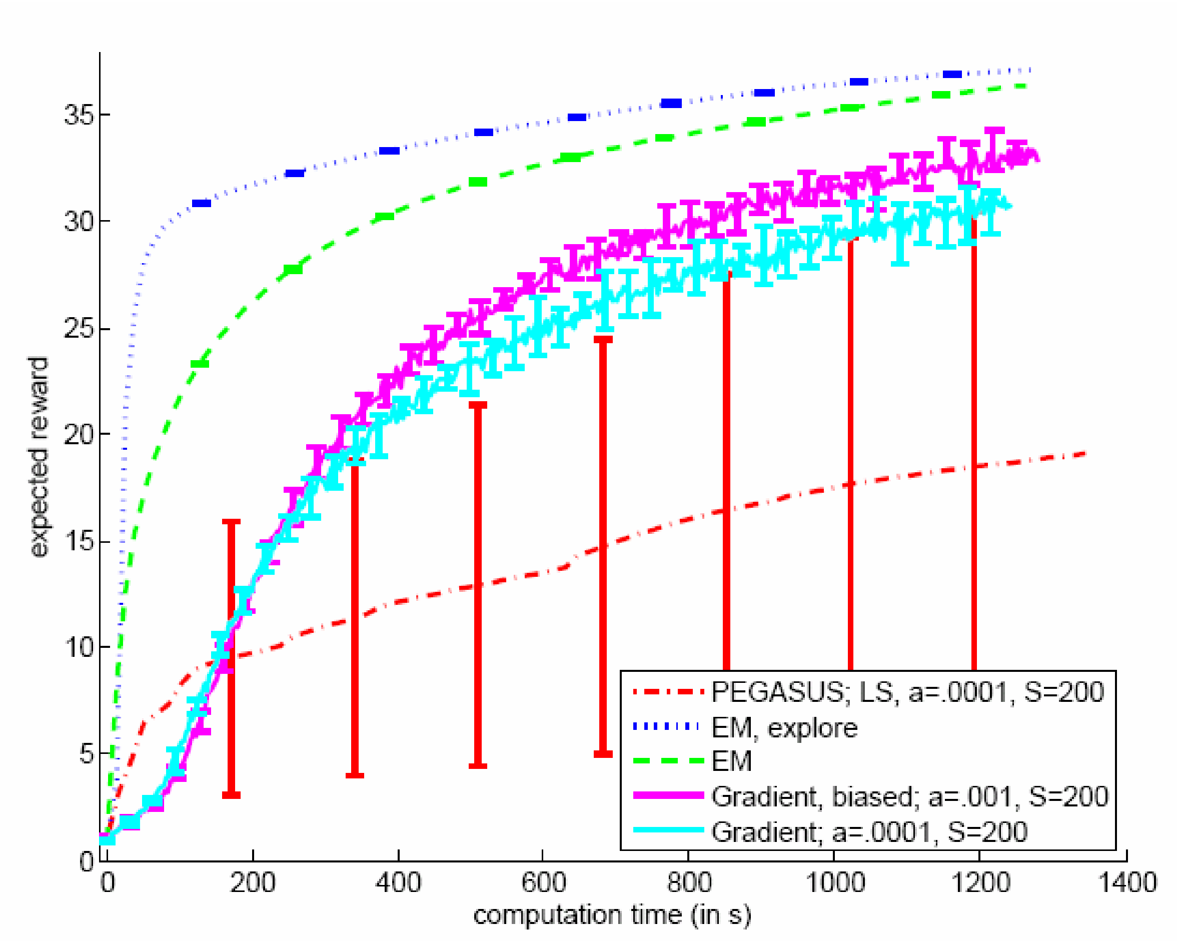
We obtain the following analytical estimates of the parameters:

$$m_i = \left\{\sum_{k=1}^{k_{\max}} k\widetilde{p}_{\theta_{i-1}}(k)\right\}^{-1} \sum_{k=1}^{k_{\max}} \widetilde{p}_{\theta_{i-1}}(k) \sum_{n=1}^{k} \mathbb{E}\left[x_n - F_{i-1}x_{n-1}\right],$$

$$F_i = \left\{\sum_{k=1}^{k_{\max}} \widetilde{p}_{\theta_{i-1}}(k) \sum_{n=1}^{k} \mathbb{E}\left[x_{n-1}x_{n-1}^T\right]\right\}^{-1} \sum_{k=1}^{k_{\max}} \widetilde{p}_{\theta_{i-1}}(k) \sum_{n=1}^{k} \mathbb{E}\left[(x_n - m_i)x_{n-1}^T\right],$$

$$\Sigma_i = \left\{\sum_{k=1}^{k_{\max}} k\widetilde{p}_{\theta_{i-1}}(k)\right\}^{-1} \sum_{k=1}^{k_{\max}} \widetilde{p}_{\theta_{i-1}}(k) \sum_{n=1}^{k} \mathbb{E}\left[(x_n - F_i x_{n-1} - m_i)(x_n - F_i x_{n-1} - m_i)^T\right]$$

Gaussian mixture rewards



The approach is a “policy iteration” version of the value iteration method for continuous state spaces of Porta, Poupart et al. Yet, here, the action models are more general.

The trans-dimensional MCMC approach with infinite mixtures

The MDP distribution is trans-dimensional. We need to sample over the space of models and model dimension in order to characterize it properly. The samples can be used to approximate the Q function.

$$p_\theta\left(k, x_{1:k}, r_k\right) = \left(1-\gamma\right)^{-1} \gamma^k \mu\left(x_1\right) g_\theta\left(r_k \middle| x_k\right) \prod_{n=2}^{k} f_\theta\left(x_n \middle| x_{n-1}\right)$$

We can take this further by specifying a prior over the parameters. Then, in this Bayesian setting, **the target is proportional to the reward**:

$$\overline{p}\left(\theta, k, x_{1:k}\right) \propto r\left(x_k\right) p_\theta\left(k, x_{1:k}\right) p\left(\theta\right)$$

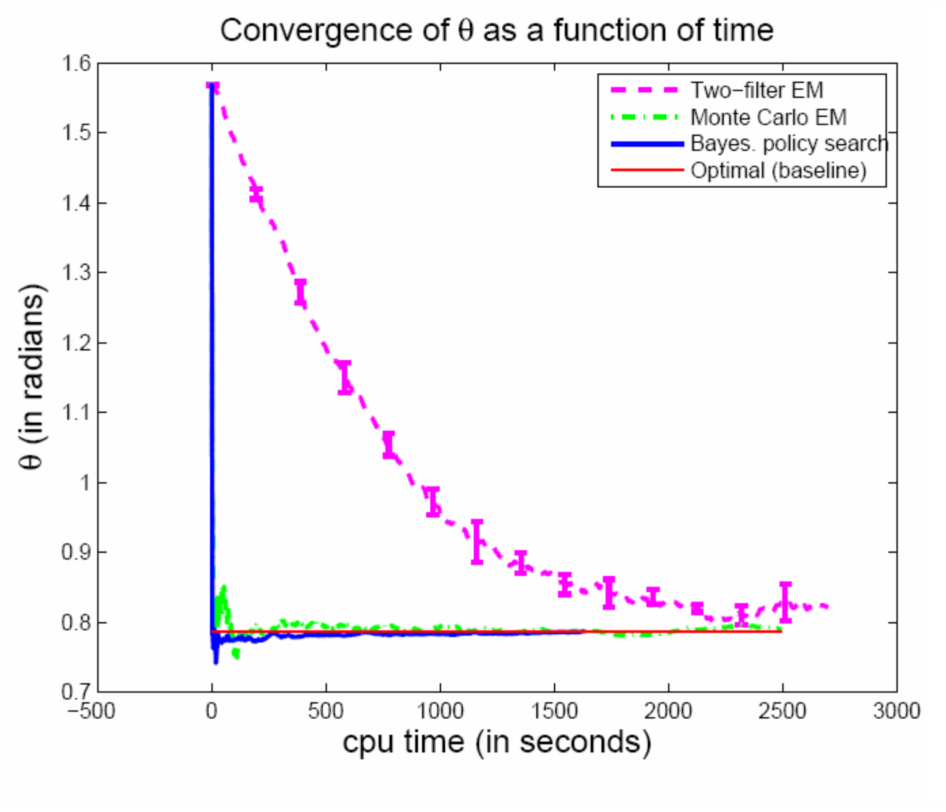
The trans-dimensional MCMC approach with infinite mixtures

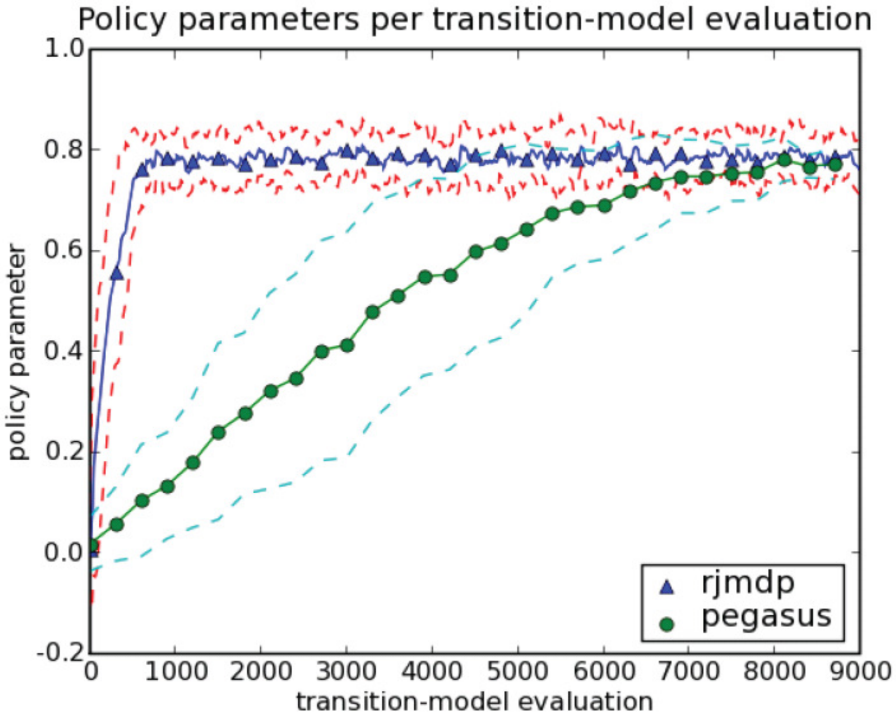
$$\alpha_{birth} = \frac{\widetilde{p}_\theta\left(k+1, x_{1:j-1}, x^*, x_{j+1:k}\right) d_{k+1}}{\widetilde{p}_\theta\left(k, x_{1:k}\right) b_k q_\theta\left(x^* \middle| x_{j-1:j+1}\right)} = \frac{\gamma f_\theta\left(x^* \middle| x_{j-1}\right) f_\theta\left(x_{j+1} \middle| x^*\right) d_{k+1}}{f_\theta\left(x_j \middle| x_{j-1}\right) b_k q_\theta\left(x^* \middle| x_{j-1:j+1}\right)}$$

1. Initialization: set $(k^{(0)}, x^{(0)}_{1:k^{(0)}}, \theta^{(0)})$.
2. For $i = 0$ to $N - 1$
   - Sample $u \sim \mathcal{U}_{[0,1]}$.
   - If $(u \leq b_k)$
     - then carry out a "birth" move: Increase the horizon length of the MDP, say $k^{(i)} = k^{(i-1)} + 1$ and insert a new state.
     - else if $(u \leq b_k + d_k)$ then carry out a "death" move: decrease the horizon length of the MDP, say $k^{(i)} = k^{(i-1)} - 1$ and an existing state.
     - else generate samples $x^{(i)}_{1:k^{(i-1)}}$ of the MDP states.

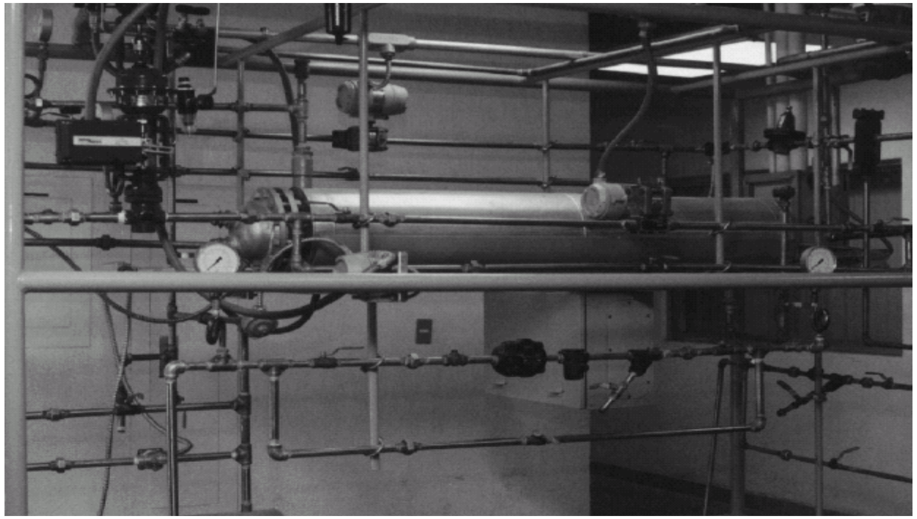
     End If.
   - Sample the policy parameters $\theta^{(i)}$ conditional on the samples $(x^{(i)}_{1:k^{(i)}}, k^{(i)})$.

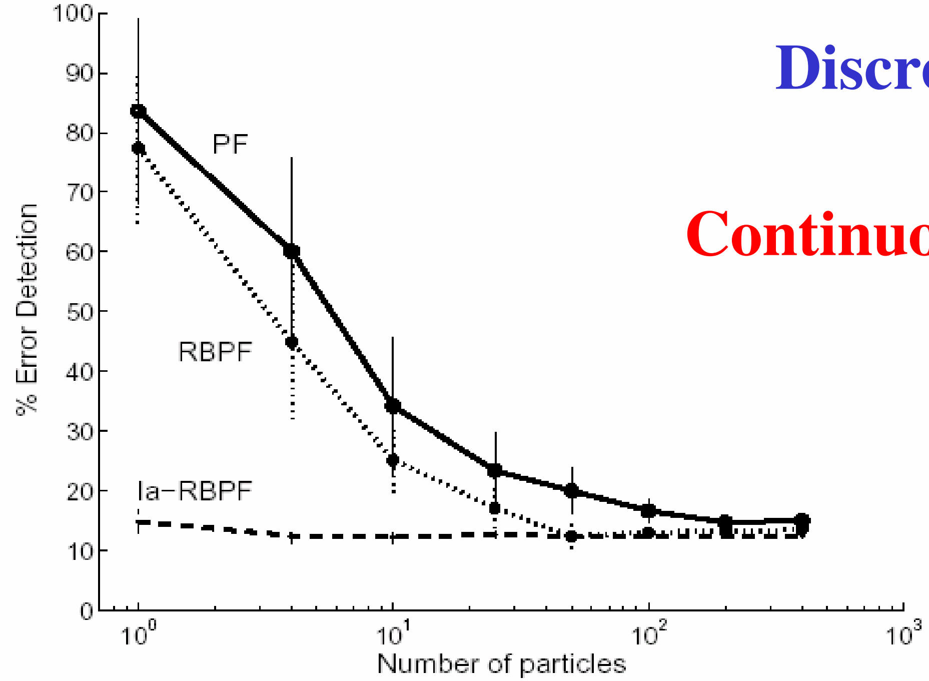
The trans-dimensional MCMC approach with infinite mixtures





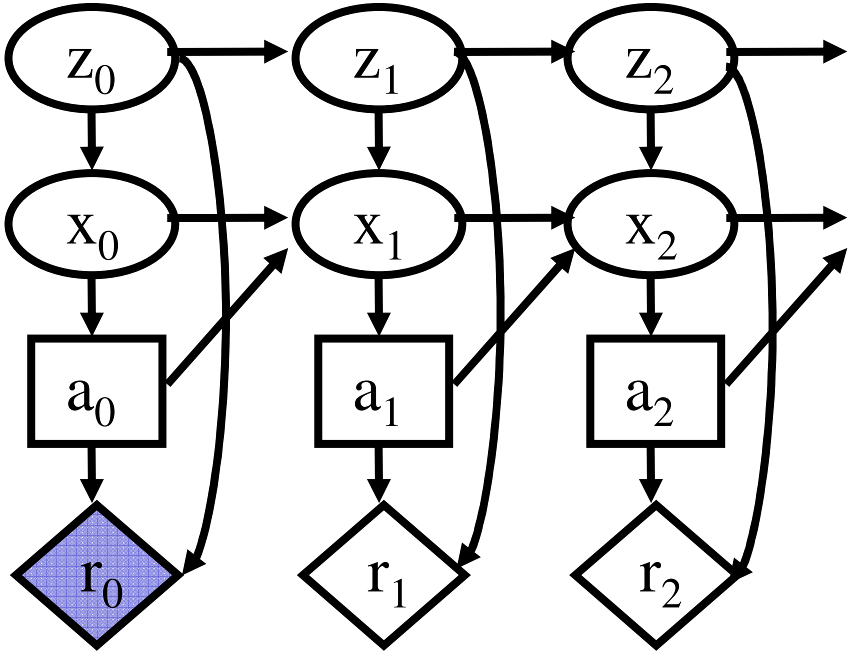
Exploiting structure: hybrid JMLS models





Discrete

Continuous



Many thanks

http://www.cs.ubc.ca/~nando

The EM approach

$$
\begin{aligned}
Q\left(\theta_{\text{old}},\theta\right) &= \sum_{k=0}^{\infty}\sum_{a_{1:k}}\sum_{x_{1:k}} \log[r_k\, p_\theta\left(k,x_{0:k},a_{0:k}\right)]\widetilde{p}_{\theta_{\text{old}}}\left(k,x_{0:k},a_{0:k}|r_k\right) \\
&\equiv \sum_{k=0}^{\infty}\sum_{a_{1:k}}\sum_{x_{1:k}} \left[\sum_{n=0}^{k}\log\pi_\theta(a_n|x_n)\right]\widetilde{p}_{\theta_{\text{old}}}\left(k,x_{0:k},a_{0:k}|r_k\right) \\
&= \sum_{k=0}^{\infty}\sum_{n=1}^{k}\sum_{a_n}\sum_{x_n}\log\pi_\theta(a_n|x_n)\widetilde{p}_{\theta_{\text{old}}}\left(k,x_n,a_n|r_k\right) \\
&= \sum_{n=0}^{\infty}\sum_{a_n}\sum_{x_n}\log\pi_\theta(a_n|x_n)\sum_{k=n}^{\infty}\widetilde{p}_{\theta_{\text{old}}}\left(k,x_n,a_n|r_k\right) \\
&= \sum_{n=0}^{\infty}\sum_{a_n}\sum_{x_n}\log\pi_\theta(a_n|x_n)\sum_{k=n}^{\infty}\widetilde{p}_{\theta_{\text{old}}}\left(x_n,a_n|k,r_k\right)\widetilde{p}_{\theta_{\text{old}}}\left(k|r_k\right)
\end{aligned}
$$