Lesson Four: Monte-Carlo methods

Richard Yi Da Xu

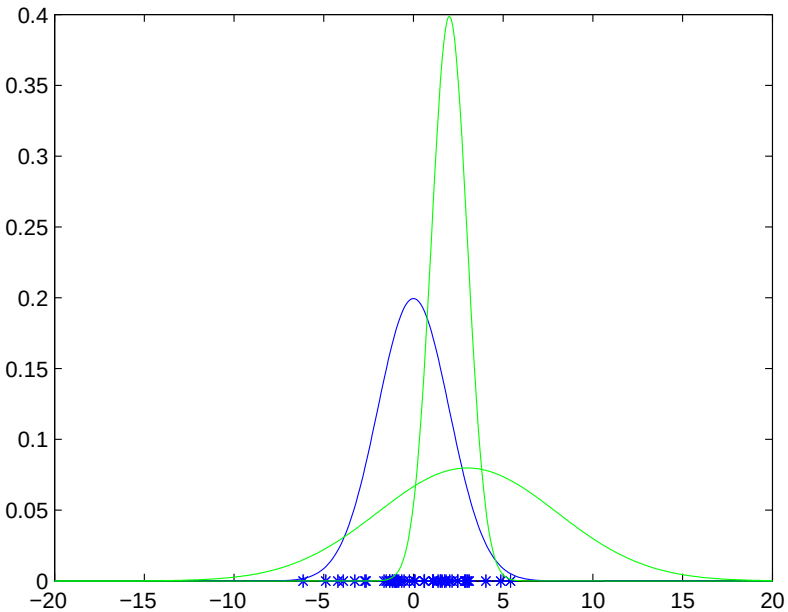
University of Technology, Sydney

May 25, 2015

MLE Example

Normal distributed data

- You believe data is Normal distributed:



Maximum Likelihood Estimation

- which "normal" distribution parameter $\theta = (\mu, \sigma)$ is more likely?

$$\theta^{\text{MLE}} = \arg\max_{\theta} \left(\log[p(X|\theta)]\right)$$

$$= \arg\max_{\theta} \left(\sum_{i=1}^{N} \log[\mathcal{N}(x_i; \mu, \sigma)] \right)$$

- How to solve "argmax"? Well easy, take the deriviative and let it equal zero. Works in the Gaussian case.

MAP Example

- What if I have some prior knowledge of of μ, for example, $\mu \sim \mathcal{N}(\mu_0, \sigma_0)$. This type of estimation is called Maximum a Posterior (MAP):

$$\theta^{\text{MAP}} = \arg\max_{\theta} \left(\log[p(X|\theta)p(\theta)]\right)$$

Say what you need is to find the mean, i.e.,

$$\mu^{\text{MAP}} = \arg\max_{\mu} \left(\sum_{i=1}^{N} \log[\mathcal{N}(x_i|\mu, \sigma)\mathcal{N}(\mu; \mu_0, \sigma_0)] \right)$$

- How to solve "argmax"? Well easy, take the deriviative and let it equal zero. Works in the Gaussian case.

MAP Example Conti.

- Same trick applies: take the derivative with respect of μ and let it equal zero
- If you write out the expression for Gaussian fully, you will get:

$$\mu^{\text{MAP}} = \frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2} \left(\frac{1}{n} \sum_{j=1}^{n} x_i \right) + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2} \mu_0$$

- see what happens if $\sigma_0 \to \infty$

Expectation Maximization

If lucky, can find $\arg\max_\theta \log[p(X|\theta)p(\theta)]$, i.e., take the derivative and let it equal zero analytically
In many cases, we have to use some numerical methods, such as Expectation-Maximization (EM)
http://www-staff.it.uts.edu.au/ ydxu/stat/incremental.pdf

Given an initial parameter θ^1, we obtain a set of parameter estimate $\{\theta^1, \ldots \theta^g, \theta^{g+1}, \ldots\}$, such that:

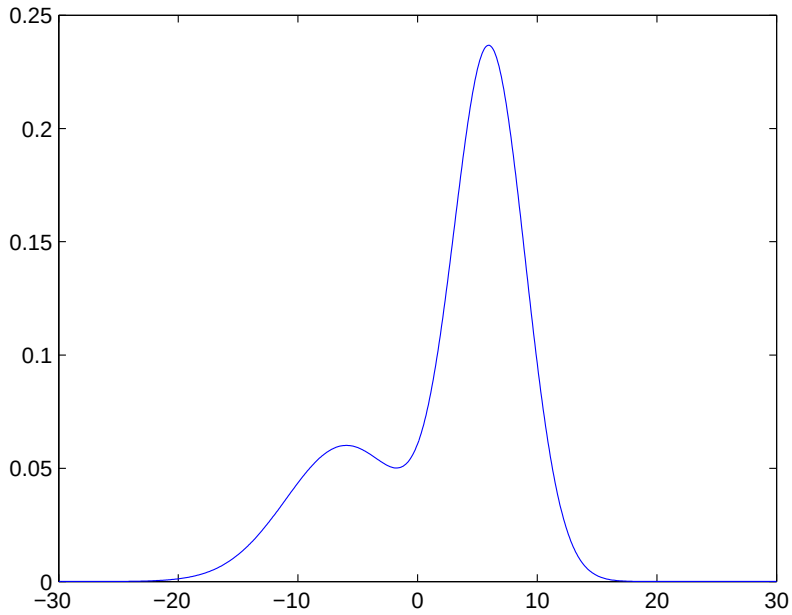
$$\log[p(X|\theta^{g+1})p(\theta^{g+1})] \geq \log[p(X|\theta^g)p(\theta^g)]$$

Two examples

- Gaussian Mixture model: $p(x|\theta) = \sum_{l=1}^{M} \mathcal{N} w_l(x; \mu_l, \sigma_l)$
- An example of my research: (Xu & Kemp, 2010 & 2013)

The moral of the story

- Doesn't matter how sophisticated they are, these algorithms are point estimators, as they simply give you a "best" **single** θ.
- In many machine learning problems, you are actually interested in the posterior distribution $p(\theta|\text{Data}) \propto p(\text{data}|\theta)p(\theta)$
- Ok, let's look at an example:

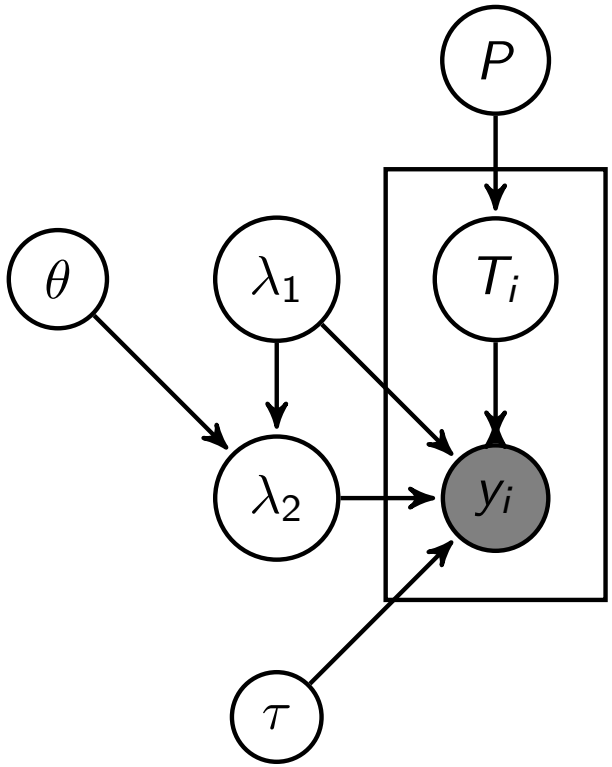


Your first "almost real" sampling algorithm

Generative model

$$
\begin{aligned}
P &\sim \text{Dir}(\alpha, \alpha) \\
\tau &\sim \text{Gamma}(a, b) \\
\theta &\sim \mathcal{N}(0, \sigma_\theta^2) \\
\lambda_1 &\sim \mathcal{N}(0, \sigma_\lambda^2) \\
T_i | P &\sim \text{Mult}(P) \\
\lambda_2 &\sim \mathbf{1}(\lambda_1 + \theta) \\
y_i | \lambda_{T_i}, \tau &\sim \mathcal{N}(\lambda_{T_i}, 1/\tau)
\end{aligned}
$$

Graphical model



What you are hoping to get is of course $p(T_i, P, \lambda_1, \lambda_2, \tau, \theta | \{y_1, \ldots, y_N\})$

Sampling or other methods?

Do these problems only can to be resolved by sampling? Well not quite. For example, in (Bishop, 2006), there is an example where the previous problem was solved by Variational Bayes.

- Variational Bayes - good starting point: chapter 10 of Bishop's textbook, and/or **http://www-staff.it.uts.edu.au/[TILDE]ydxu/stat/variation_bayes_with_gmm.pdf**
- Convex optimization
- Laplace approximation
- ...

Any easy way out?

What if I don't want to write my own sampling code? The good news is that you don't have to. You can simply use WINBUGS. **http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml**, get a feeling for it: this is from http://www.mrc-bsu.cam.ac.uk/bugs/documentation/exampVol2/node5.html

```
model eyes;
const
    N=48;
var
    y[N],
    T[N],
    lambda[2],
    theta,
    tau,
    sigma,
    P[2],
    alpha[2];
data y in "eyes.dat";
inits in "eyes.in";
{for (i in 1:N){
    y[i] ~ dnorm(lambda[T[i]],tau);
    T[i] ~ dcat(P[])}
sigma <- 1/sqrt(tau);
tau ~ dgamma(0.01,0.01);
lambda[1] ~ dnorm(0,1.0E-6);
lambda[2] <- lambda[1]+theta;
theta ~ dnorm(0,1.0E-6) I(0,);
P[] ~ ddirch(alpha[]);
alpha[1] <- 1;
alpha[2] <- 1;}
```

Can I leave now?

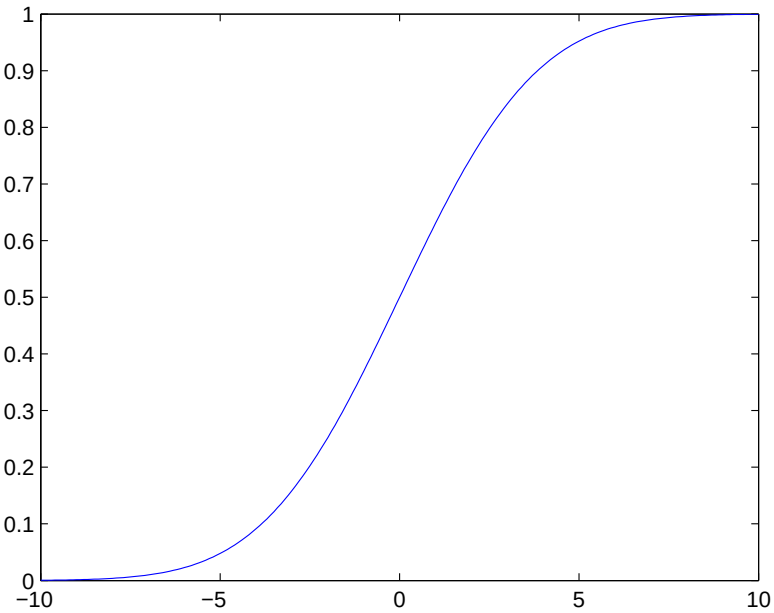
No :)

- Can't just use Winbugs for all models, it's a black-box,
- In many scenarios, you need to develop your own efficient sampling
- ...

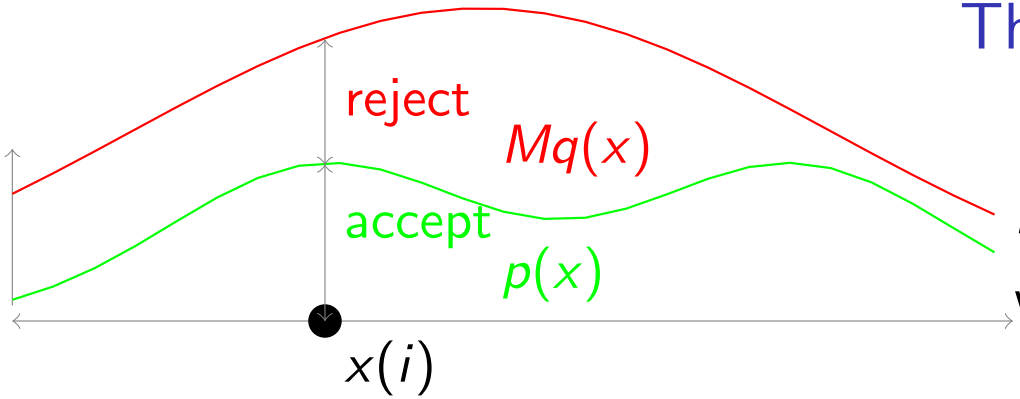
Ok, let's start sampling!

Can I leave now?

Simplist sampling method: sample the inverse of CDF!



Rejection Sampling



Sampling is all about efficiency, rejection sampling can give you quite low acceptance ratio, should you choose a non-compatible $q(.)$:

The algorithm

$i = 0$

while $i \neq N$

$x(i) \sim q(x)$ and $u \sim U(0, 1)$

if $u < \dfrac{p(x(i))}{Mq(x(i))}$ then

accept $x(i)$

$i = i + 1$

else

reject $x(i)$

end

end

Adaptive Rejection Sampling

Sometimes, rejection sampling can be made more efficient. One example is when $p(x)$ is log-concave. For example, in Dirichlet Process, the concentration factor α has probabillity:

$$p(\alpha|k,n) \propto \frac{\alpha^{k-3/2}\exp(-1/(2\alpha))\Gamma(\alpha)}{\Gamma(n+\alpha)}$$

where $\log(\alpha)$ is log-concave.

ARS steps

- Let $\{x_i, \ldots, x_k\}$ be the k starting points.
- Calculate $u_k(x)$, the piece-wise linear upper bound formed from the tangents to $h(x)$ at each point x_i
- $s_k(x) = \frac{\exp(u_k(x))}{\int \exp(u_k(x'))dx'}$
- $z_j = \frac{h(x_{j+1}) - h(x_j) - x_{j+1}h'(x_{j+1}) + x_j h'(x_j)}{h'(x_j) - h'(x_{j+1})}$
- Piece-wise upper bound: $u_k(x) = h(x_j) + (x - x_j)h'(x_j)$ for $x \in [z_{j1}, z_j]$ and $j = 1, \ldots, k$

The Sampling steps

- Sample $x^* \sim s_k(x)$ and $u^* \sim U(0, 1)$.
- **If** $u^* \leq \exp\{h(x^*)u_k(x^*)\}$ **then** accept x^*, otherwise reject x^*.
- Include x^* in the list, so it has $K + 1$ elements, and rearrange in ascending order and reconstruct functions $u_{k+1}(x), s_{k+1}(x)$

ARS demos

- Watch demo of sampling a Gaussian distribution (no need to use ARS, but ok for demo)
- You can further impove its efficiency by including a lower bound. (Any guess?)

Importance Sampling

Say, for example, the aim for this task is to compute the integral:

$$\begin{aligned}
\mathrm{E}_{p(z)}[f(z)] &= \int f(z)p(z)dz \\
&= \int \underbrace{f(z)\frac{p(z)}{q(z)}}_{\text{new}\tilde{f}(z)} q(z)dz \\
&\approx \frac{1}{N}\sum_{n=1}^{N} f(z^i)\frac{p(z^i)}{q(z^i)}
\end{aligned}$$

$\tilde{p}$ is the un-normalized pdf, i.e., $p(z) = \frac{\tilde{p}}{Z}$

Markov Chain Monte Carlo

The product between two markov matrix is again a markov matrix. For example:

$$
\begin{aligned}
AB &= \begin{bmatrix} a_{11} & 1-a_{11} \\ a_{21} & 1-a_{21} \end{bmatrix} \begin{bmatrix} b_{11} & 1-b_{11} \\ b_{21} & 1-b_{21} \end{bmatrix} \\
&= \begin{bmatrix} a_{11}b_{11} + (1-a_{11})b_{21} & a_{11}(1-b_{11}) + (1-a_{11})(1-b_{21}) \\ a_{21}b_{11} + (1-a_{21})b_{21} & a_{21}(1-b_{11}) + (1-a_{21})(1-b_{21}) \end{bmatrix}
\end{aligned}
$$

It is clear that, also can be proven in the general case:

$$
\begin{aligned}
& a_{11}b_{11} + (1-a_{11})b_{21} + a_{11}(1-b_{11}) + (1-a_{11})(1-b_{21}) \\
=& a_{11}b_{11} + b_{21} - a_{11}b_{21} + a_{11} - a_{11}b_{11} + 1 - a_{11} - b_{21} + a_{11}b_{21} \\
=& 1
\end{aligned}
$$

Markov Chain Monte Carlo

Assume that there is a markov transition matrix P, there is a corresponding vector v_i, such that:

$$\begin{bmatrix} v_1 & \dots & v_d \end{bmatrix} \begin{bmatrix} p_{11} & \dots & p_{1d} \\ \dots & \dots & \dots \\ b_{21} & \dots & p_{1d} \end{bmatrix} = \begin{bmatrix} v_1 & \dots & v_d \end{bmatrix}$$

Question What is the replationshiop between $\begin{bmatrix} v_1 & \dots & v_d \end{bmatrix}$ and P?
Assume:

$$\lim_{n \to \infty} P^n = P^\infty \implies \lim_{n \to \infty} P^{n-1} = P^\infty$$

Markov Chain Monte Carlo

A computer scientist's perspective:

- A Markov chain is a sequence of random variables $X_0, \ldots, X_n$ generated by a Markov process.
- It has a transition kernel K contains $\{k(x^*|x)\}\forall x, x^*$: each $k(x^*|x)$ is probability a process at state x moves to state x^* in a single step.
- At some stage, you want the chain to reach a **stationary distribution** π^*, where apply K to samples distributed from π^* does NOT change its distribution.

Markov Chain Monte Carlo

Chapman-Kologronvo equation, where in general, in a markov process:

$$\pi_t(x^*) = \int \pi_{t-1}(x)k(x^*|x)dx$$

Obviously, at equilibrium, that stationary distribution satisfies

$$\pi(x^*) = \int \pi(x)k(x^*|x)dx$$

The **detailed balance** condition holds when
$\pi(x)k(x^*|x) = \pi(x^*)k(x|x^*)$

$$\int \pi(x)k(x^*|x)dx = \int \pi(x^*)k(x|x^*)dx = \pi(x^*)\int k(x|x^*)dx = \pi(x^*)$$

Metropolis Hasting Algorithm

1. Initialise x^0
2. For i $= 0$ to $N-1$

   $u \sim U(0,1)$

   $x^* \sim q(x^*|x^{(i)}$

   if $u < \alpha(x^*) = \min\left(1, \frac{\pi(x^*)q(x|x^*)}{\pi(x)q(x^*|x)}\right)$ $\qquad\qquad$ $x^{(i+1)} = x^*$

   if

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

The take-home message here, is that it does not "disgard" samples like rejection sampling. It simply "repeats" samples.
If the same sample repeats too many times, it has bad mixing.

See demo for an example.

Metropolis Hasting - Why it work?

Propose x^* from $q(x^*|x)$, then accept x^* with ratio $\alpha(x^*)$, where

$$\alpha(x^*) = \min\left(1, \frac{\pi(x^*)q(x|x^*)}{\pi(x)q(x^*|x)}\right)$$

You can show very easily why it satisfy detailed balance:

$$\begin{aligned}
\pi(x)q(x^*|x)\alpha(x^*) &= \pi(x)q(x^*|x)\min\left(1, \frac{\pi(x^*)q(x|x^*)}{\pi(x)q(x^*|x)}\right) \\
&= \min\left(\pi(x)q(x^*|x), \pi(x^*)q(x|x^*)\right) \\
&= \pi(x^*)q(x|x^*)\min\left(1, \frac{\pi(x)q(x^*|x)}{\pi(x^*)q(x|x^*)}\right) \\
&= \pi(x^*)q(x|x^*)\alpha(x)
\end{aligned}$$

More on the efficiency of Metropolis Hasting

- You can imagine that if $q(.)$ has nothing to do $\pi(.)$, then the move isn't really that great:
- Think about a $\pi(x)$ is a postive correlated 2-d multivariate Gaussian, but with a q(.) is a sphereical gaussian proposal. Could you see potential problem?
- Hybrid Metropolis Hasting tries to combat this: one of the important note is that the step is taken according to $\frac{\partial \log p(x)}{\partial x}$
- $\frac{\partial \log p(x)}{\partial x} = \frac{p'(x)}{p(x)}$ Think about that for a moment.

Gibbs sampling Demo

In this toy example, let's sample:

$$x \sim \mathcal{N}(\mu, \Sigma)$$

where $\mu = [23]^T$ and $\Sigma = \begin{bmatrix} 3 & 2 \\ 2 & 5 \end{bmatrix}$

Gibbs sampling

- Perhpas the most frequently used sampling method - well it's easy to implement.
- Whenever you see "inference section" in a machine learning paper, it's most likely to show a Gibbs sampling scheme.

Gibbs sampling algorithm: given a starting sample $(x^1, y^1, z^1)^T$, you want to sample $(x^2, y^2, z^2)^T$, $(x^3, y^3, z^3)^T$,..., $(x^N, y^N, z^N)^T$ from P(x,y,z), then the algorithm goes:

$$
\begin{aligned}
x^2 &\sim P(x|y^1, z^1) \\
y^2 &\sim P(y|x^2, z^1) \\
z^2 &\sim P(z|x^2, y^2) \\
x^3 &\sim P(x|y^2, z^2) \\
y^3 &\sim P(y|x^3, z^2) \\
z^3 &\sim P(z|x^3, y^3) \\
&\dots
\end{aligned}
$$

You can see the algorithm won't be able to "parallellise". However, under some models (and clever work-around) machine learning researcher able to parallelise some Gibbs sampling schems for various models. Well, make sense to perform inference to **Big data** with CUDA, multiple processors.

A special case of M-H

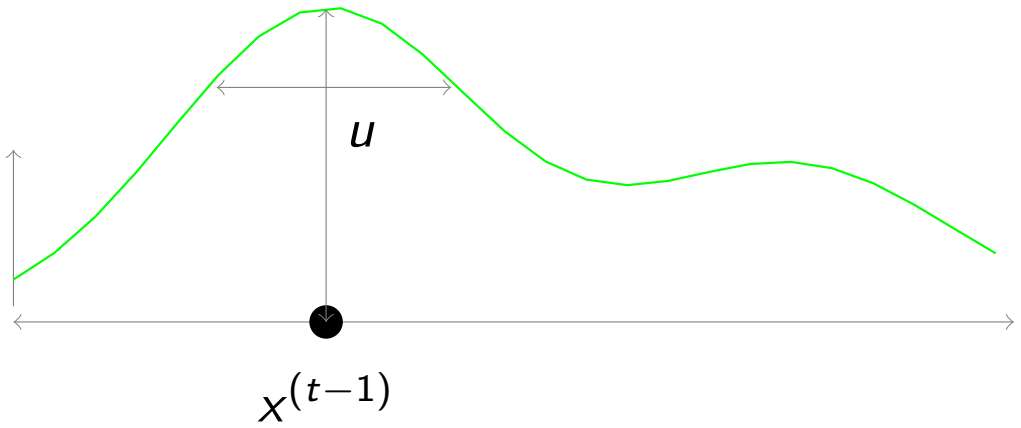
Looking at the M-H accpetance ratio

- Let $\mathbf{x} = x_1, \ldots, x_D$.
- When sampling k^{th} component, $q_k(\mathbf{x}^*|\mathbf{x}) = \pi(x_k^*|\mathbf{x}_{-k})$
- When sampling k^{th} component, $\mathbf{x}^*_{-k} = \mathbf{x}_{-k}$

$$\frac{\pi(\mathbf{x}^*)q(\mathbf{x}|\mathbf{x}^*)}{\pi(\mathbf{x})q(\mathbf{x}^*|\mathbf{x})} = \frac{\pi(\mathbf{x}^*)\pi(x_k|\mathbf{x}^*{}_{-k})}{\pi(\mathbf{x})\pi(x_k^*|\mathbf{x}_{-k})} = \frac{\pi(x_k^*|\mathbf{x}^*_{-k})\pi(x_k|\mathbf{x}^*{}_{-k})}{\pi(x_k|\mathbf{x}_{-k})\pi(x_k^*|\mathbf{x}_{-k})} = 1$$

A real example

Slice Sampling



$$u \sim U(0, p(x^{(t-1)}))$$

$$x^{(t)} \sim \mathbf{1}[p(x) > u]$$

Very powerful technique, been working with it in a number of non-parametric Bayes settings.

Slice Sampling - Why it works?

Introduce auxiliary variable u a joint distribution over (x, u) is defined as:

$$p(x,u) = \begin{cases} 1/Z & \text{if } 0 < u < f(x) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{where } Z = \int f(x)dx \implies p(x) = \frac{f(x)}{Z}$$

The joint density of (x, u) is uniform over the region

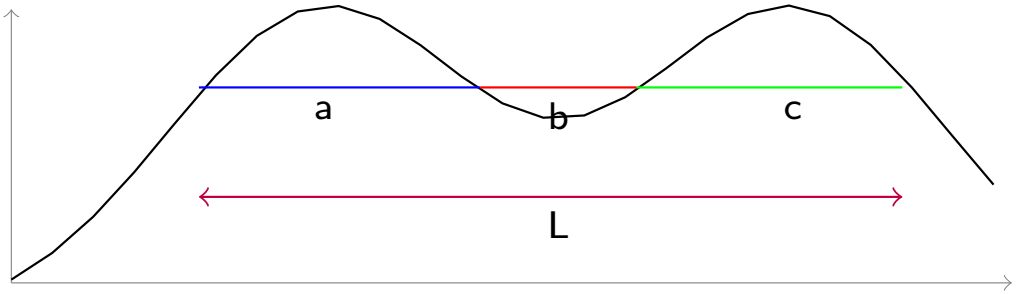
$$U = \{(x,u) : 0 < u < f(x)\}$$

Marginal distribution over x is:

$$p(x) = \int_0^{f(x)} \frac{1}{Z} du = \frac{f(x)}{Z} = p(x)$$

Slice Sampling - Shrink algorithm

- It is not always simple to sample $x^{(t)} \sim \mathbf{1}[p(x) > u]$
- We need to use "shrinkage algoirthm" or "expansion algorithm"
- to see why "shrinkage" targets the right distribution:



$$\Pr(X^{(t)} \in a \to X^{(t+1)} \in a) = \frac{a+b}{L}$$
$$\Pr(X^{(t)} \in a \to X^{(t+1)} \in c) = \frac{c}{L}$$
$$\Pr(X^{(t)} \in c \to X^{(t+1)} \in a) = \frac{a}{L}$$
$$\Pr(X^{(t)} \in c \to X^{(t+1)} \in c) = \frac{b+c}{L}$$

Therefore,

$$\begin{aligned}\Pr(X^{(t+1)} \in a) &= \Pr(X^{(t)} \in a \to X^{(t+1)} \in a)\Pr(X^{(t)} \in a) + \Pr(X^{(t)} \in c \to X^{(t+1)} \in a)\Pr(X^{(t)} \in c) \\ &= \frac{a+b}{L}\frac{a}{L} + \frac{a}{L}\frac{c}{L} = \frac{a^2+ab+ac}{L^2} = \frac{a}{L} = \Pr(X^{(t)} \in a)\end{aligned}$$

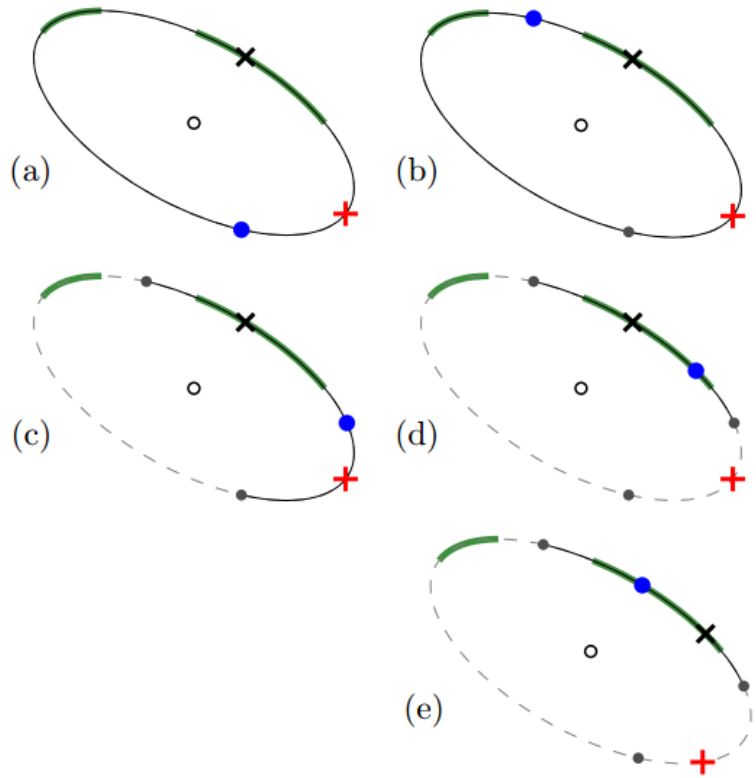
$$\begin{aligned}\Pr(X^{(t+1)} \in c) &= \Pr(X^{(t)} \in a \to X^{(t+1)} \in c)\Pr(X^{(t)} \in a) + \Pr(X^{(t)} \in c \to X^{(t+1)} \in c)\Pr(X^{(t)} \in c) \\ &= \frac{c}{L}\frac{a}{L} + \frac{b+c}{L}\frac{c}{L} = \frac{ac+bc+c^2}{L^2} = \frac{c}{L} = \Pr(X^{(t)} \in c)\end{aligned}$$

$$\Pr(X^{(t+1)}) = \Pr(X^{(t)})$$

Elliptical Slice Sampling

Murray, Iain, and Ryan P. Adams. "Slice sampling covariance hyperparameters of latent Gaussian models.", NIPS 2010.

1. Choose ellipse: $\boldsymbol{\nu} \sim \mathcal{N}(0, \Sigma)$
2. Log-likelihood threshold:
$$u \sim \text{Uniform}[0, 1]$$
$$\log y \leftarrow \log L(\mathbf{f}) + \log u$$
3. Draw an initial proposal, also defining a bracket:
$$\theta \sim \text{Uniform}[0, 2\pi]$$
$$[\theta_{\min}, \theta_{\max}] \leftarrow [\theta - 2\pi, \theta]$$
4. $\mathbf{f}' \leftarrow \mathbf{f} \cos\theta + \boldsymbol{\nu} \sin\theta$
5. **if** $\log L(\mathbf{f}') > \log y$ **then:**
6. Accept: **return** $\mathbf{f}'$
7. **else:**
Shrink the bracket and try a new point:
8. **if** $\theta < 0$ **then:** $\theta_{\min} \leftarrow \theta$ **else:** $\theta_{\max} \leftarrow \theta$
9. $\theta \sim \text{Uniform}[\theta_{\min}, \theta_{\max}]$
10. **GoTo** 4.



"Same-dimension" rotation factor of Gaussian

Let $\begin{bmatrix} x_1' \\ v_1' \end{bmatrix} = \mathcal{R}(\theta) \begin{bmatrix} x_1 \\ v_1 \end{bmatrix}$ and $\begin{bmatrix} x_2' \\ v_2' \end{bmatrix} = \mathcal{R}(\theta) \begin{bmatrix} x_2 \\ v_2 \end{bmatrix}$

Collectively, we write the above as: $\begin{bmatrix} \mathbf{x}' \\ \mathbf{v}' \end{bmatrix} = \mathcal{R}(\theta) \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix} \implies \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix} = \mathcal{R}(-\theta) \begin{bmatrix} \mathbf{x}' \\ \mathbf{v}' \end{bmatrix}$

$$\begin{aligned} &\mathcal{N}(\mathbf{x}'|0,\Sigma)\mathcal{N}(\mathbf{v}'|0,\Sigma) \\ &= \exp\left[\frac{-1}{2}\left(\Sigma_{1,1}({x'}_1^2 + {v'}_1^2) + 2\Sigma_{1,2}({x'}_1{x'}_2 + {v'}_1{v'}_2) + \Sigma_{2,2}({x'}_2^2 + {v'}_2^2)\right)\right] \end{aligned}$$

We know that,

$$\begin{aligned} {x'}_1{x'}_2 + {v'}_1{v'}_2 &= [\cos(\theta)x_1 - \sin(\theta)v_1][\cos(\theta)x_2 - \sin(\theta)v_2] + [\sin(\theta)x_1 + \cos(\theta)v_1][\sin(\theta)x_2 + \cos(\theta)v_2] \\ &= \cos^2(\theta)x_1x_2 - \cos(\theta)\sin(\theta)x_1v_2 - \sin(\theta)\cos(\theta)v_1x_2 + \sin^2(\theta)v_1v_2 \\ &\quad + \sin^2(\theta)x_1x_2 + \cos(\theta)v_1x_2 + \sin(\theta)x_1v_2 + \cos^2(\theta)v_1v_2 \\ &= x_1x_2 + v_1v_2 \end{aligned}$$

It is then obvious that,

$$\mathcal{N}(\mathbf{x}'|0,\Sigma)\mathcal{N}(\mathbf{v}'|0,\Sigma) = \mathcal{N}(\mathbf{x}|0,\Sigma)\mathcal{N}(\mathbf{v}|0,\Sigma)$$

Elliptical Slice Sampling - detailed balance

$$
\begin{aligned}
p(f, y, v, \{\theta_k\}) &= p^*(f)p(y|f)p(v)p(\{\theta_k\}|f, v, y) \\
&= L(f)\mathcal{N}(f|0, \Sigma)\frac{1}{L(f)}\mathcal{N}(v|0, \Sigma)p(\{\theta_k\}|f, v, y) \\
&= \mathcal{N}(f|0, \Sigma)\mathcal{N}(v|0, \Sigma)p(\{\theta_k\}|f, v, y)
\end{aligned}
$$

We need to prove that,

$$
\mathcal{N}(f'|0, \Sigma)\mathcal{N}(v'|0, \Sigma)p(\{\theta_k\}|f', v', y') = \mathcal{N}(f|0, \Sigma)\mathcal{N}(v|0, \Sigma)p(\{\theta_k\}|f, v, y)
$$

Importance sampling again

To approximate the integral, but $p(z)$ is hard to sample.

$$\begin{aligned} \mathrm{E}_{p(z)}[f(z)] &= \int f(z)p(z)dz \\ &= \int \underbrace{f(z)\frac{p(z)}{q(z)}}_{\text{new}\tilde{f}(z)} q(z)dz \\ &\approx \frac{1}{N}\sum_{n=1}^{N} f(z^i)\frac{p(z^i)}{q(z^i)} \end{aligned} \tag{1}$$

Revision on SMC

Take Importance Sampling to higher dimensions, the importance weights are:

$$w_n(x_{1:n}) = \frac{\gamma(x_{1:n})}{q_n(x_{1:n})} \tag{2}$$

Hard to choose $q(.)$ in high-dimension
Solution : rewrite equation (2) in the following:

$$w_n(x_{1:n}) = \frac{\gamma(x_{1:n})}{q_n(x_n|x_{1:n-1})q_{n-1}(x_{1:n-1})} \times \frac{\gamma(x_{1:n-1})}{\gamma(x_{1:n-1})}$$

re-arrange:

$$w_n(x_{1:n}) = \frac{\gamma(x_{1:n})}{q(x_{1:n})} = w_{n-1}(x_{1:n-1}) \times \frac{\gamma(x_{1:n})}{\gamma(x_{1:n-1})q(x_n|x_{1:n-1})}$$

Revision on SMC (2)

Top-down:

$$w_n(x_{1:n}) = w_{n-1}(x_{1:n-1}) \frac{\gamma(x_{1:n})}{\gamma(x_{1:n-1}) q(x_n | x_{1:n-1})} \tag{3}$$

Bottom-up:

$$w_n(x_{1:n}) = w_1(x_1) \prod_{j=2}^{n} \frac{\gamma(x_{1:j})}{\gamma(x_{1:j-1}) q(x_j | x_{1:j-1})}$$

The two are equivalent

Just too easy to put it all in an algorithm:

The SIS algorithm:

$$
\begin{aligned}
&\text{At dimension } n = 1\text{: For each particle } i \\
&\quad \text{Sample } x_1^i \sim q_1(x_1) \\
&\quad \text{Compute the weights } w_1^i \propto \frac{\gamma(x_1^i)}{q_1(x_1^i)} \\
&\text{At dimension } n \geq 2\text{: For each particle } i \\
&\quad \text{Sample } x_n^i \sim q_n(x_n | x_{1:n-1}^i) \\
&\quad \text{Compute the weights } w_n^i \propto w_{n-1}^i \frac{\gamma(x_{1:n}^i)}{\gamma(x_{1:n-1}^i) q(x_n^i | x_{1:n-1}^i)}
\end{aligned}
\tag{4}
$$

Particle Filter

Put this in a state-space setting, you have particle filter!
By changing n to t to reflect time sequentiality. In here, we assume that:

$$p(x_{1:t}|y_{1:t}) = \frac{p(x_{1:t}, y_{1:t})}{p(y_{1:t})} = \frac{\gamma_t(x_{1:t})}{\mathcal{Z}}$$

In here, we assume:

$$\begin{aligned}\gamma_t(x_{1:t}) &= p(x_{1:t}, y_{1:t}) \\ &= p(y_t|x_{1:t}, y_{1:t-1})p(x_t|x_{1:t-1}, y_{1:t-1})\gamma_{t-1}(x_{1:t-1}) \\ &= p(y_t|x_t)p(x_t|x_{t-1})\gamma_{t-1}(x_{1:t-1})\end{aligned} \tag{5}$$

Particle Filter

Divide by the proposal distribution $q(.)$, and do the same trick, this time, we use:

$$w_t(x_{1:t}) = \frac{\gamma_{(1:t)}}{q_{(1:t)}} = \frac{\gamma_{(1:t-1)}}{q_{(1:t-1)}} \times \frac{p(y_t|x_t)p(x_t|x_{t-1})}{q(x_t|x_{1:t-1})}$$

we can make a "reasonable" assumption that:

$$q(x_t|x_{1:t-1}) \equiv q(x_t|x_{t-1}, y_t) \tag{6}$$

Hence,

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

question is How are we going to choose $q(.)$ **a short answer** Choose $q(.)$ somehow from your dynamic model

Optimal proposal: $q(x_t|x_{k-1}, y_t) = p(x_t|x_{k-1}, y_t)$

Stated in [Doucet 1998], $q(x_t|x_{k-1}, y_t) = p(x_t|x_{k-1}, y_t)$ is optimal,then:

$$\begin{aligned} w_{(1:t)} &\propto w_{(1:t-1)} \times \frac{p(y_t|x_t)p(x_t|x_{t-1})}{p(x_t|x_{t-1}, y_t)} \\ &= w_{(1:t-1)} \times \frac{p(y_t|x_t)p(x_t|x_{t-1})p(y_t|x_{t-1})p(x_{t-1})}{p(y_t|x_t)p(x_t|x_{t-1})p(x_{t-1})} \\ &= w_{(1:t-1)} \times p(y_t|x_{t-1}) \end{aligned}$$

However, $p(y_t|x_{t-1})$ is quite meaningless:

$$w_{(1:t)} \propto w_{(1:t-1)} \times \int_{x_t} p(y_t|x_t)p(x_t|x_{t-1}) \tag{7}$$

Two problem: (1) Difficult to sample from $p(x_t|x_{k-1}, y_t)$ and (2) integral is difficult to perform!

Main talk: sub-optimal methods

In this talk, I will present two "popular" sub-optimal sampling methods first:

- Bootstrap Particle Filter
- Auxiliary Particle Filter

Bootstrap Particle Filter

Sometimes calling it Condensational Filter. (Famous Michael Isard)
Let $q(x_t|x_{k-1}, y_t) = p(x_t|x_{k-1})$, i.e., y_t does not participate in the proposal $q(.)$

$$\begin{aligned} w_{(1:t)} &\propto w_{(1:t-1)} \times \frac{p(y_t|x_t)p(x_t|x_{t-1})}{p(x_t|x_{t-1})} \\ &= w_{(1:t-1)} \times p(y_t|x_t) \end{aligned} \tag{8}$$

- particles x_t^i are sampled from $p(.|x_{t-1})$, but are weighted by $p(y_t|x_t^i)$
- the danger is that x_t^i may receive close to zero weight if $p(y_t|x_t^i)$ is very small.

The Condensational Filter algorithm:

At time t

For each particle i:

$$\begin{aligned} &\text{Sample } x_t^i \sim p(x_t|x_{t-1}^i) \qquad \left(\text{Or } x_1^i \sim p(x_1) \text{ when } t=1\right) \\ &\text{Compute the weights } w_t^i \propto \pi_{t-1}^i p(y_t|x_t^i) \end{aligned} \tag{9}$$

$$\text{normalize weights } \pi_t^i = \frac{w_t^i}{\sum_{i=1}^N w_t^i}$$

Problem particle degeneracy occurs very quickly.
Solution break those big particle into smaller ones, from the "re-sampling" step. To determine if "big particles" exist, check effective particle size.
BTW re-sampling does not solve particle degeneracy problem altogether.

Introducing Re-Sampling

Re-sampling sometimes can be considered as jointly "sample" an index i^j to indicate which $x_{t-1}^{i^j}$ generated x_t^i, and x_t^i itself.

$$\begin{aligned} x_t^i &\sim q(x_t|x_{t-1}^i, y_t) \\ &\text{becomes:} \\ j &\sim \pi_{t-1}(x_{1:t-1}) \\ x_t^i &\sim q(x_t|x_{t-1}^{i^j}, y_t) \end{aligned} \tag{10}$$

For each particle i at time t, you get (x_t^i, i^j).

Introducing Re-Sampling

Substituting N of the (x_t^i, i^j) into the following:

$$w_t(x_{1:t}) \propto \pi_{t-1}(x_{1:t-1}) \times \frac{p(y_t|x_t)p(x_t|x_{t-1})}{q(x_t|x_{t-1}, y_t)}$$

$$w_t^i(x_{1:t}) \propto \pi_{(t-1)}^{i^j} \times \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^{i^j})}{\pi_{(t-1)}^{i^j} q(x_t^i|x_{t-1}^{i^j}, y_t)}$$

$$= \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^{i^j})}{q(x_t^i|x_{t-1}^{i^j}, y_t)}$$

In the bootstrap filter:

$$w_t^i(x_{1:t}) \propto p(y_t|x_t^i)$$

The Condensational Filter algorithm:

At time t

For each i:

Sample $j \sim \pi_{t-1}(x_{1:t-1})$ – choose an an ancestor

Sample $x_t^i \sim p(x_t|x_{t-1}^{i^j})$ $\left(\text{Or } x_1^i \sim p(x_1) \text{ when } t = 1\right)$ (11)

Compute the weights $w_t^i \propto p(y_t|x_t^i)$

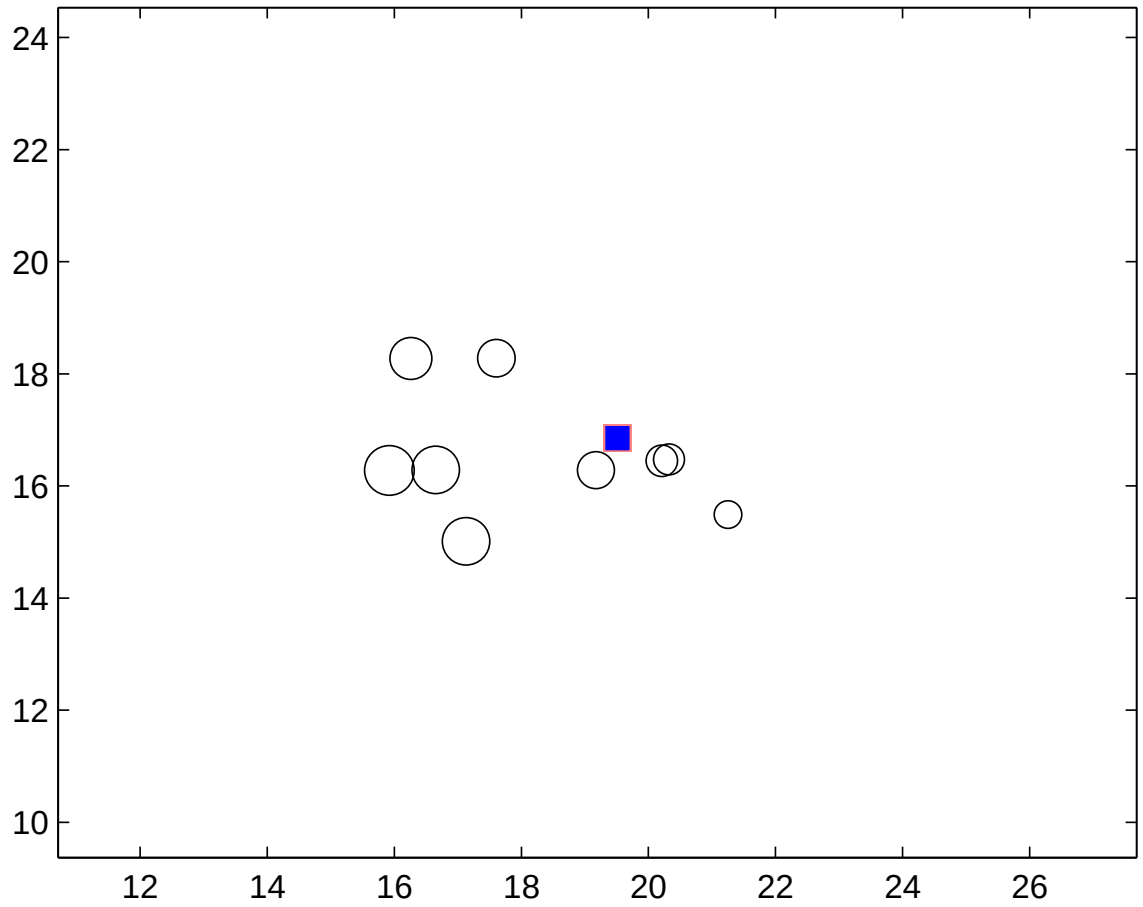
normalize weights $\pi_t^i = \dfrac{w_t^i}{\sum_{i=1}^N w_t^i}$

A little demo

- $p(x_t|x_{t-1}) = \mathcal{N}(Ax_{t-1} + B, Q)$
- $p(y_t|x_t) = \mathcal{N}(x_t, R)$

This is just for demo purpose, you can compute $p(x_t|y_{1:t})$ exactly using Kalman Filter!

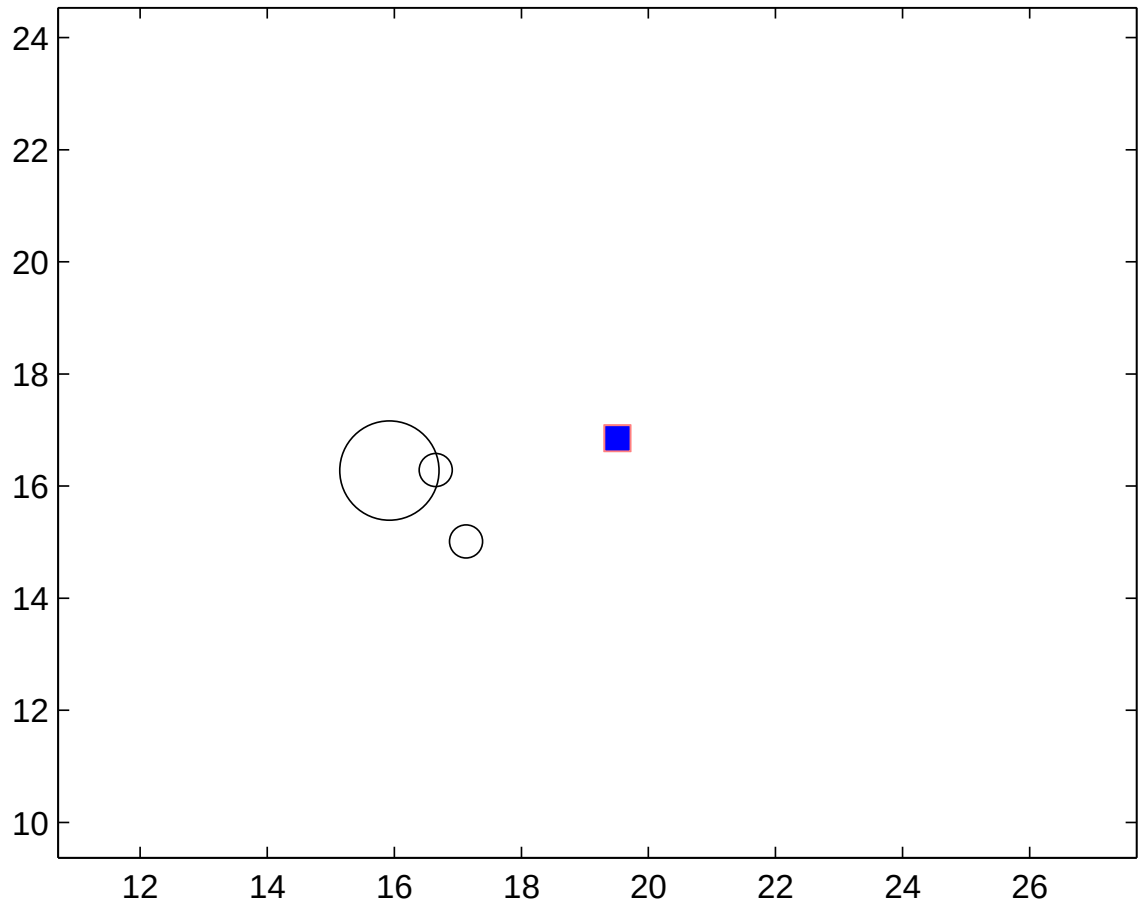
Representation for $p(x_{t-1}|y_{1:t-1})$



- Circles are weighted particle representation of $p(x_{t-1}|y_{1:t-1})$
- The blue square is y_t

Re-sampling

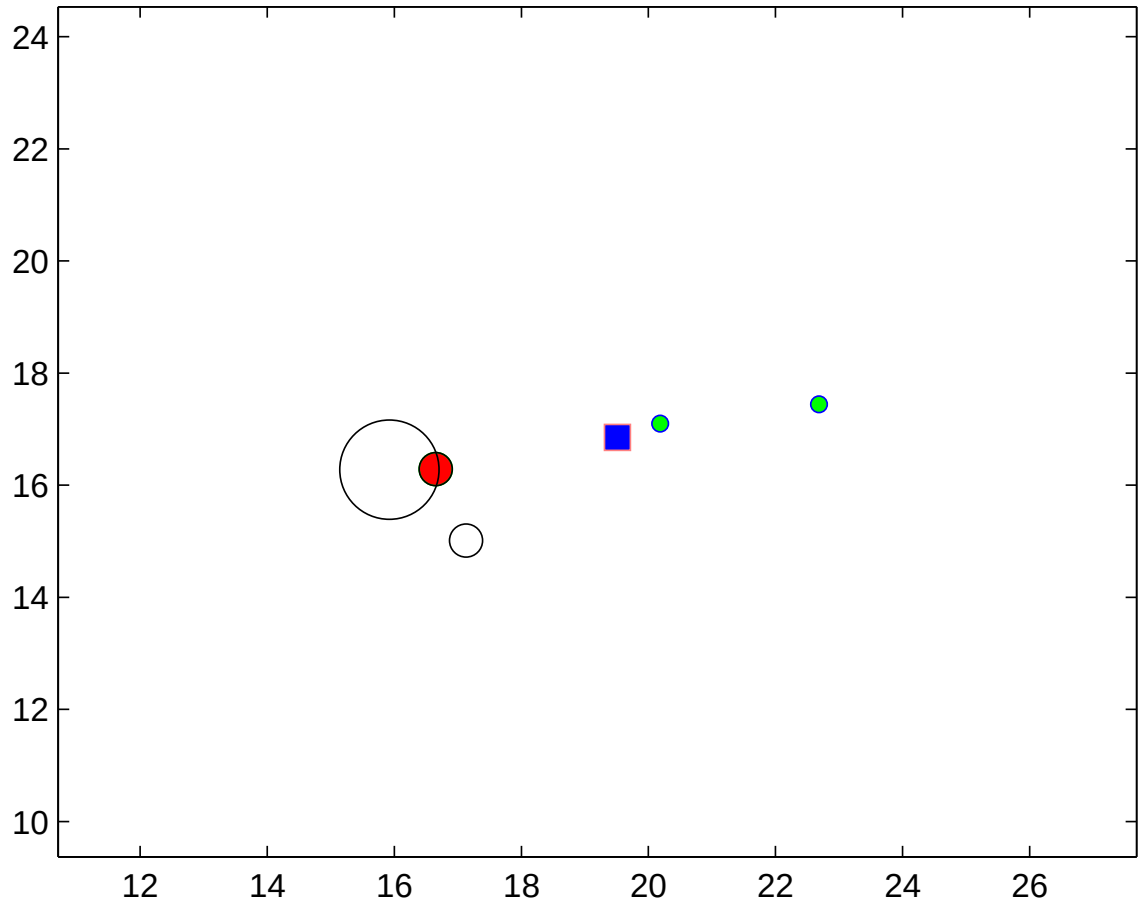
To sample $j \sim \pi_{t-1}(x_{1:t-1})$:



- Size of the circle indicates the number of times $x_{t-1}^{i^j}$ was selected.

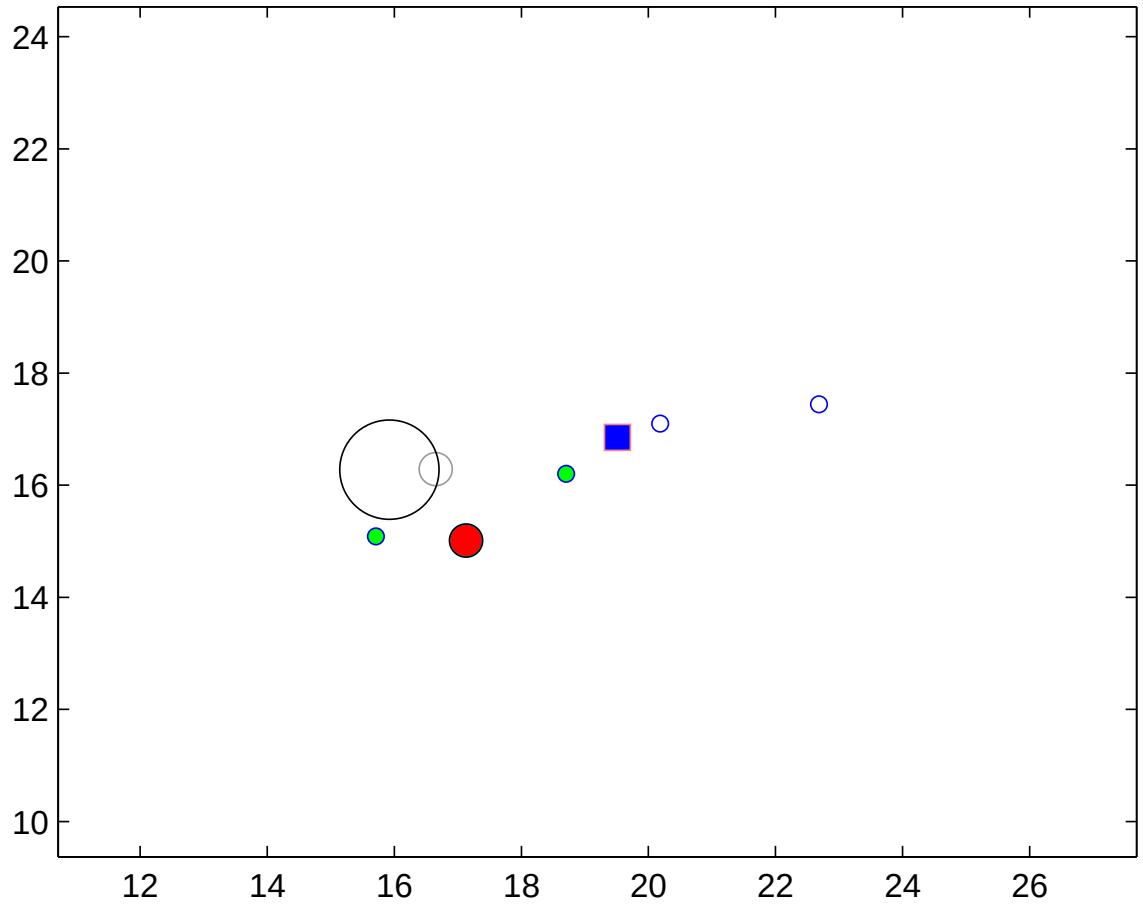
Transition demos

Sample $x_t^i \sim p(x_t|x_{t-1}^{i^j}) : \forall i^j = 1$



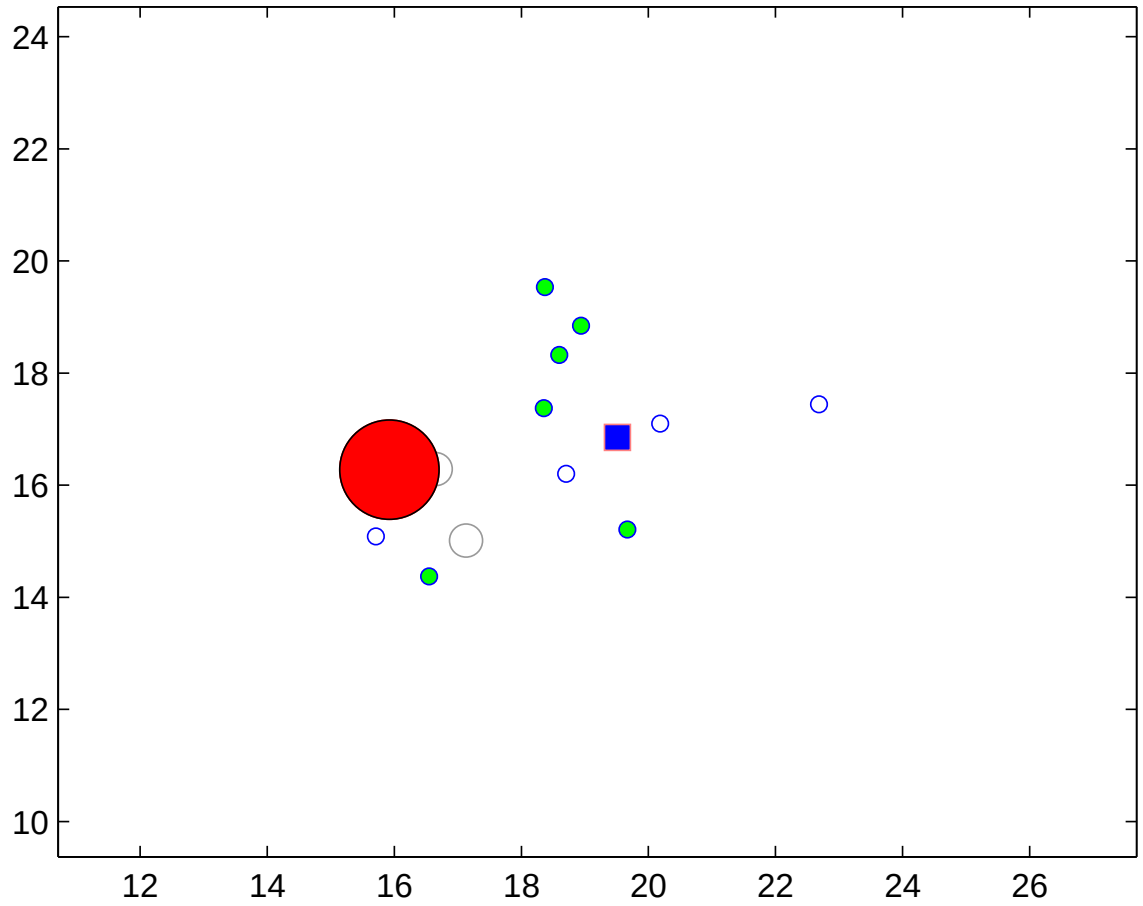
Transition demos

Sample $x_t^i \sim p(x_t|x_{t-1}^{i^j}) : \forall i^j = 2$



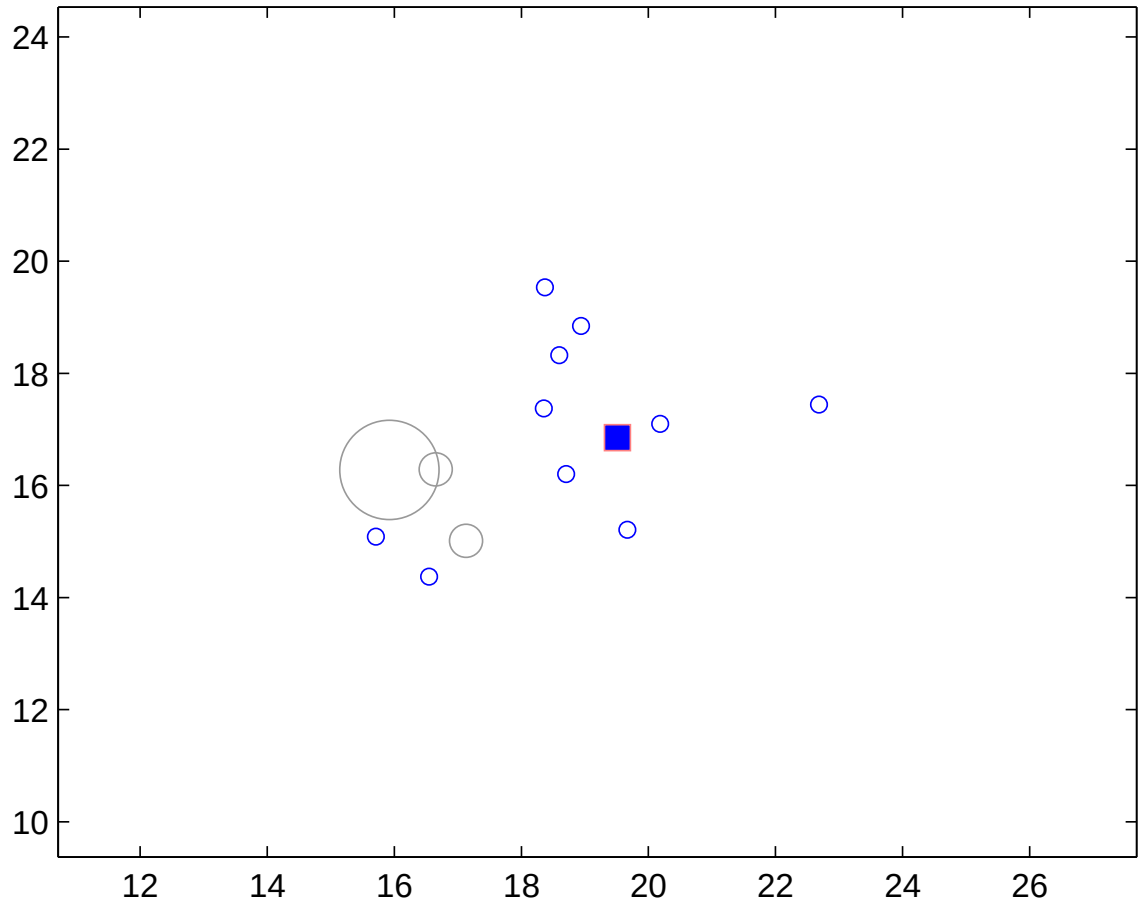
Transition demos

Sample $x_t^i \sim p(x_t|x_{t-1}^{i^j}) : \forall i^j = 3$



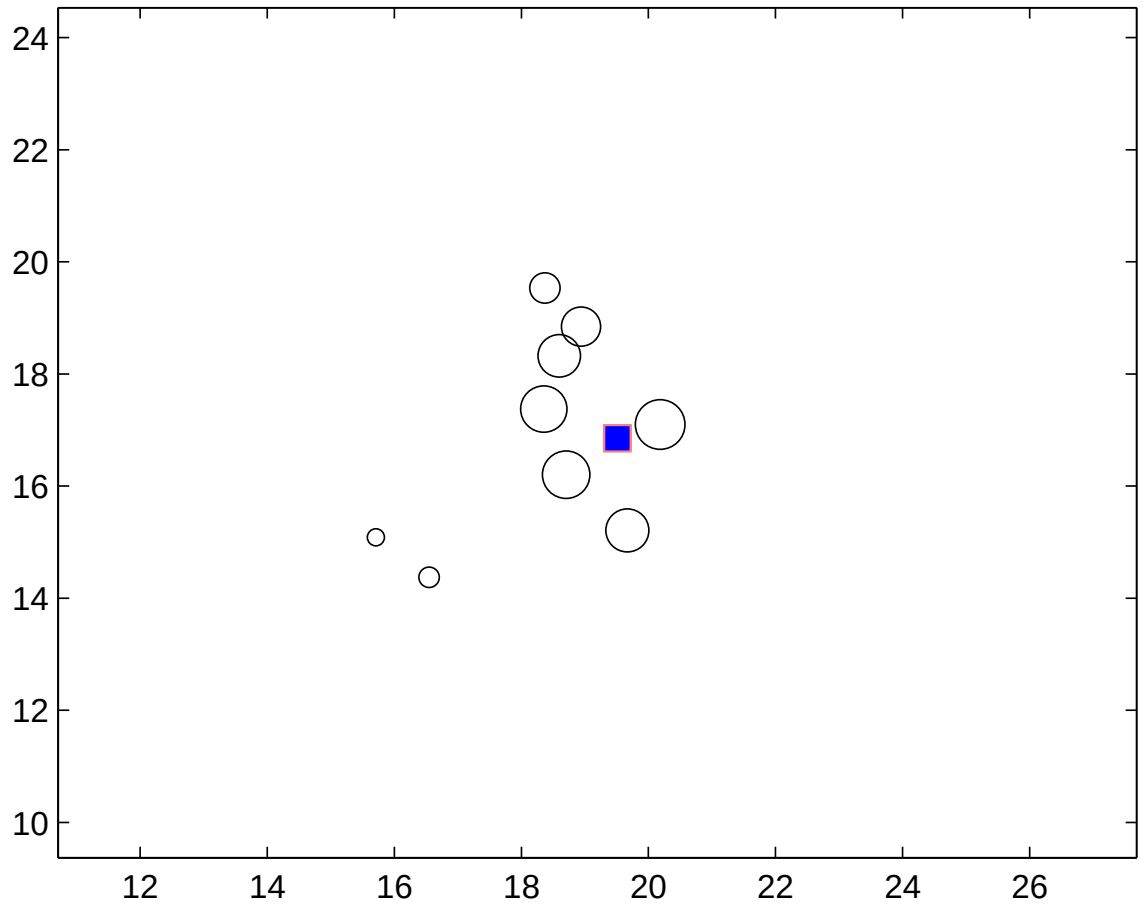
Transition demos

Here are the complete $\{x_t^i\}_1^N$ sampled.



After re-weighting

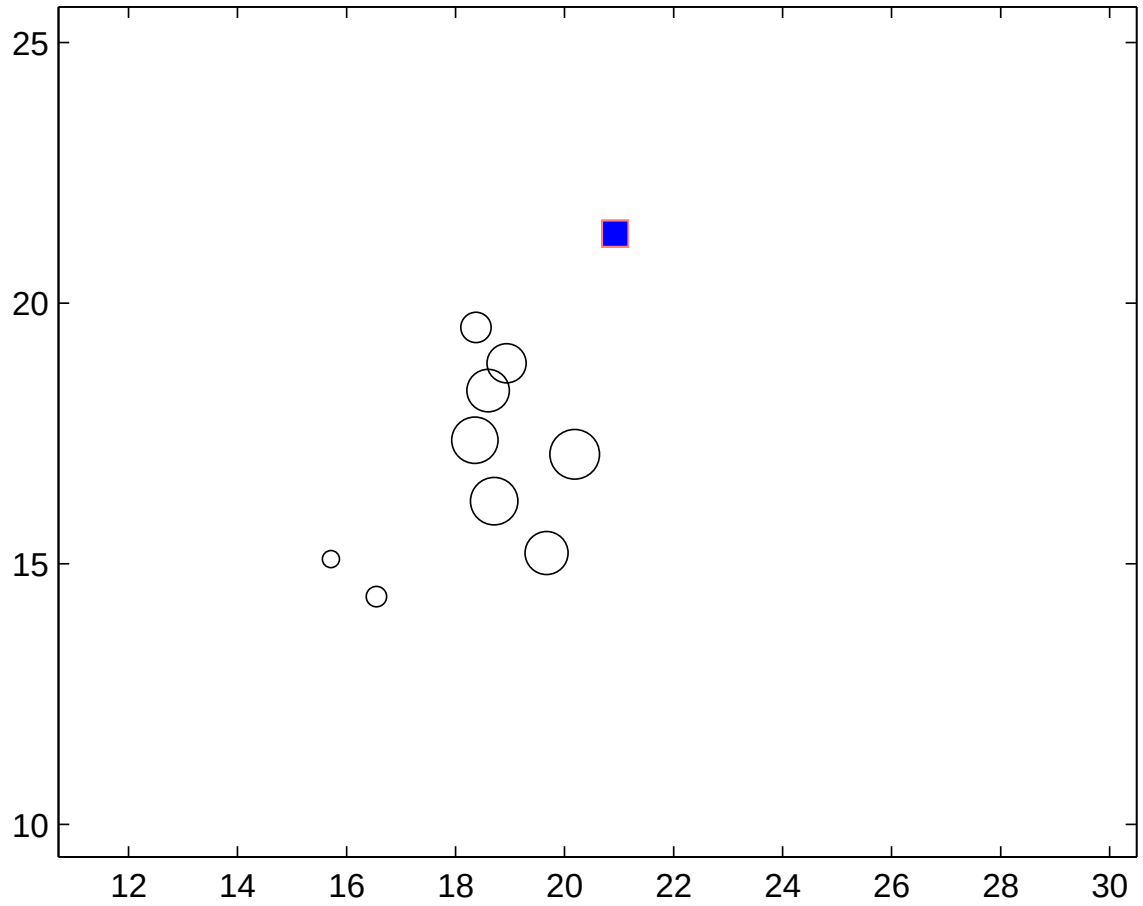
Compute the weights $w_t^i \propto p(y_t|x_t^i)$:



The above is the representation for $p(x_t|y_{1:t})$ Note that weights are in log

Next t

So the recursion will repeat:



The above is the representation for $p(x_{t-1}|y_{1:t-1})$ in the next t:

Some cool things you can do just with Bootstrap Filter

For example, A Coupled two-states dynamic model:
To estimate $p(x^1_{1:t}, x^2_{1:t} | y^1_{1:t}, y^2_{1:t})$

$$
\begin{aligned}
& w^i_t(x^1_{1:t}, x^2_{1:t}) \propto \\
& = \frac{g_1(y^1_t|x^1_t) g_2(y^2_t|x^2_t) f_1(x^1_t|x^1_{t-1}, x^2_{t-1}) f_2(x^2_t|x^1_{t-1}, x^2_{t-1})}{q^1(x^1_t|y^1_t, x^1_{t-1}, x^2_{t-1}) q^2(x^2_t|y^2_t, x^1_{t-1}, x^2_{t-1})} \\
& \quad w^i_{t-1}(x^1_{1:t-1}, x^2_{1:t-1})
\end{aligned} \tag{12}
$$

Sampler for Coupled dynamic model

(leaving out the case of $t = 1$, and re-sampling step)

At time t:

Sample $x_t^{1,(i)} \sim f_1(x_t^1 | x_{t-1}^{1,(i)}, x_{t-1}^{2,(i)})$

Sample $x_t^{2,(i)} \sim f_2(x_t^2 | x_{t-1}^{1,(i)}, x_{t-1}^{2,(i)})$

Compute the weights $w_t^{1,(i)} \propto \pi_{t-1}^{1,(i)} g_1(y_t^{1,(i)} | x_t^{1,(i)})$ (13)

Compute the normalized weights $\pi_t^{1,(i)}$

Compute the weights $w_t^{2,(i)} \propto \pi_{t-1}^{2,(i)} g_2(y_t^{2,(i)} | x_t^{2,(i)})$

Compute the normalized weights $\pi_t^{2,(i)}$

Auxiliary Particle Filter

- **idea**: Let y_t also participates in the proposal.
- **how**: In bootstrap sampling, x_t^i is more likely to be generated from $x_{t-1}^{i^j}$ when the value of $\pi_{t-1}^{i^j}$ is high. **Then** , how about let's also give preference to those $x_{t-1}^{i^j}$ where their proposed $x^i \sim x_{t-1}^{i^j}$ can be weighted higher by $p(y_t|x^i)$ as well?
- **in my word**: Have a bit of scouting before sampling!

Auxiliary Particle Filter algorithm

$$\mu_t^i = \mathrm{E}_{x_t}[x_t|x_{t-1}^i], \text{ OR: } \mu_t^i \sim p(x_t|x_{t-1}^i) \tag{14}$$

At time t, for each particle i:

Calculate μ_t^i

Compute the weights $w_t^i \propto p(y_t|\mu_t^i)\pi_{t-1}^i$

Normalize w_t^i

Sample $i^j \sim \{w_t^i\}$ (15)

Sample $x_t^i \sim p(x_t|x_{t-1}^{i^j})$

Assign $w_t^i \propto \dfrac{p(y_t|x_t^i)}{p(y_t|\mu_t^{i_j})}$

Normalize $w_t^i \to \pi_t^i$

Why $w_t^i \propto \frac{p(y_t|x_t^i)}{p(y_t|\mu_t^{i_j})}$? The proposal

Looking at the proposal:

$$q(x_t^i, i^j|.) = \underbrace{q(x_t^i|i^j, x_{t-1}, y_{1:t})}_{\text{2: choose } x_t} \underbrace{q(i^j|x_{t-1}, y_{1:t})}_{\text{1: choose the index}} \tag{16}$$

From the algorithm of the previous page:

$$\begin{aligned} &\text{1st Step: choose the index: } q(i^j|x_{t-1}, y_{1:t}) \propto p(y_t|\mu_t^{i^j})\pi_{t-1}^{i^j} \\ &\text{2nd Step: choose the } x_t\text{: } q(x_t^i|i^j, x_{t-1}, y_{1:t}) \equiv p(x_t^i|x_{t-1}^{i^j}) \end{aligned} \tag{17}$$

Why $w_t^i \propto \frac{p(y_t|x_t^i)}{p(y_t|\mu_t^{i_j})}$?

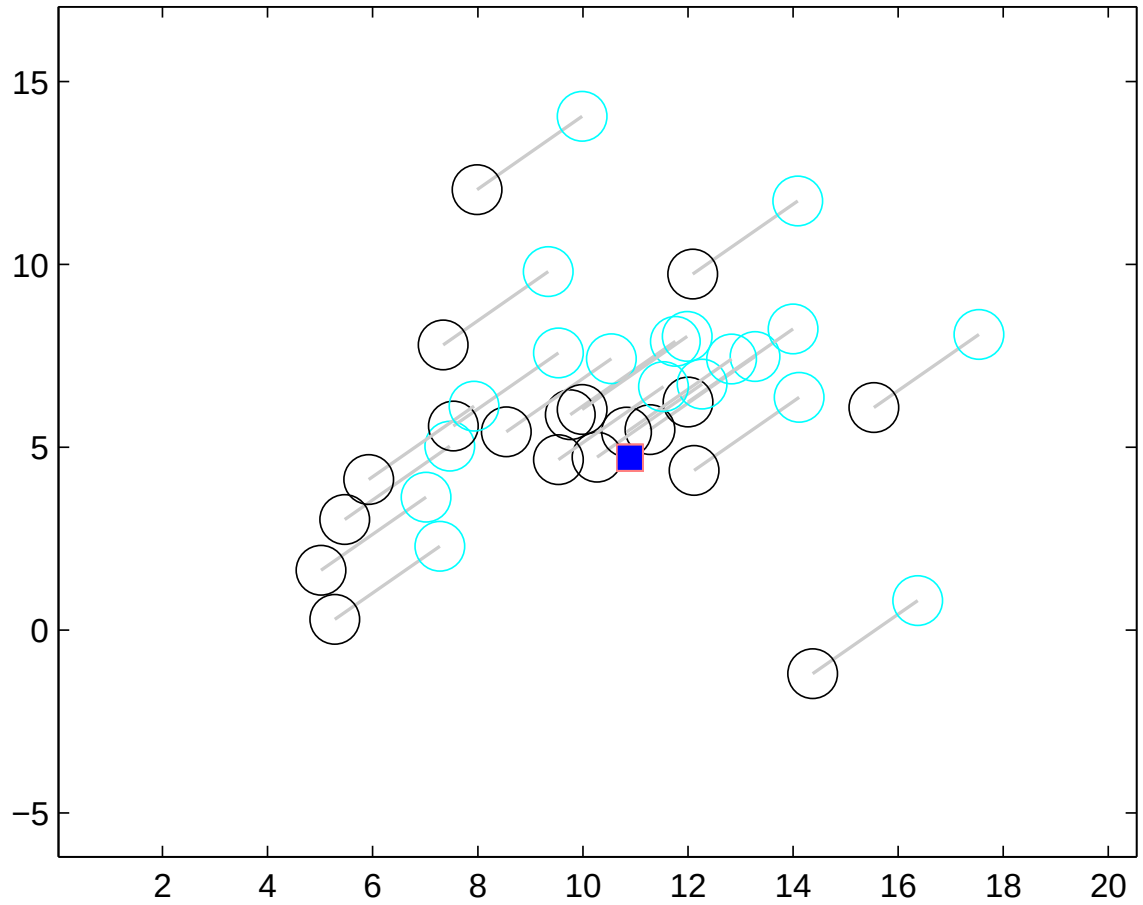
Substituting N of the (x^i, i^j) into the following:

$$w_t(x_{1:t}) \propto \pi_{t-1}(x_{1:t-1}) \times \frac{p(y_t|x_t)p(x_t|x_{t-1})}{q(x_t|x_{t-1}, y_t)}$$

$$w_t^i(x_{1:t}) \propto \pi_{t-1}^{i^j} \times \frac{p(y_t|x_t^i)p(x_t|x_{t-1}^{i^j})}{p(y_t|\mu_t^{i^j})\pi_{t-1}^{i^j}p(x_t^i|x_{t-1}^{i^j})}$$

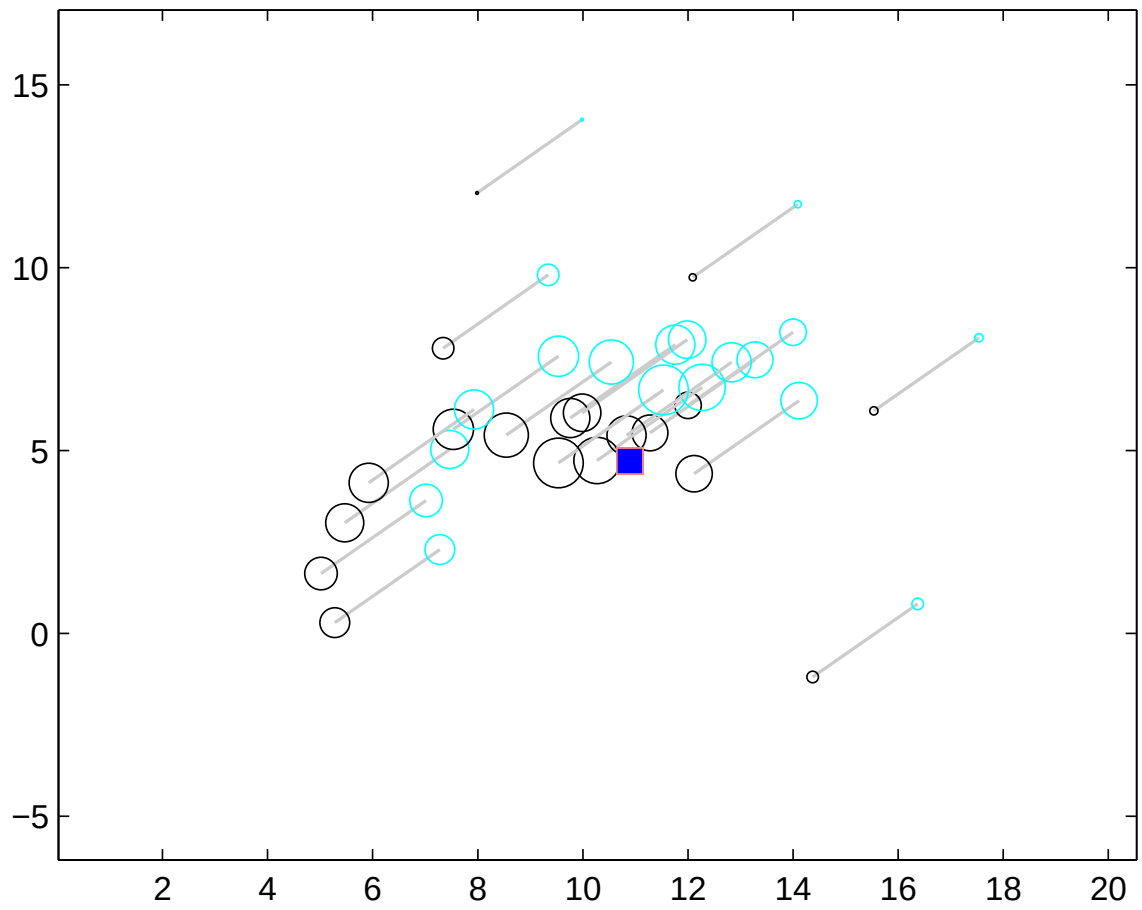
$$= \frac{p(y_t|x_t^i)}{p(y_t|\mu_t^{i^j})}$$

Representation for $p(x_{t-1}|y_{1:t-1})$ and μ_t^i

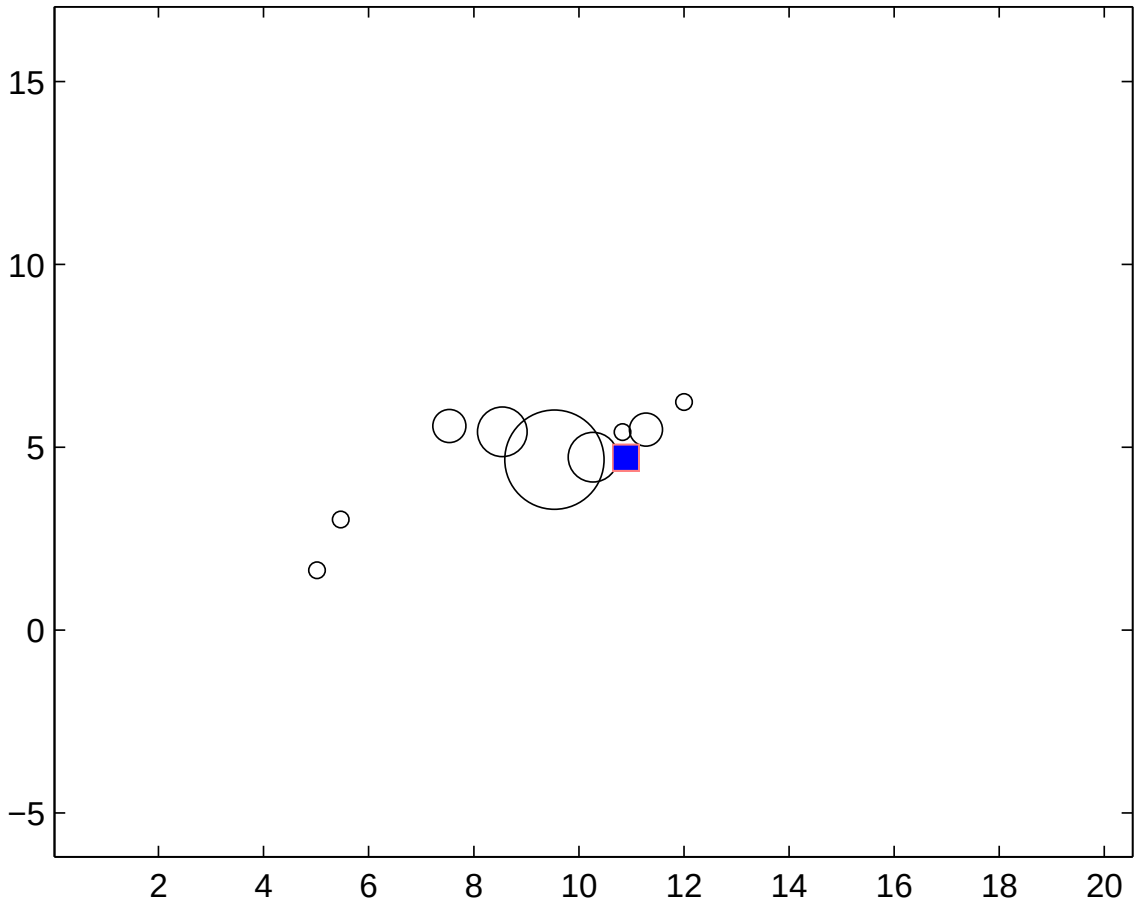


- Light blue circles are μ_t^i for each x_{t-1}^i

New weights: $\propto p(y_t|\mu_t^i)\pi_{t-1}^i$

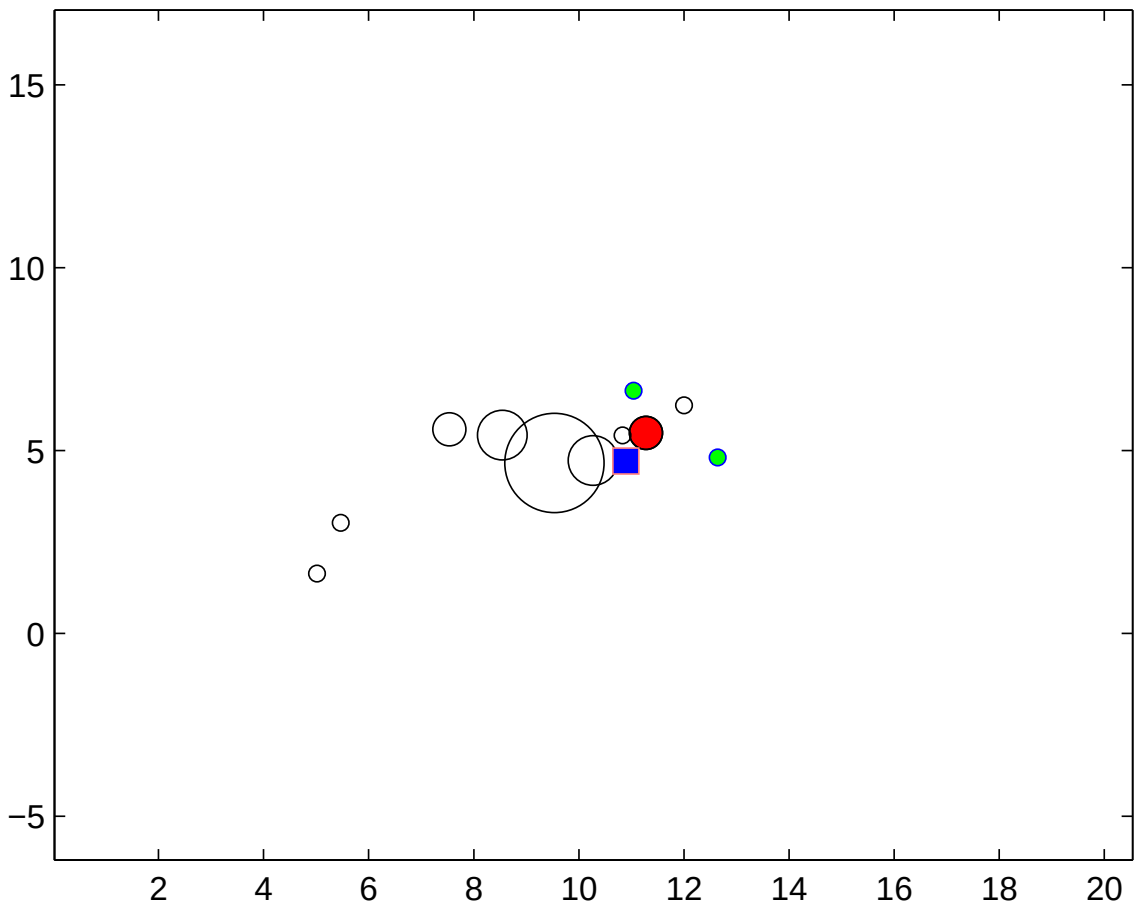


Re-sampling

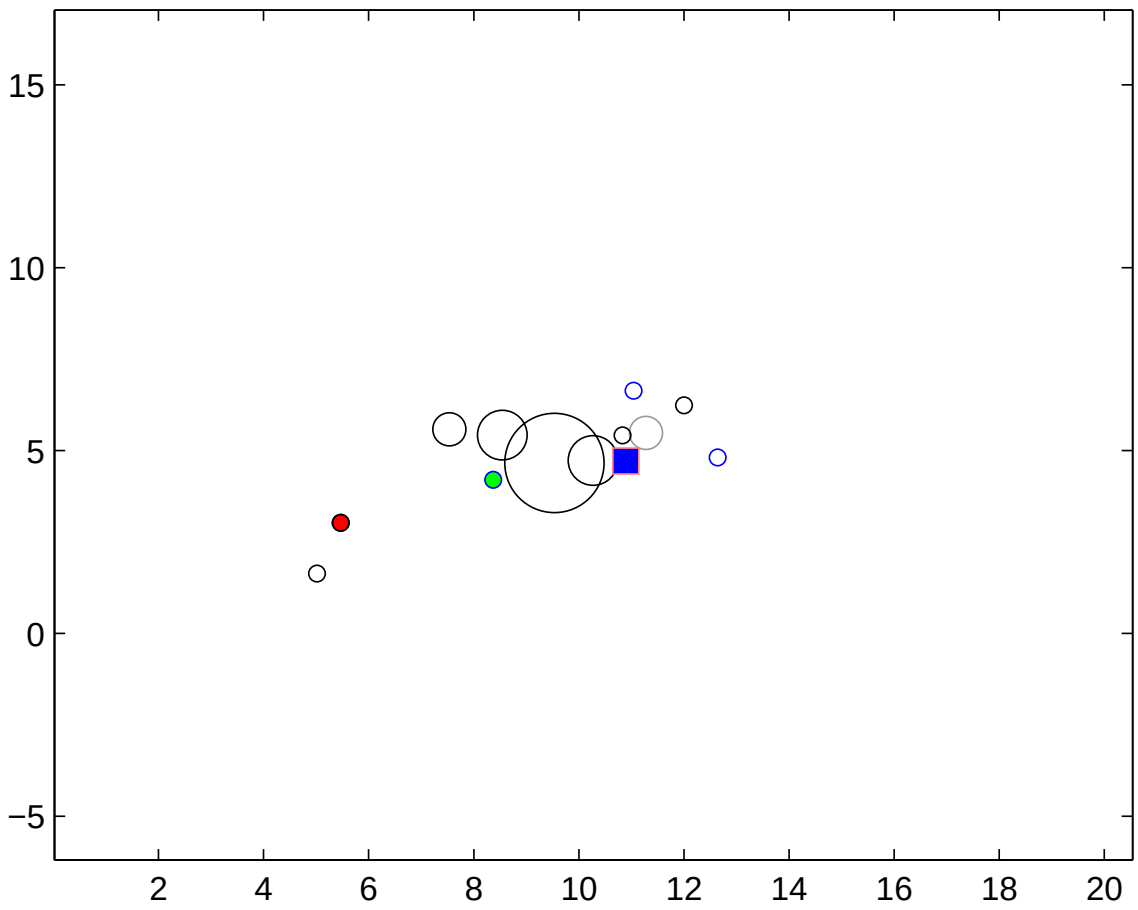


- Size of the circle indicates the number of times x_{t-1}^{ij} was selected.

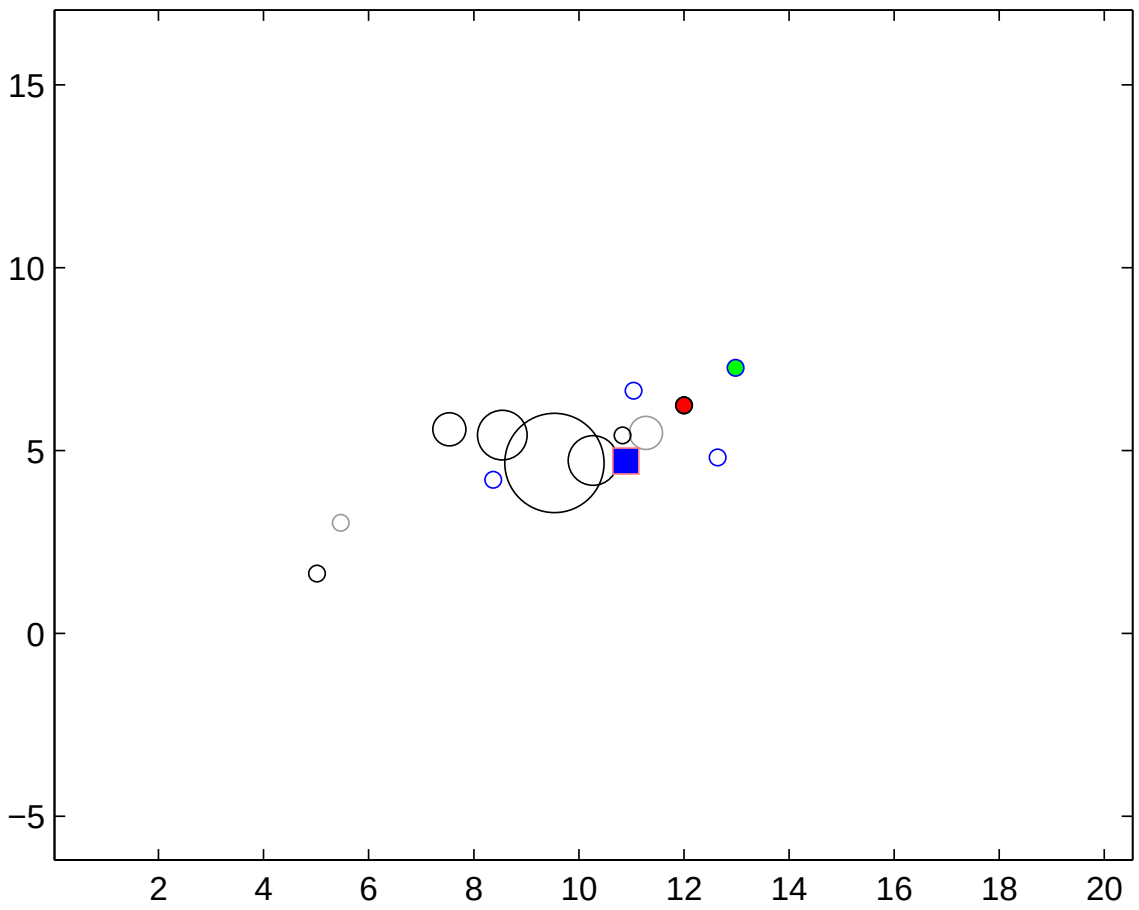
Transition demos



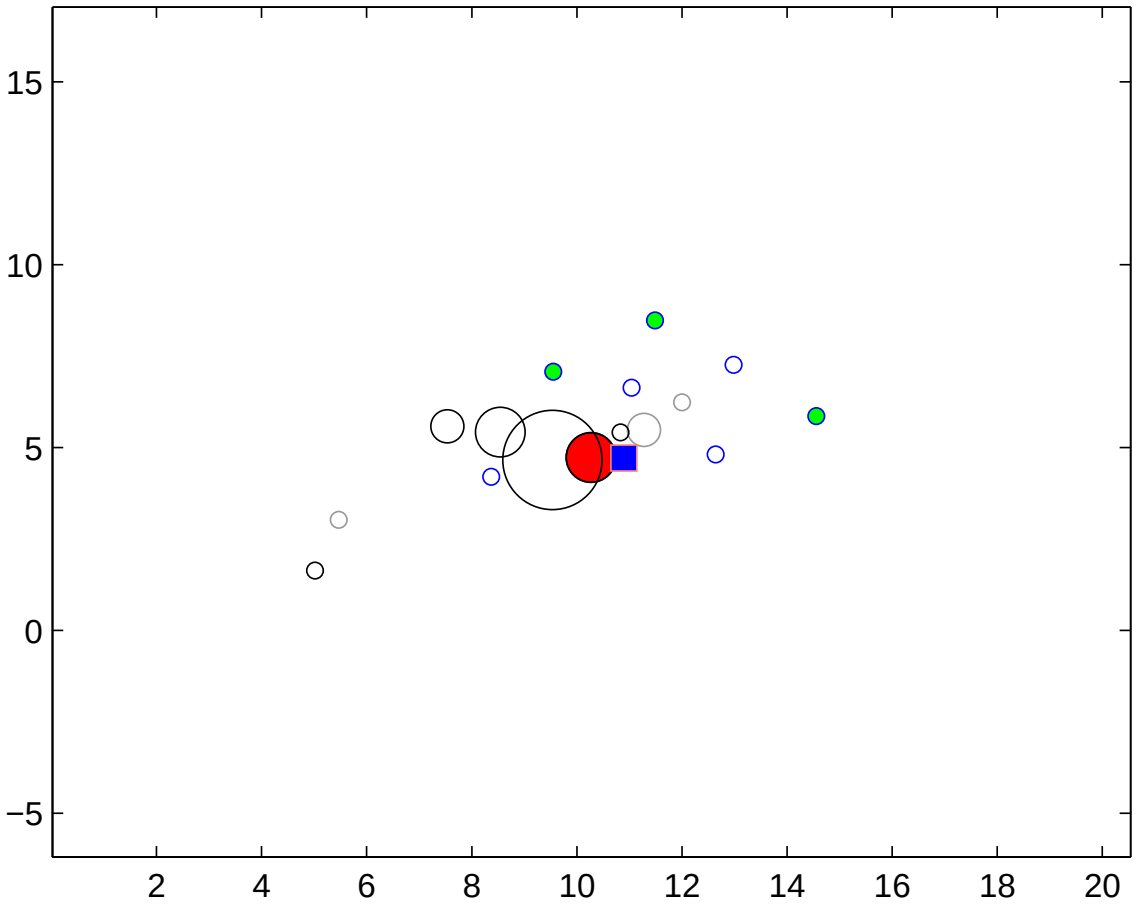
Transition demos



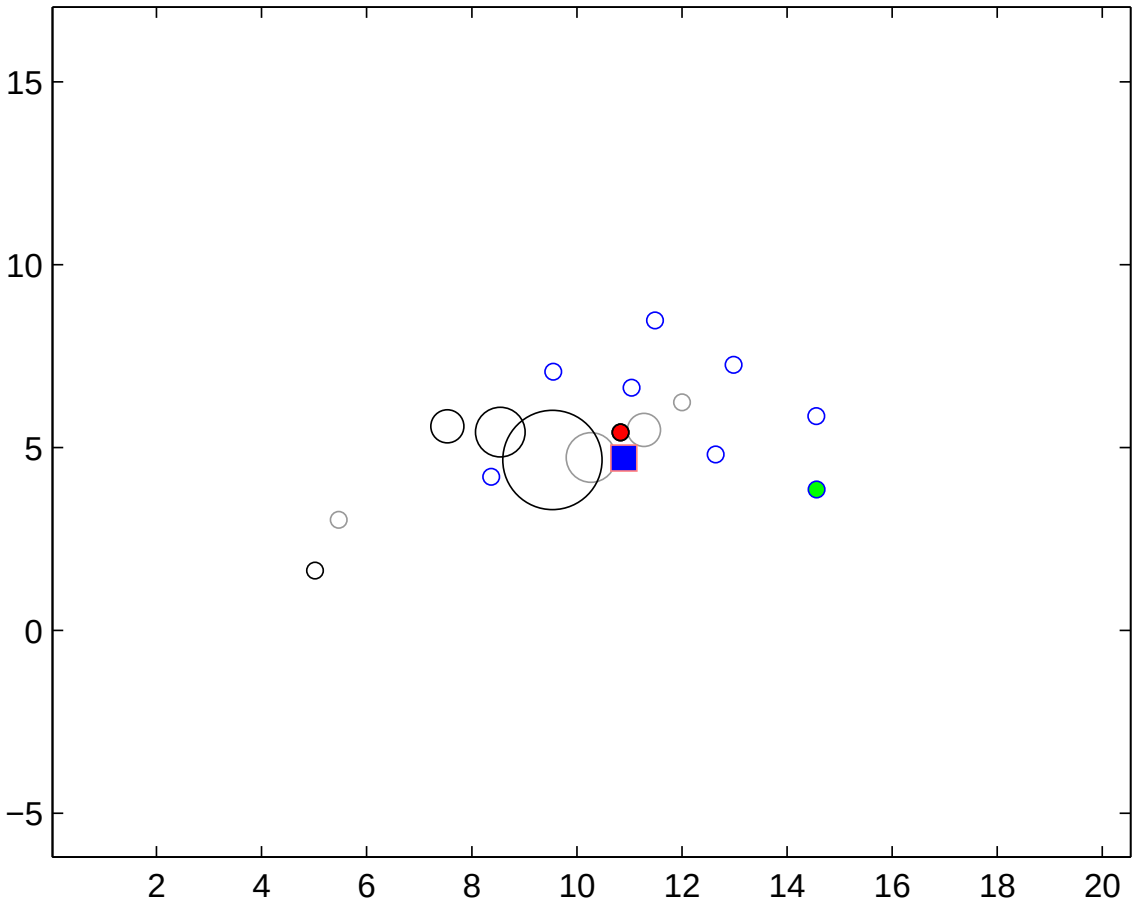
Transition demos



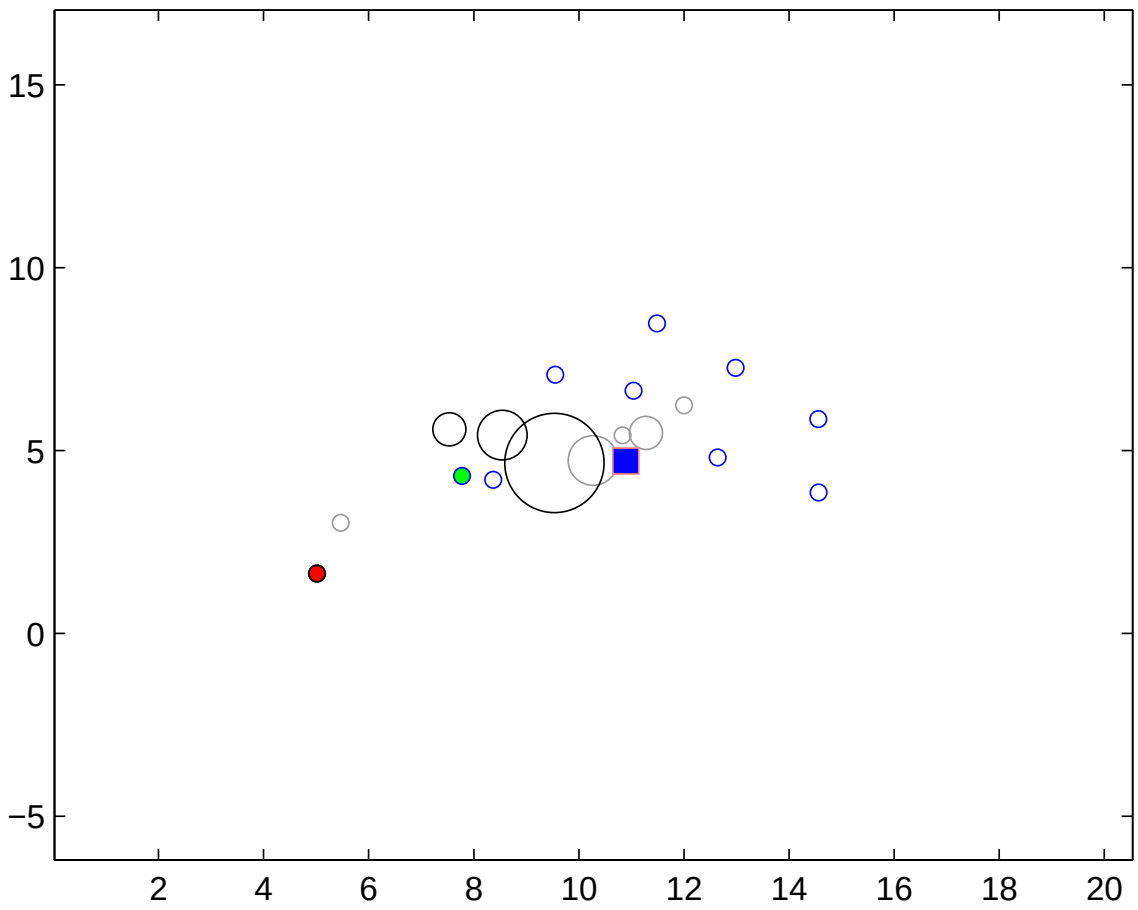
Transition demos



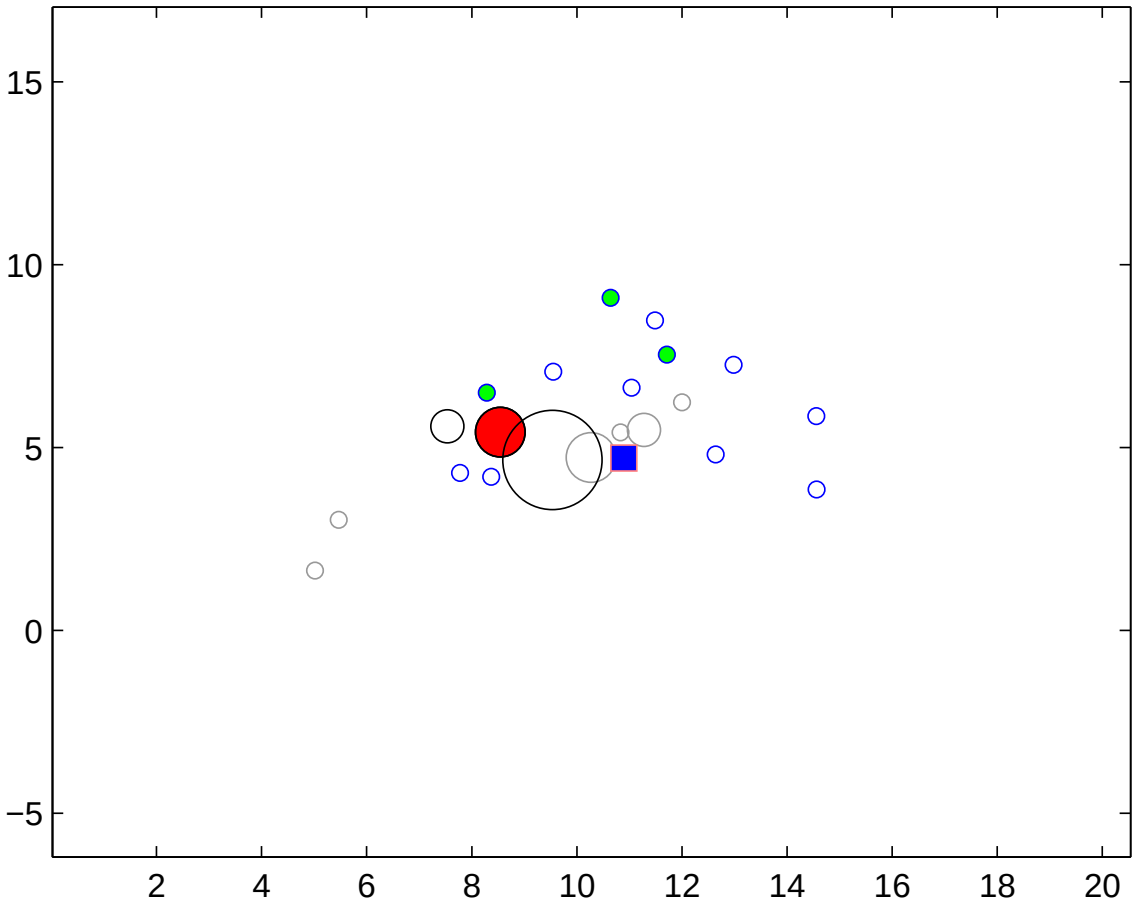
Transition demos



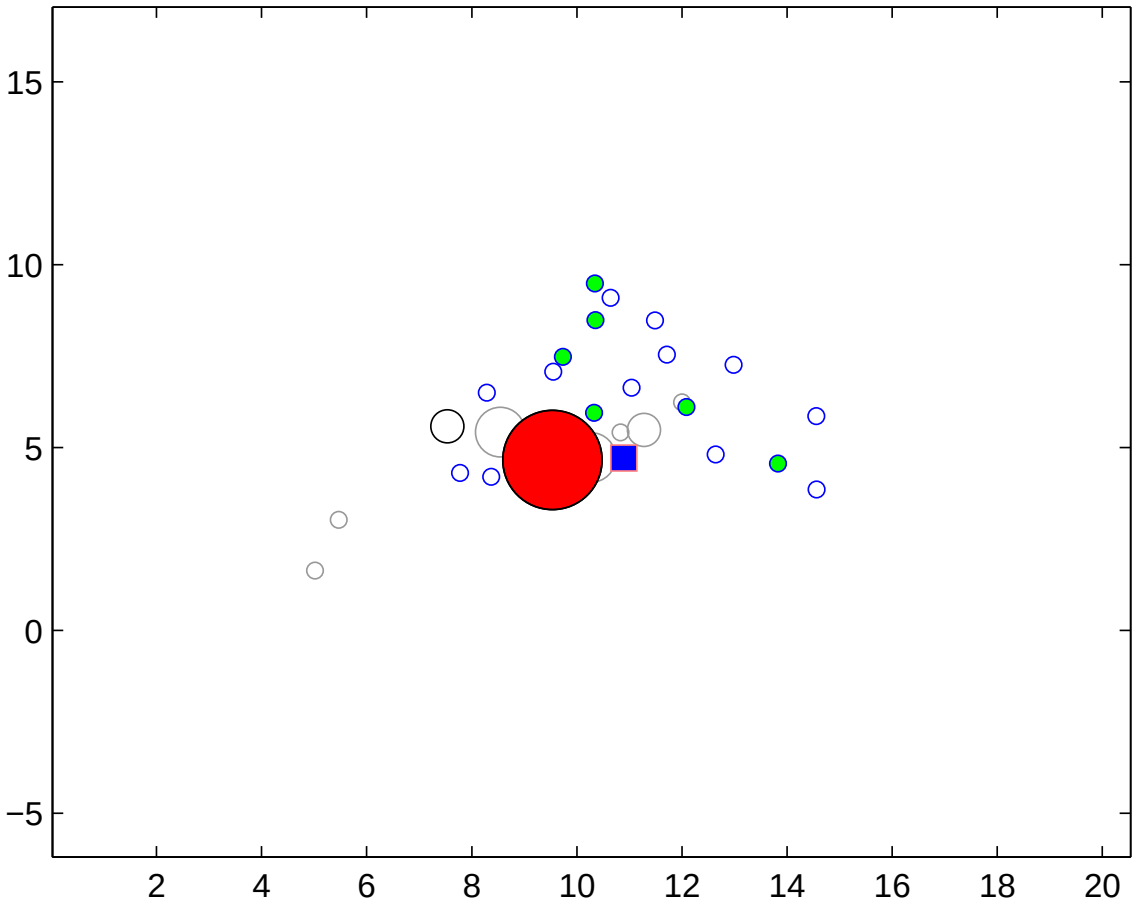
Transition demos



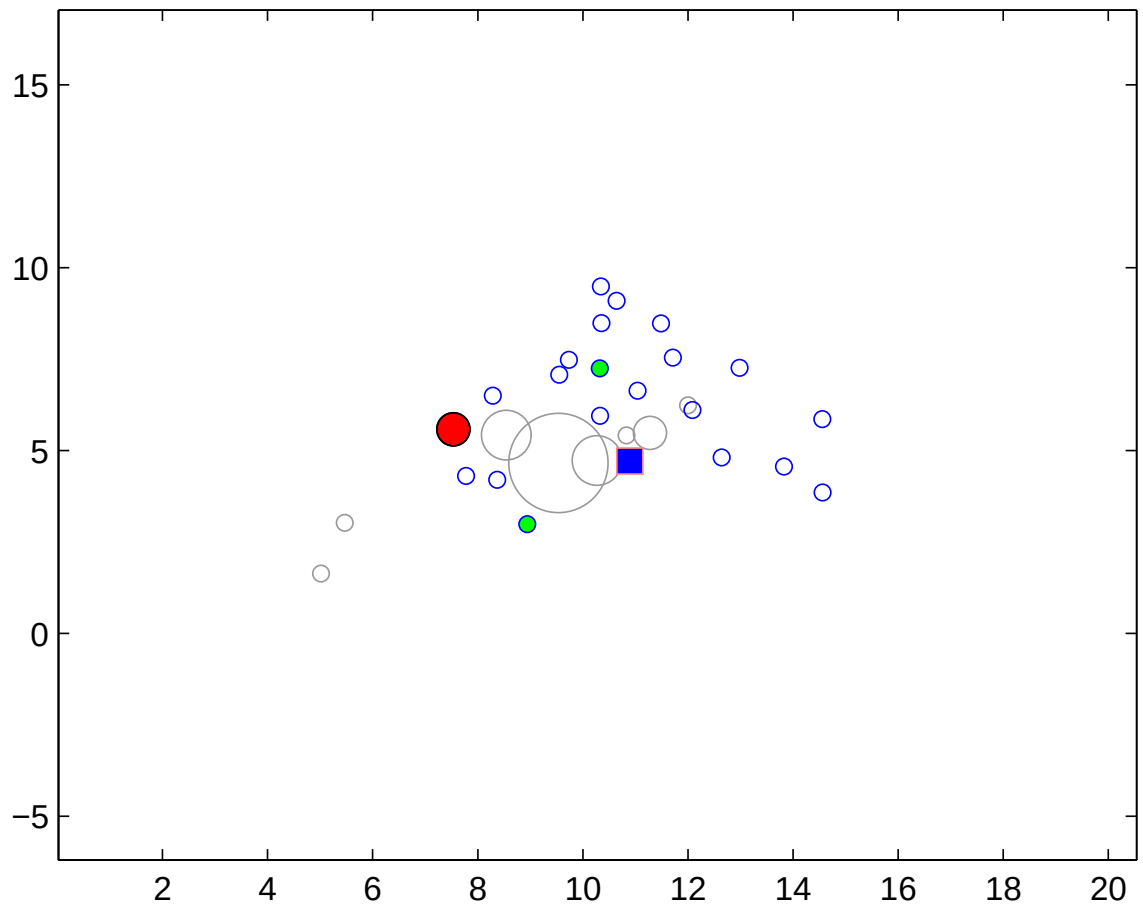
Transition demos



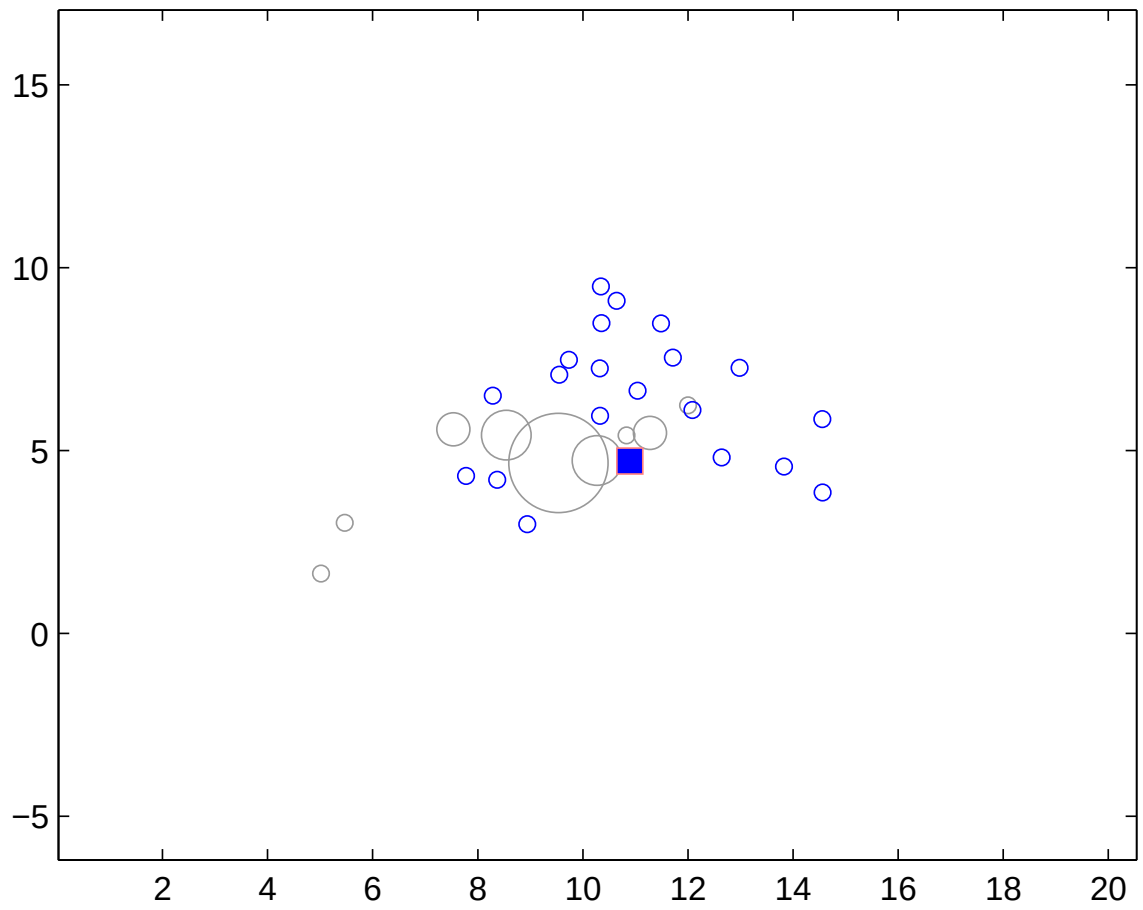
Transition demos



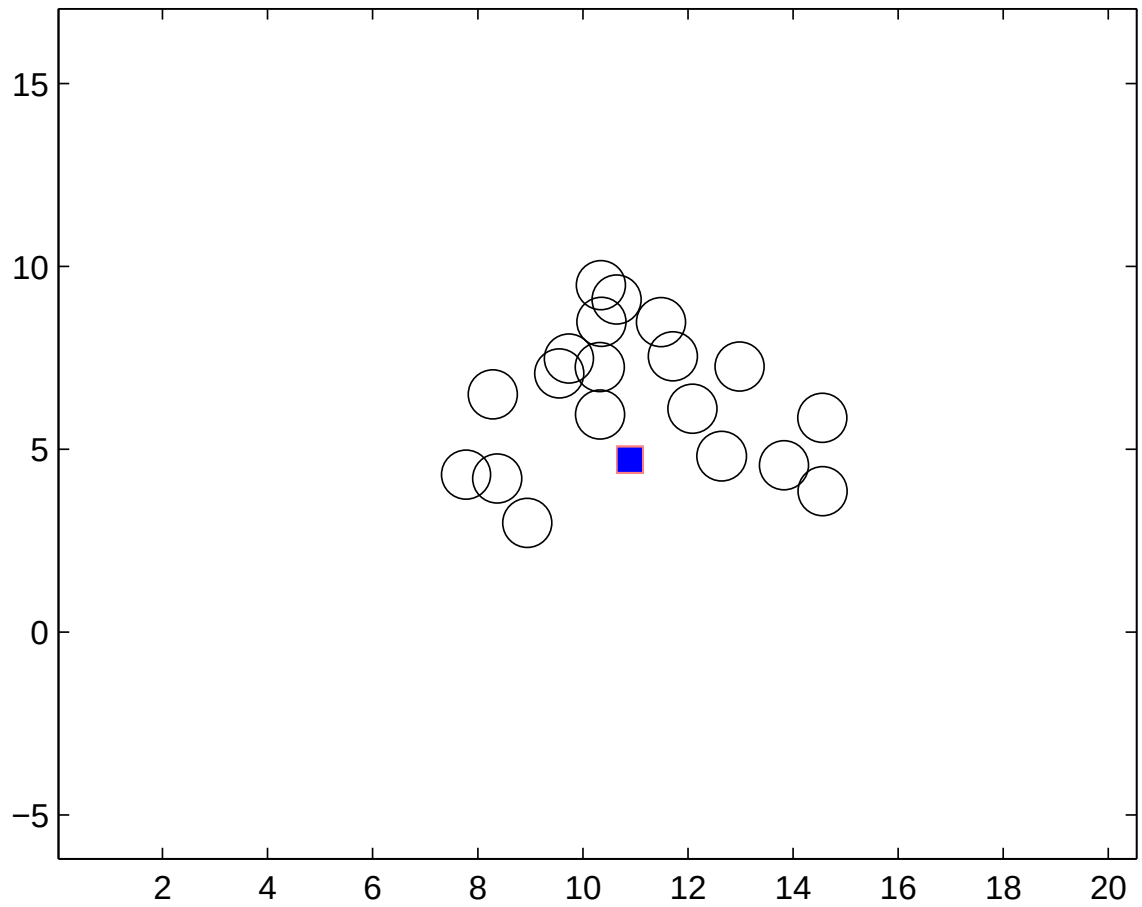
Transition demos



Transition demos

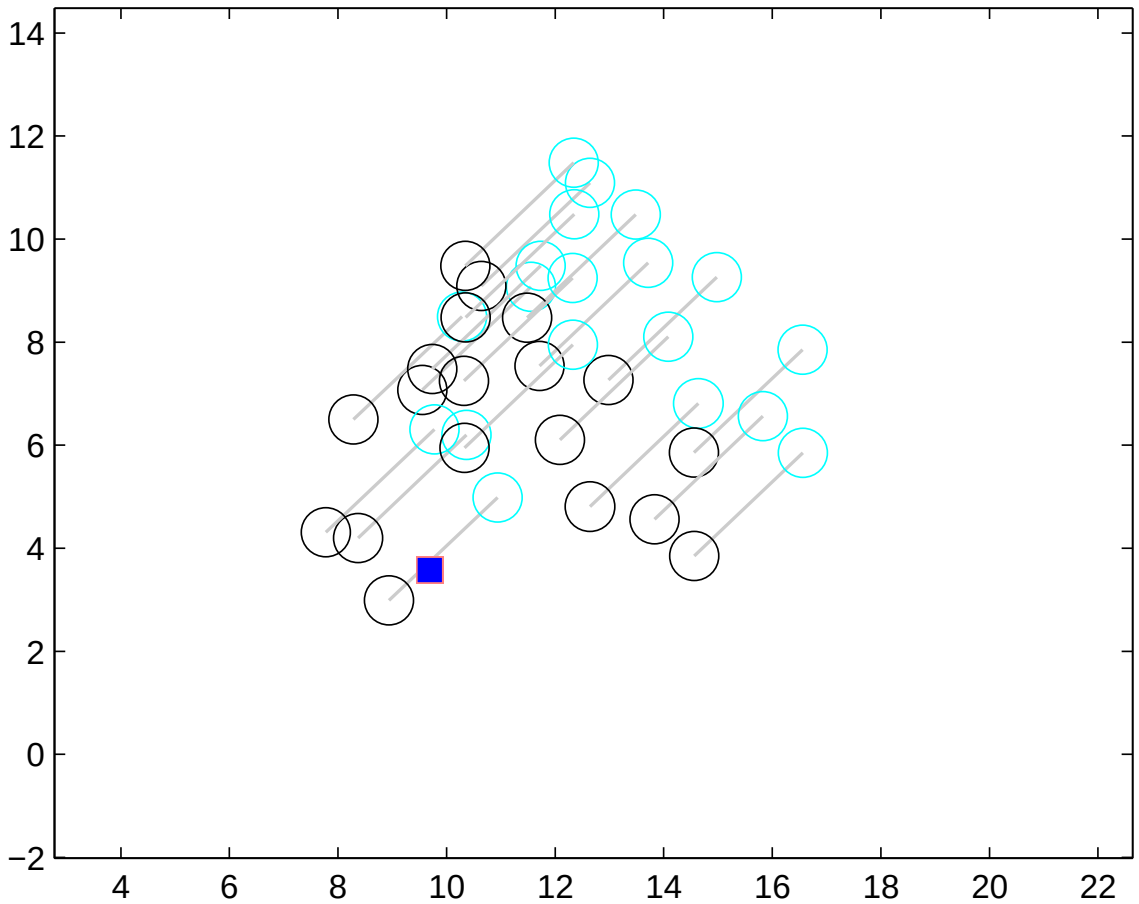


After re-weighting



The above is the representation for $p(x_t|y_{1:t})$ Note that weights are in log scale..

Next t



The above is the representation for $p(x_{t-1}|y_{1:t-1})$ in the next t:

References and a set of good place to study sampling

- Christopher Bishop's textbook Pattern Recognition and Machine Learning - include a whole chapter on sampling
- The BUGS project: (http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml)
- For PhD student, Gary Walsh has a great lecture notes on MCMC tutorial, very gentle, called "Markov Chain Monte Carlo and Gibbs Sampling Lecture Notes for EEB 581"
- For SMC stuff, see Doucet and Johansen, "A Tutorial on Particle Filtering and Smoothing: Fifteen years later"
- Arulampalam, M.S. and Maskell, S. and Gordon, N. and Clapp, T, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, IEEE Transactions on Signal Processing, 2002
- Pitt, M.K.; Shephard, N. (1999). "Filtering Via Simulation: Auxiliary Particle Filters". Journal of the American Statistical Association (American Statistical Association) 94 (446): 590591