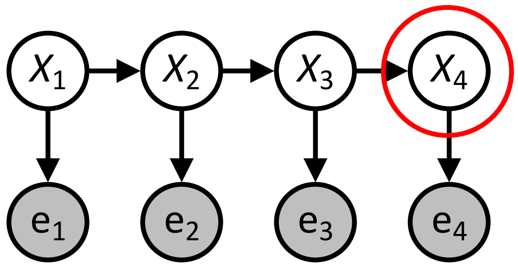
Filtering algorithm

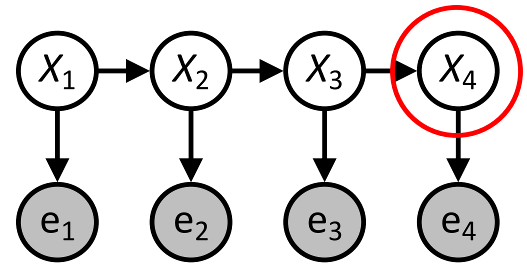
- Aim: devise a ***recursive filtering*** algorithm of the form
  - $P(X_{t+1} | e_{1:t+1}) = g(e_{t+1}, P(X_t | e_{1:t}))$



- $P(X_{t+1} | e_{1:t+1}) = P(X_{t+1} | e_{1:t}, e_{t+1})$
- $= \alpha\, P(e_{t+1} | X_{t+1}, e_{1:t})\, P(X_{t+1} | e_{1:t})$
- $= \alpha\, P(e_{t+1} | X_{t+1})\, P(X_{t+1} | e_{1:t})$
- $= \alpha\, P(e_{t+1} | X_{t+1}) \sum_{x_t} P(x_t | e_{1:t})\, P(X_{t+1} | x_t, e_{1:t})$
- $= \alpha\, \underbrace{P(e_{t+1} | X_{t+1})}_{\text{Given by HMM}} \sum_{x_t} \underbrace{P(x_t | e_{1:t})}_{\text{Pre-computed}}\, \underbrace{P(X_{t+1} | x_t)}_{\text{Given by HMM}}$

Filtering algorithm

- Aim: devise a ***recursive filtering*** algorithm of the form
  - $P(X_{t+1} | e_{1:t+1}) = g(e_{t+1}, P(X_t | e_{1:t}))$



- $P(X_{t+1} | e_{1:t+1}) = P(X_{t+1} | e_{1:t}, e_{t+1})$
- $= \alpha\, P(e_{t+1} | X_{t+1}, e_{1:t})\, P(X_{t+1} | e_{1:t})$

LHS: $P(X_{t+1}, e_{1:t}, e_{t+1}) / P(e_{1:t}, e_{t+1})$

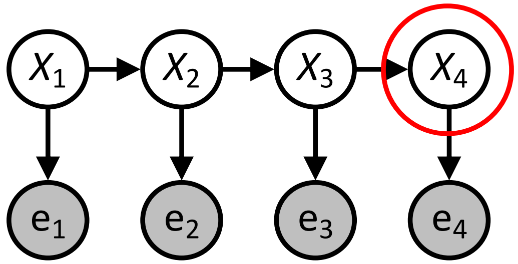
RHS: $\alpha\, P(e_{t+1}, X_{t+1}, e_{1:t}) / \cancel{P(X_{t+1}, e_{1:t})} * \cancel{P(X_{t+1}, e_{1:t})} / P(e_{1:t})$

RHS: $\alpha\, P(e_{t+1}, X_{t+1}, e_{1:t}) / P(e_{1:t})$

$\alpha = P(e_{1:t}) / P(e_{1:t}, e_{t+1})$ which is the same for all x_{t+1}

Filtering algorithm

- Aim: devise a ***recursive filtering*** algorithm of the form
  - $P(X_{t+1} | e_{1:t+1}) = g(e_{t+1}, P(X_t | e_{1:t}))$



- $P(X_{t+1} | e_{1:t+1}) = P(X_{t+1} | e_{1:t}, e_{t+1})$
- $= \alpha\, P(e_{t+1} | X_{t+1}, e_{1:t})\, P(X_{t+1} | e_{1:t})$
- $= \alpha\, P(e_{t+1} | X_{t+1})\, P(X_{t+1} | e_{1:t})$

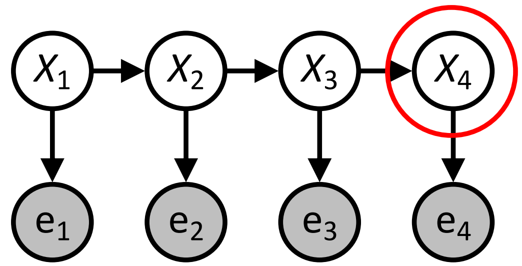
Why does $P(e_{t+1} | X_{t+1}, e_{1:t}) = P(e_{t+1} | X_{t+1})$?

Variables are independent of non-descendants given parents

If I know X_4, nothing else will help be better predict e_4

Filtering algorithm

- Aim: devise a ***recursive filtering*** algorithm of the form
  - $P(X_{t+1} | e_{1:t+1}) = g(e_{t+1}, P(X_t | e_{1:t}))$



- $P(X_{t+1} | e_{1:t+1}) = P(X_{t+1} | e_{1:t}, e_{t+1})$
- $= \alpha\, P(e_{t+1} | X_{t+1}, e_{1:t})\, P(X_{t+1} | e_{1:t})$
- $= \alpha\, P(e_{t+1} | X_{t+1})\, P(X_{t+1} | e_{1:t})$
- $= \alpha\, P(e_{t+1} | X_{t+1}) \sum_{x_t} P(x_t \mid e_{1:t})\, P(X_{t+1} | x_t, e_{1:t})$

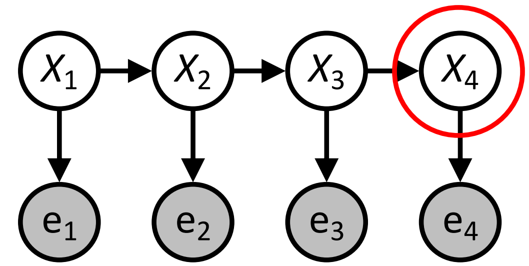
P(A|B)P(B) = P(A,B)

Marginalization over x_t

$$\sum_{x_t} P(X_{t+1} | x_t, e_{1:t})\, P(x_t \mid e_{1:t}) = \sum_{x_t} P(X_{t+1}, x_t \mid e_{1:t}) = P(X_{t+1} | e_{1:t})$$

Filtering algorithm

- Aim: devise a ***recursive filtering*** algorithm of the form
  - $P(X_{t+1} | e_{1:t+1}) = g(e_{t+1}, P(X_t | e_{1:t}))$

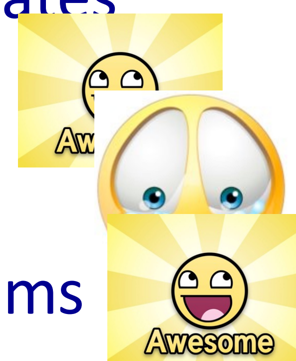


- $P(X_{t+1} | e_{1:t+1}) = P(X_{t+1} | e_{1:t}, e_{t+1})$
- $= \alpha\, P(e_{t+1} | X_{t+1}, e_{1:t})\, P(X_{t+1} | e_{1:t})$
- $= \alpha\, P(e_{t+1} | X_{t+1})\, P(X_{t+1} | e_{1:t})$
- $= \alpha\, P(e_{t+1} | X_{t+1}) \sum_{x_t} P(x_t | e_{1:t})\, P(X_{t+1} | x_t, e_{1:t})$
- $= \alpha\, \underbrace{P(e_{t+1} | X_{t+1})}_{\text{Given by HMM}} \sum_{x_t} \underbrace{P(x_t | e_{1:t})}_{\text{Pre-computed}}\, \underbrace{P(X_{t+1} | x_t)}_{\text{Given by HMM}}$

Variables are independent of non-descendants given parents

"Forward" algorithm

- $P(X_{t+1}|e_{1:t+1}) = \underbrace{\alpha}_{\text{Normalize}} \underbrace{\overbrace{P(e_{t+1}|X_{t+1})}^{\text{Given by HMM (vector)}}}_{\text{Update}} \underbrace{\sum_{x_t} \overbrace{P(x_t \mid e_{1:t})}^{\text{Pre-computed (scalar for each term in sum)}} \overbrace{P(X_{t+1} \mid x_t)}^{\text{Given by HMM (vector for each term in sum)}}}_{\text{Predict}}$
- $\boldsymbol{f}_{1:t+1} = \textsc{Forward}(\boldsymbol{f}_{1:t}, e_{t+1})$; $\boldsymbol{f}_{1:t}$ is $P(X_t|e_{1:t})$ *for t=0, note $e_{1:0}$ is empty
- Cost per time step: $O(|X|^2)$ where $|X|$ is the number of states
- Time and space costs are ***constant***, independent of t
- $O(|X|^2)$ is infeasible for models with many state variables
- We get to invent really cool approximate filtering algorithms



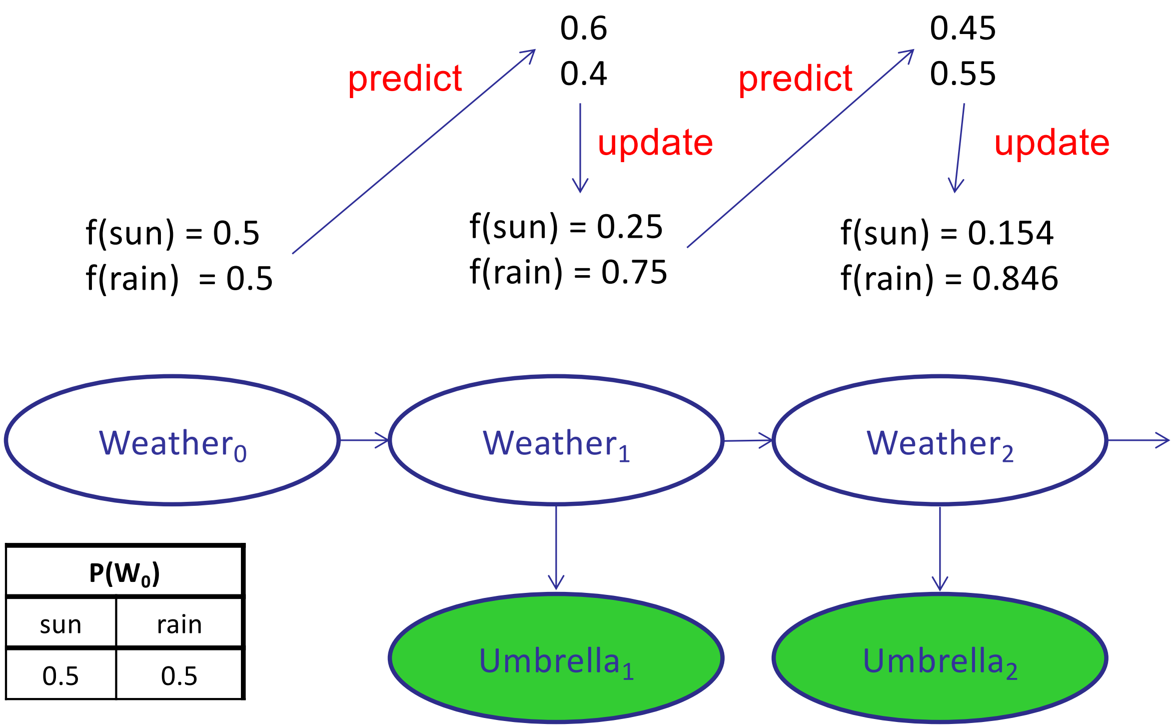
And the same thing in linear algebra

- Transition matrix T, observation matrix O_t
  - Observation matrix has state likelihoods for E_t along diagonal
  - E.g., for U_1 = true, $O_1 = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.9 \end{pmatrix}$
- Filtering algorithm becomes
  - $\boldsymbol{f}_{1:t+1} = \alpha\, O_{t+1} T^{\mathrm{T}} \boldsymbol{f}_{1:t}$

X_{t-1}	$P(X_t \mid X_{t-1})$	
	sun	rain
sun	0.9	0.1
rain	0.3	0.7

W_t	$P(U_t \mid W_t)$	
	true	false
sun	0.2	0.8
rain	0.9	0.1

Example: Weather HMM

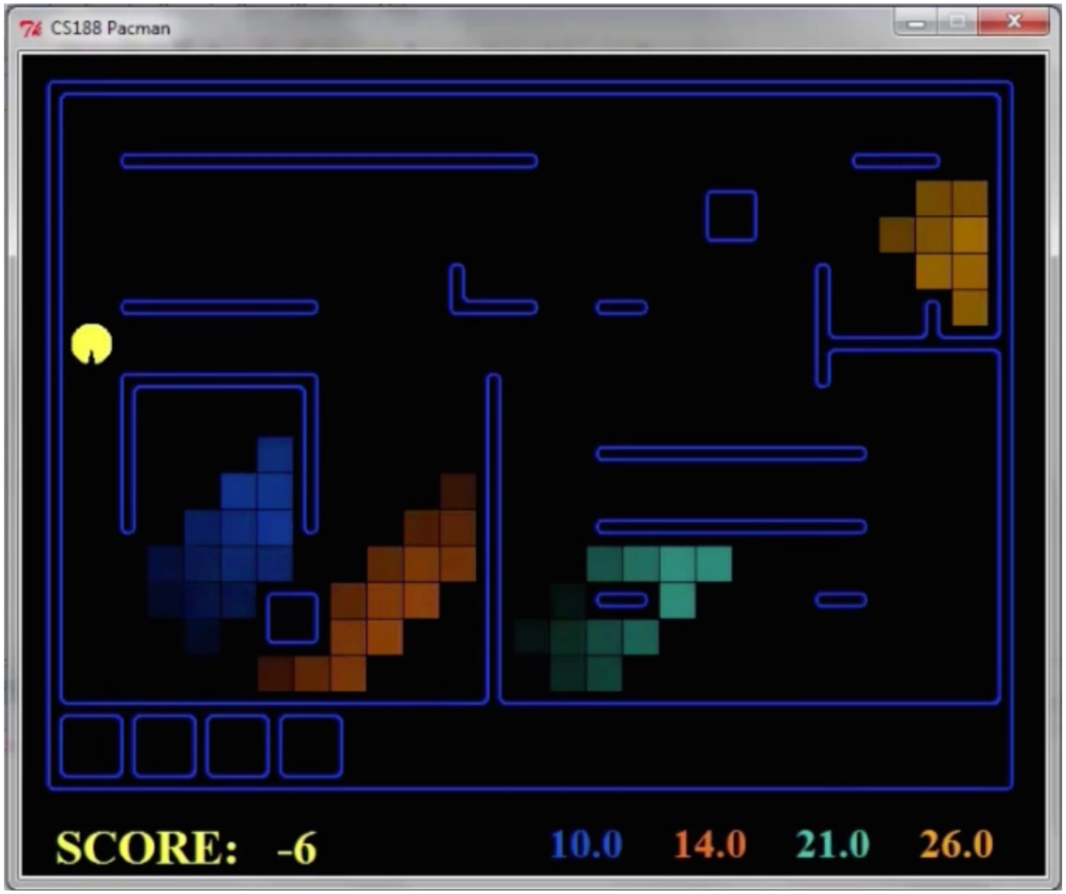


P(W_0)	
sun	rain
0.5	0.5

W_{t-1}	P(W_t | W_{t-1})	
	sun	rain
sun	0.9	0.1
rain	0.3	0.7

W_t	P(U_t | W_t)	
	true	false
sun	0.2	0.8
rain	0.9	0.1

Pacman – Hunting Invisible Ghosts with Sonar



[Demo: Pacman – Sonar – No Beliefs(L14D1)]

Video of Demo Pacman – Sonar

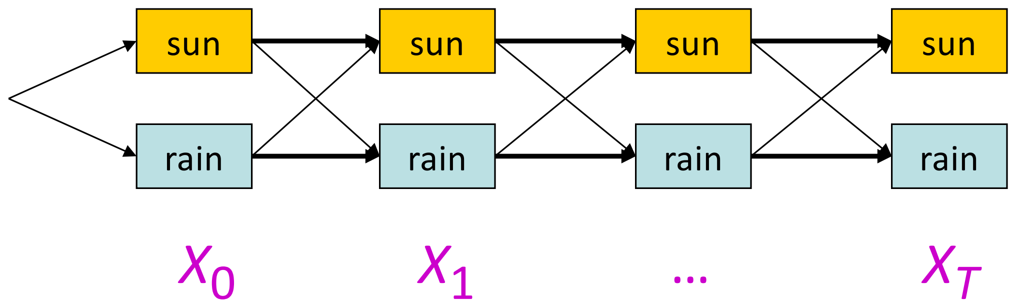
Most Likely Explanation

Inference tasks

- ***Filtering***: $P(X_t | e_{1:t})$
  - ***belief state***—input to the decision process of a rational agent
- ***Prediction***: $P(X_{t+k} | e_{1:t})$ for $k > 0$
  - evaluation of possible action sequences; like filtering without the evidence
- ***Smoothing***: $P(X_k | e_{1:t})$ for $0 \le k < t$
  - better estimate of past states, essential for learning
- ***Most likely explanation***: $\arg\max_{x_{1:t}} P(x_{1:t} \mid e_{1:t})$
  - speech recognition, decoding with a noisy channel

Most likely explanation = most probable path

- ***State trellis***: graph of states and transitions over time



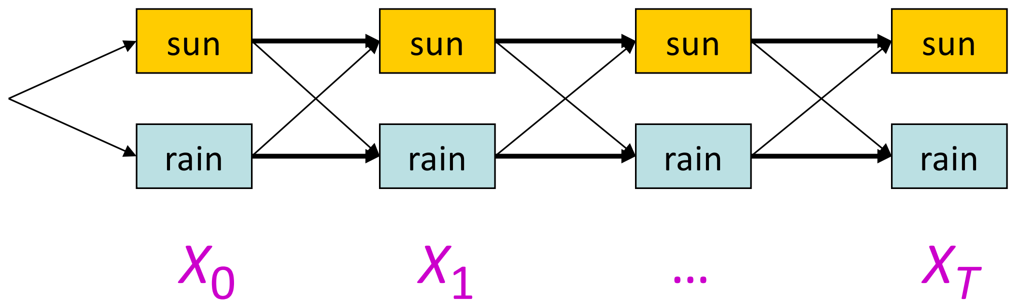
- arg $\max_{x_{1:t}} P(x_{1:t} \mid e_{1:t})$
- = arg $\max_{x_{1:t}} \alpha\, P(x_{1:t}, e_{1:t})$
- = arg $\max_{x_{1:t}} P(x_{1:t}, e_{1:t})$
- = arg $\max_{x_{1:t}} P(x_0) \prod_t P(x_t \mid x_{t-1}) P(e_t \mid x_t)$
- = arg $\max_{x_{1:t}} \log \left[P(x_0) \prod_t P(x_t \mid x_{t-1}) P(e_t \mid x_t) \right]$
- = arg $\min_{x_{1:t}} -\log P(x_0) + \sum_t -\log P(x_t \mid x_{t-1}) + -\log P(e_t \mid x_t)$

All given by HMM

Alternative form

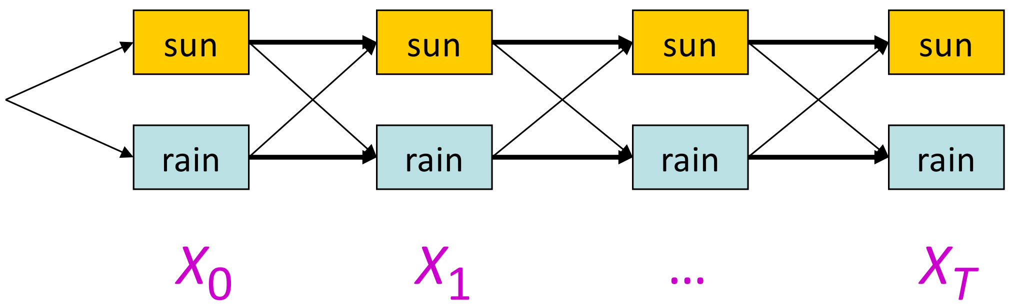
Most likely explanation = most probable path

- ***State trellis***: graph of states and transitions over time



- Each arc represents some transition $x_{t-1} \rightarrow x_t$
- Each arc has weight $P(x_t \mid x_{t-1})\, P(e_t \mid x_t)$ (arcs to initial states have weight $P(x_0)$)
- The ***product*** of weights on a path is proportional to that state sequence's probability
- Forward algorithm computes sums of paths, ***Viterbi algorithm*** computes best paths

Forward / Viterbi algorithms



Forward Algorithm (sum)

For each state at time t, keep track of the ***total probability of all paths*** to it

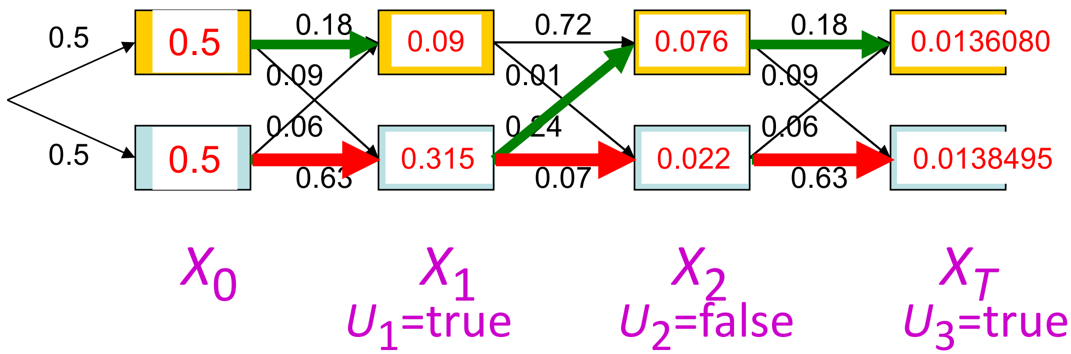
$$\mathbf{f}_{1:t+1} = \textsc{Forward}(\mathbf{f}_{1:t}, e_{t+1}) = \alpha\, P(e_{t+1}|X_{t+1}) \textstyle\sum_{x_t} P(X_{t+1}\,|\, x_t)\, \mathbf{f}_{1:t}$$

Viterbi Algorithm (max)

For each state at time t, keep track of the ***maximum probability of any path*** to it

$$\mathbf{m}_{1:t+1} = \textsc{Viterbi}(\mathbf{m}_{1:t}, e_{t+1}) = P(e_{t+1}|X_{t+1}) \max_{x_t} P(X_{t+1}\,|\, x_t)\, \mathbf{m}_{1:t}$$

Viterbi algorithm contd.



W_{t-1}	$P(W_t \mid W_{t-1})$	
	sun	rain
sun	0.9	0.1
rain	0.3	0.7

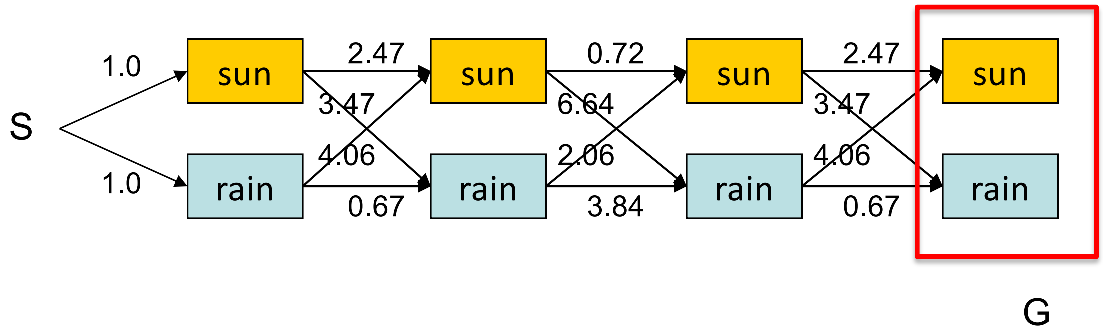
W_t	$P(U_t \mid W_t)$	
	true	false
sun	0.2	0.8
rain	0.9	0.1

Time complexity?
$O(|X|^2 T)$

Space complexity?
$O(|X| T)$

Number of paths?
$O(|X|^T)$

Viterbi in negative log space



W_{t-1}	$P(W_t \mid W_{t-1})$	
	sun	rain
sun	0.9	0.1
rain	0.3	0.7

W_t	$P(U_t \mid W_t)$	
	true	false
sun	0.2	0.8
rain	0.9	0.1

argmax of product of probabilities

= argmin of sum of negative log probabilities

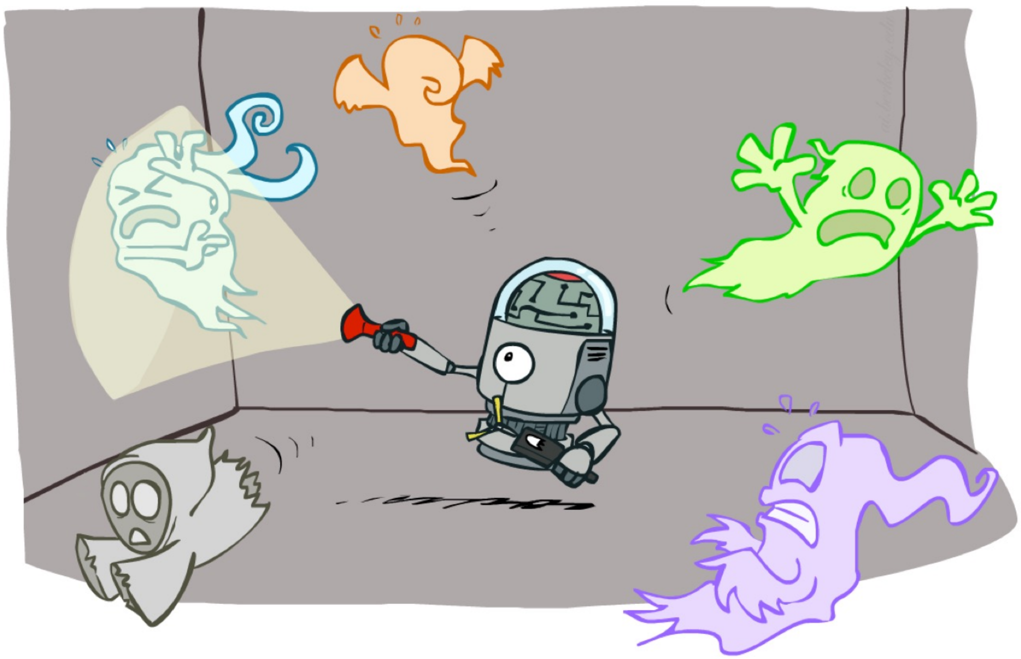
= minimum-cost path

Viterbi is essentially breadth-first graph search

What about A*?

CS 188: Artificial Intelligence

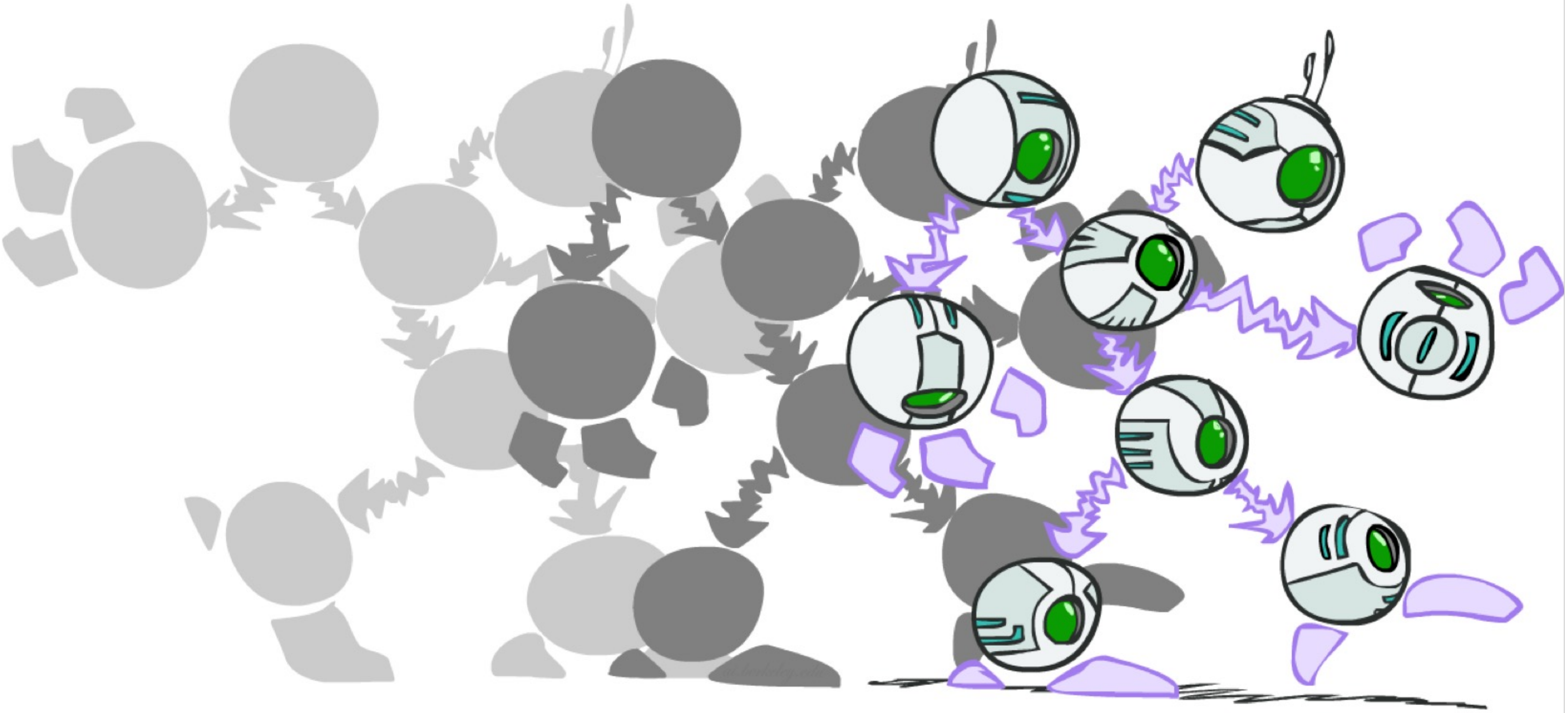
Dynamic Bayes Nets and Particle Filters



Slides from Stuart Russell

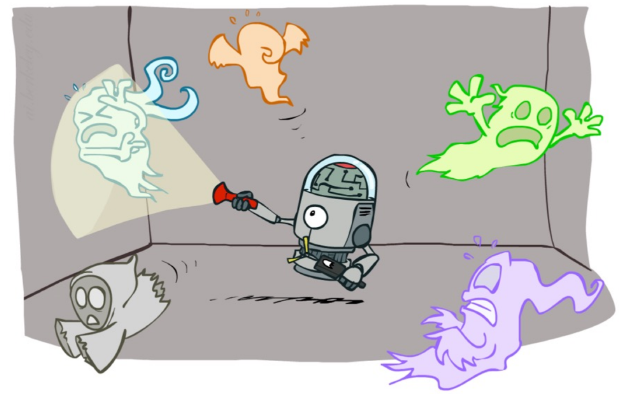
University of California, Berkeley

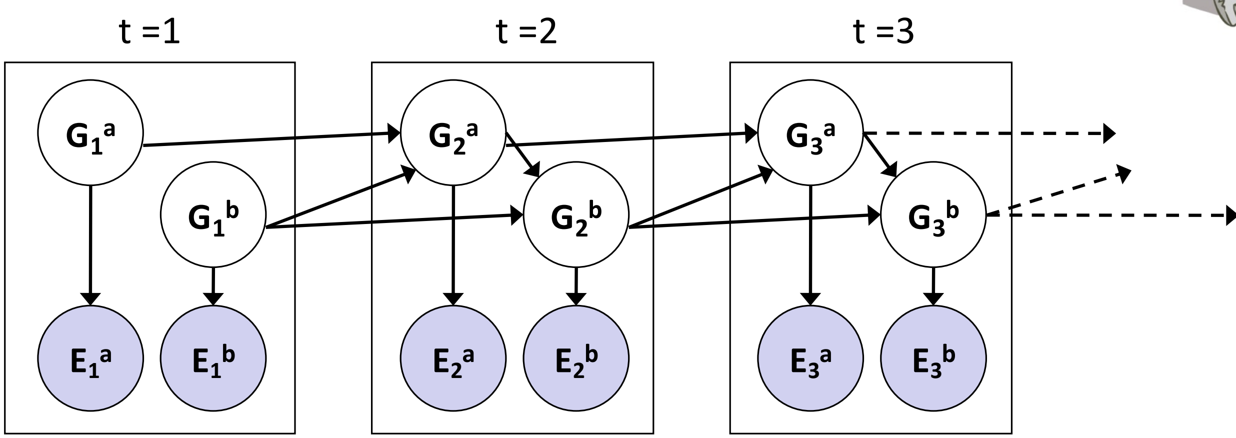
Dynamic Bayes Nets



Dynamic Bayes Nets (DBNs)

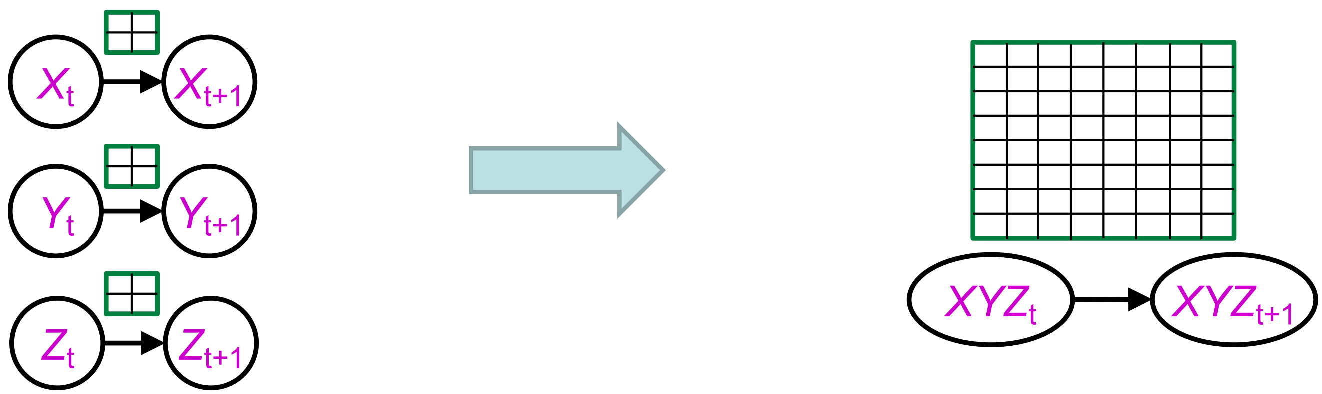
- We want to track multiple variables over time, using multiple sources of evidence
- Idea: Repeat a fixed Bayes net structure at each time
- Variables at time *t* can have parents at time *t* or *t-1*





DBNs and HMMs

- Every HMM is a single-variable DBN
- Every discrete DBN is an HMM
  - HMM state is Cartesian product of DBN state variables

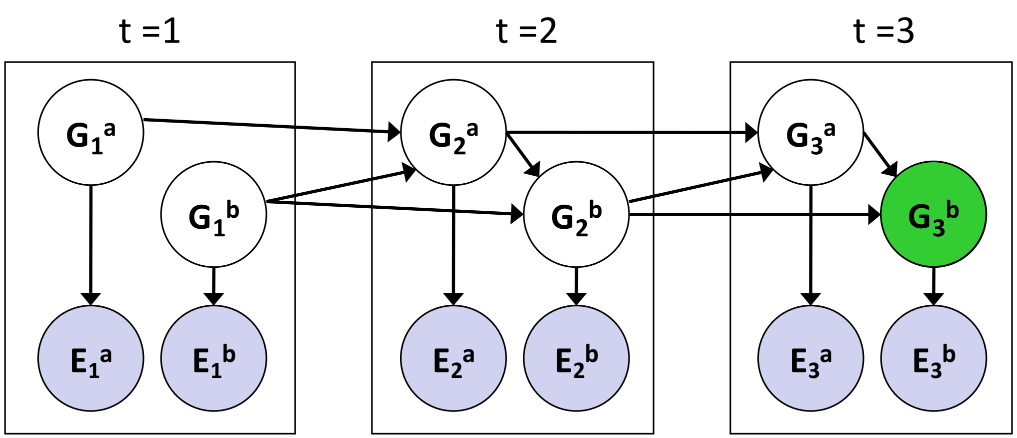


- Sparse dependencies => exponentially fewer parameters in DBN
  - E.g., 20 Boolean state variables, 3 parents each;

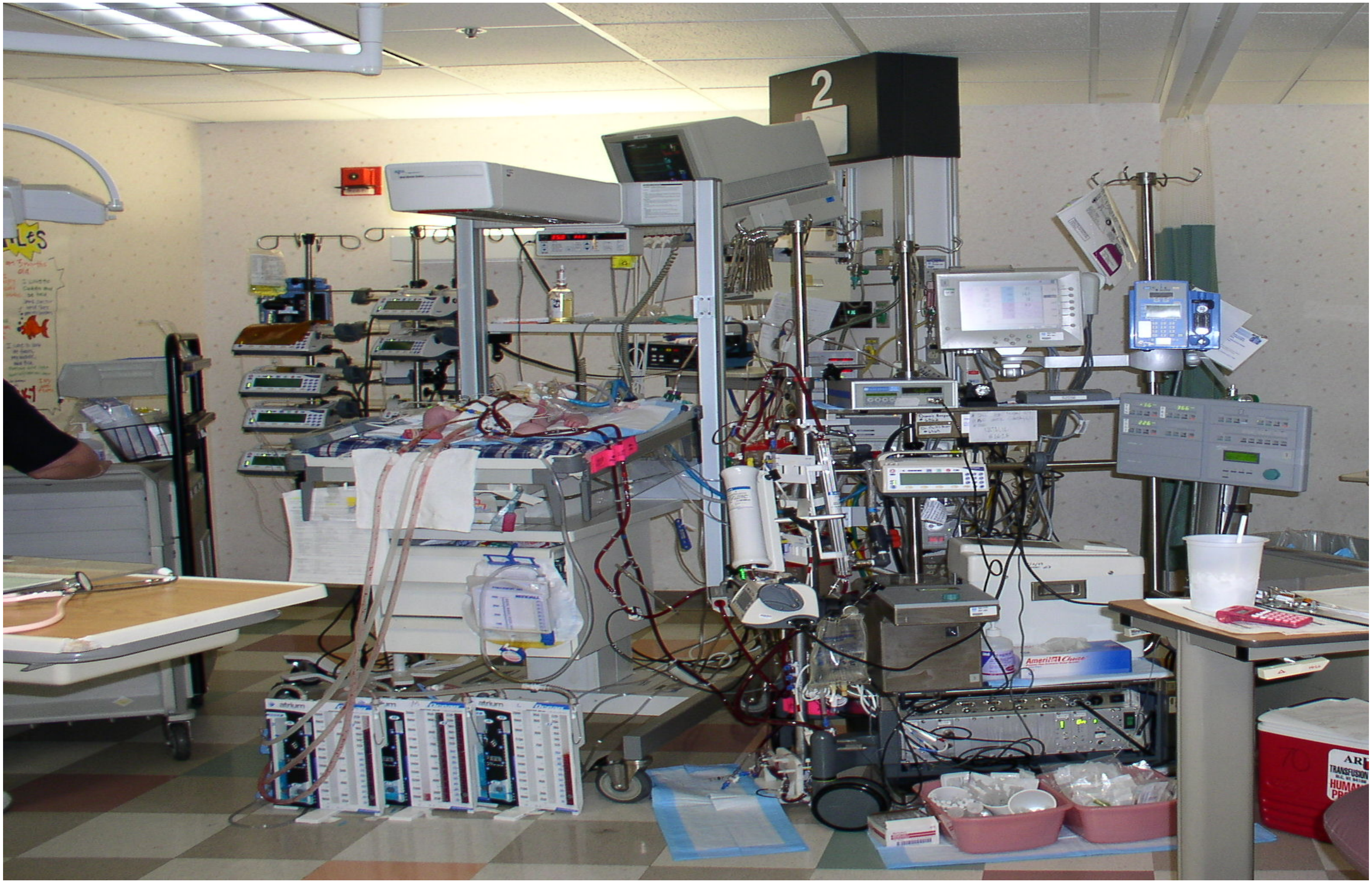
    DBN has 20 x 2^3 = 160 parameters, HMM has 2^{20} x 2^{20} =~ 10^{12} parameters

Exact Inference in DBNs

- Variable elimination applies to dynamic Bayes nets
- Offline: "unroll" the network for T time steps, then eliminate variables to find $P(X_T | e_{1:T})$



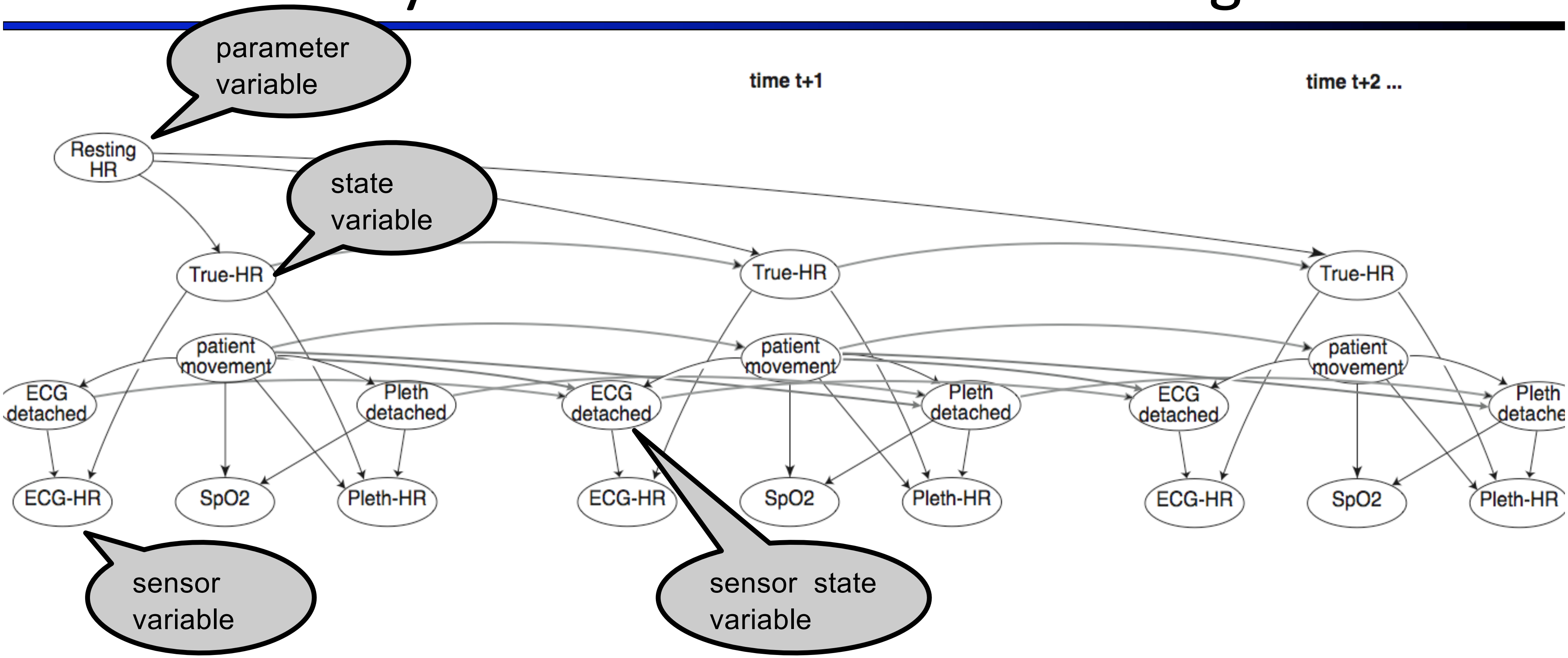
- Online: eliminate all variables from the previous time step; store factors for current time only
- Problem: largest factor contains all variables for current time (plus a few more)


2
atrium
atrium

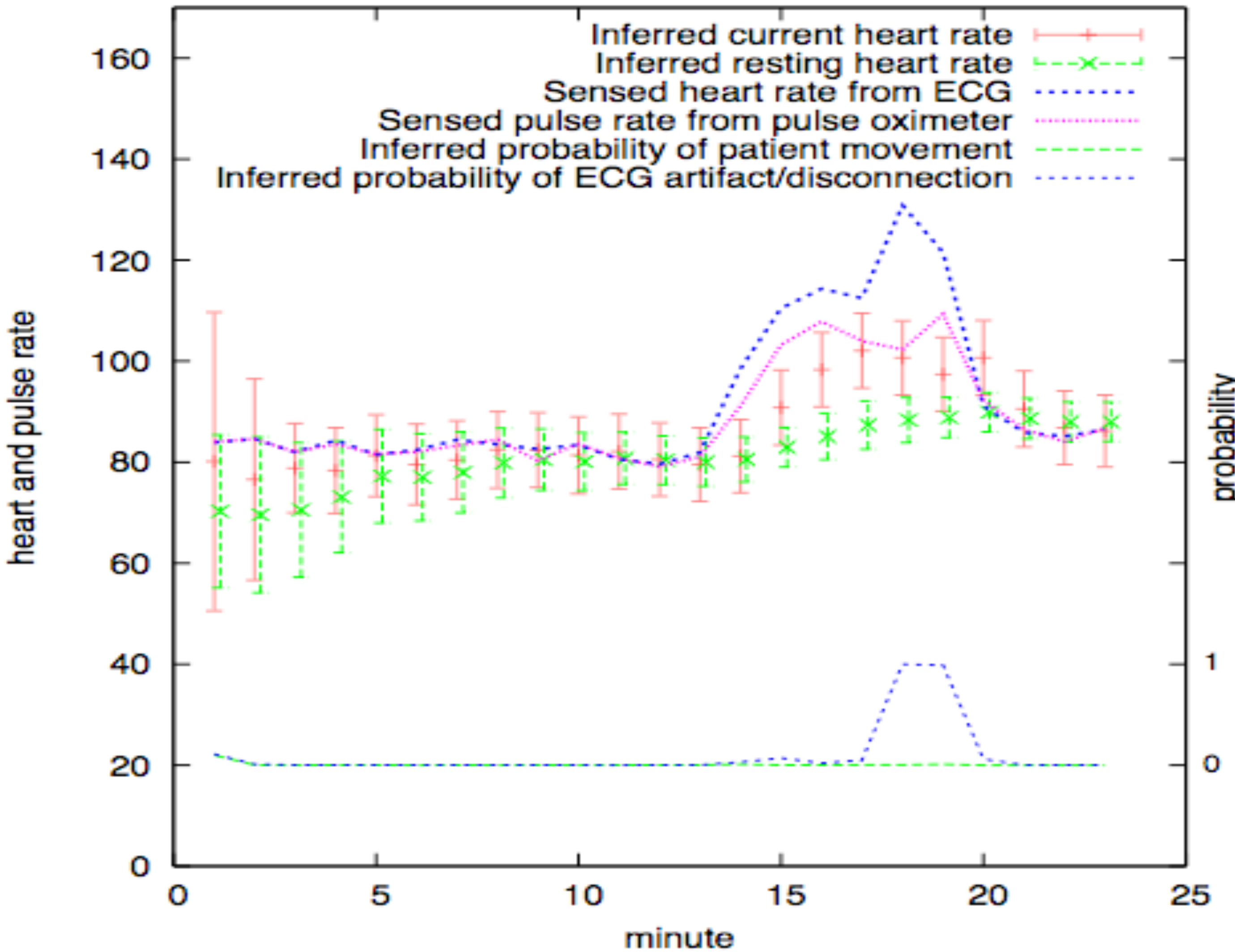
Application: ICU monitoring

- ***State***: variables describing physiological state of patient
- ***Evidence***: values obtained from monitoring devices
- ***Transition model***: physiological dynamics, sensor dynamics
- ***Query variables***: pathophysiological conditions (a.k.a. bad things)

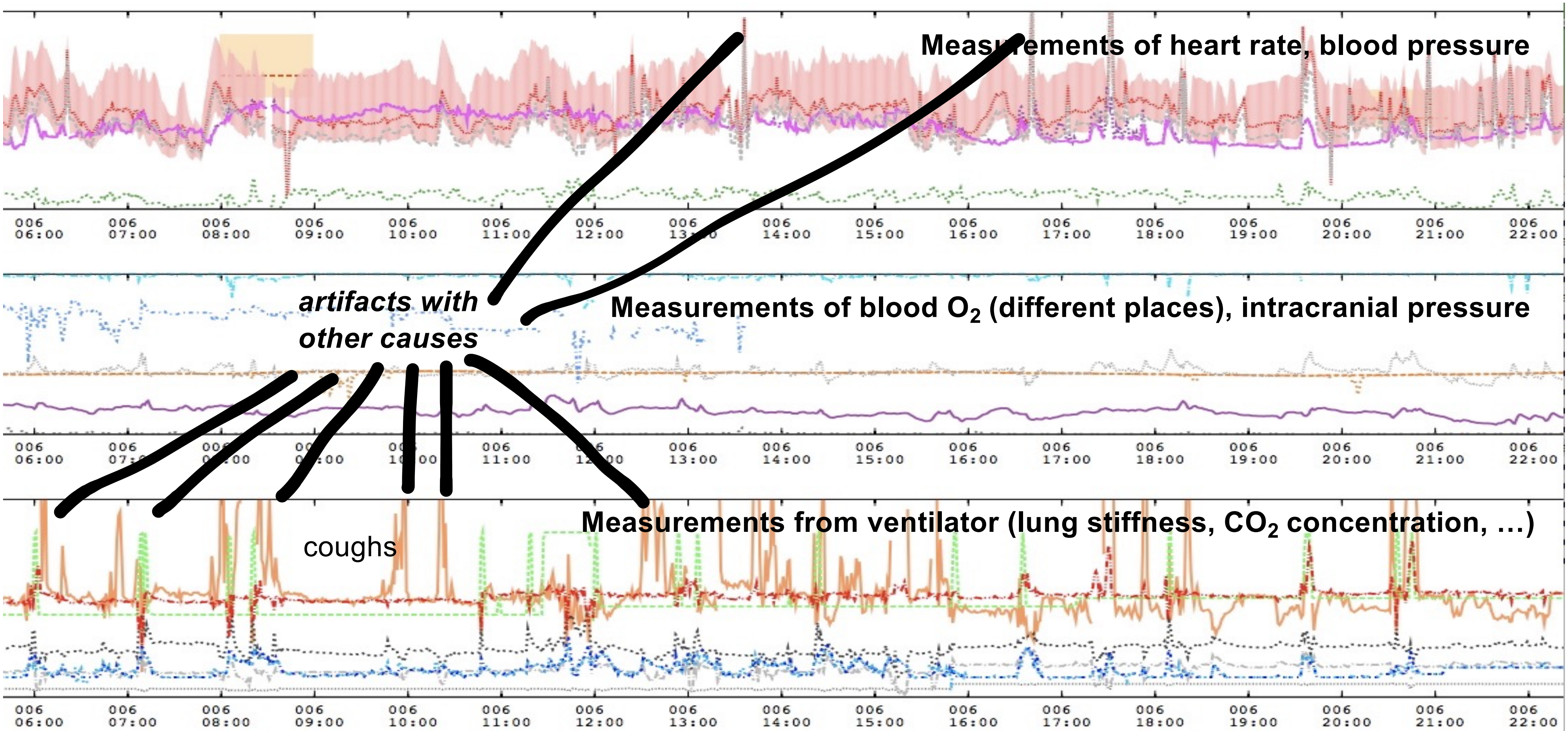
Toy DBN: heart rate monitoring

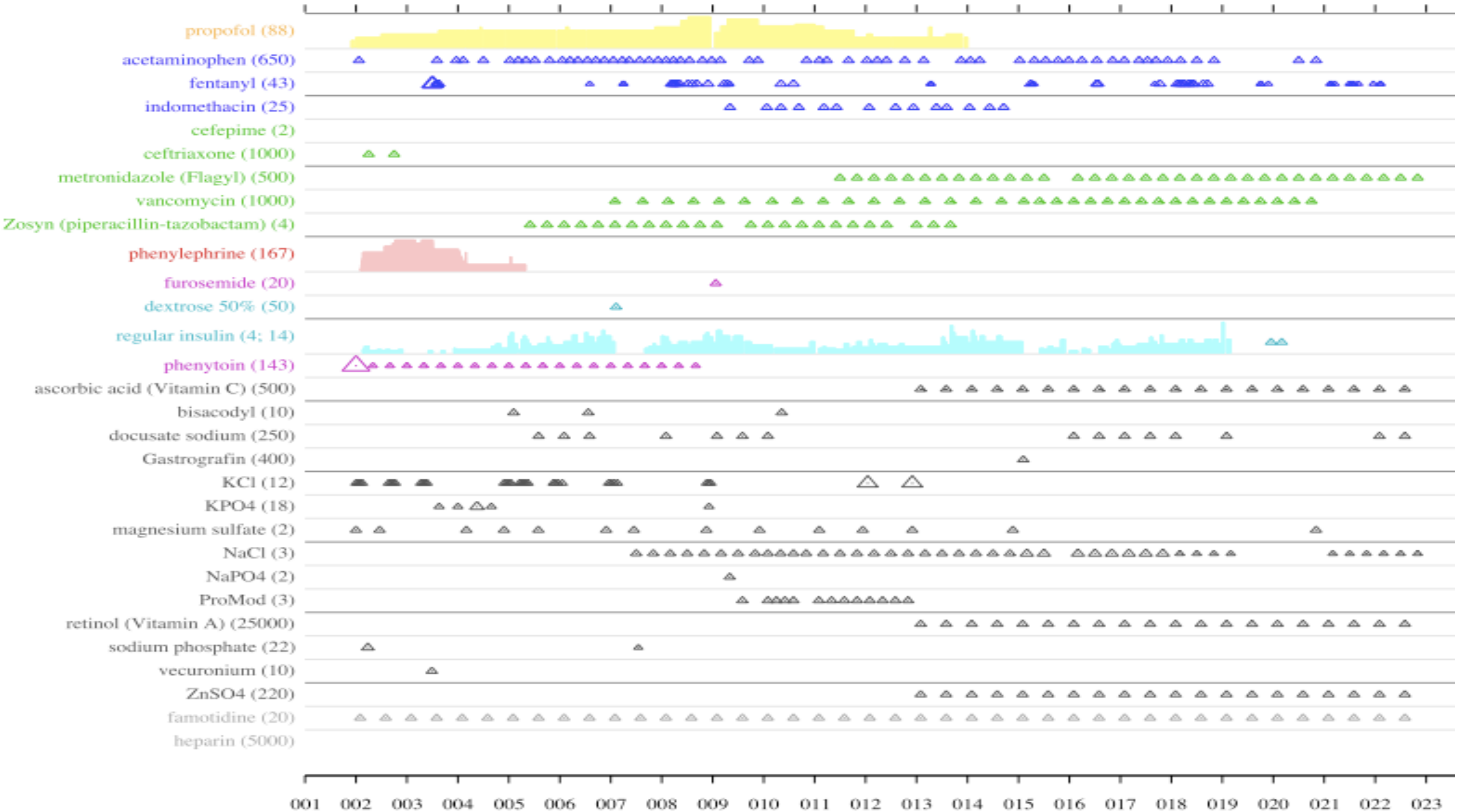


The enhanced heart-rate DBN's inferences on data from a healthy 40-year-

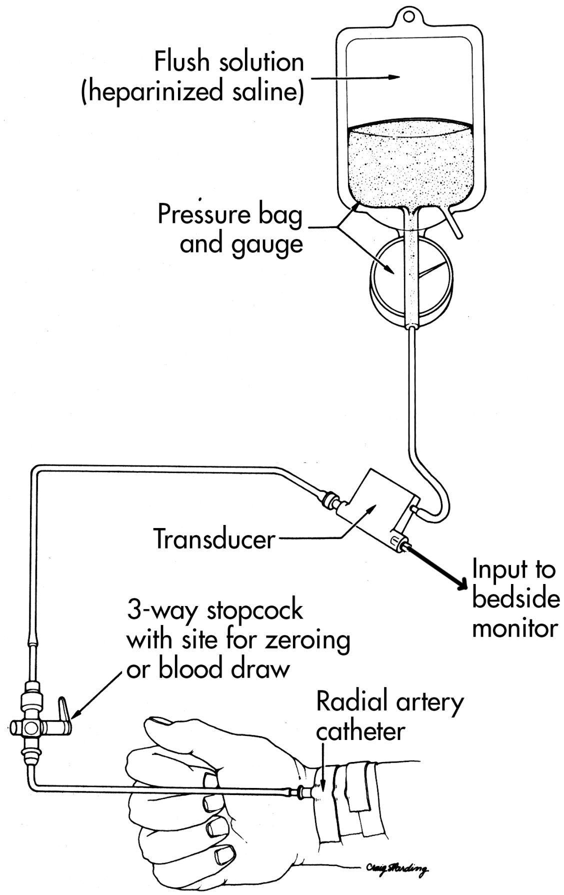


ICU data: 22 variables, 1min ave

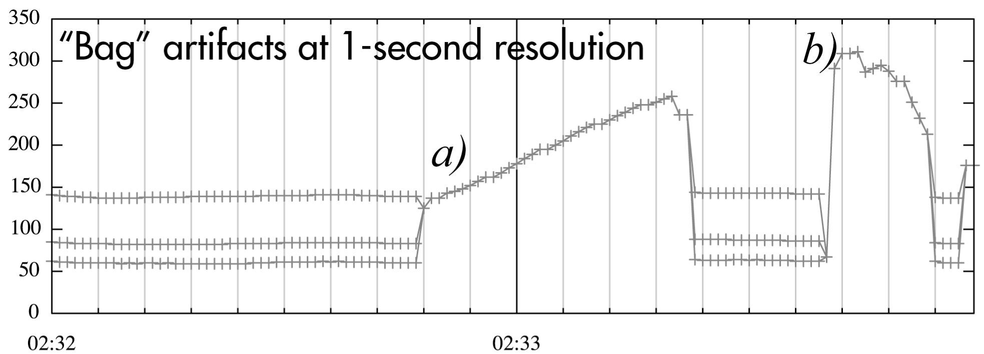


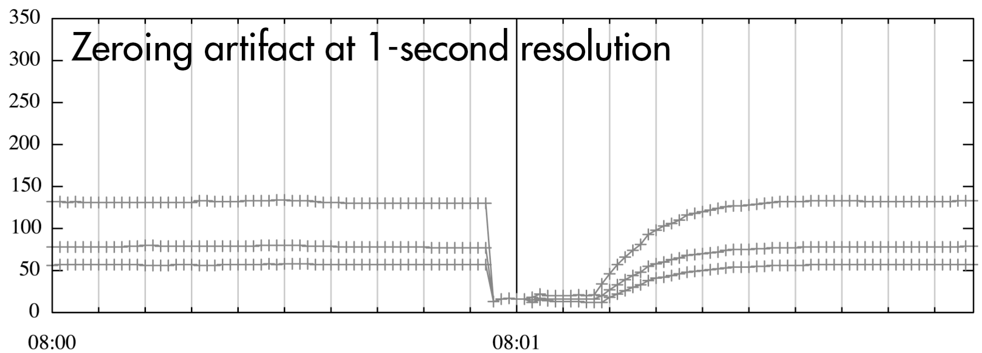

propofol (88)
acetaminophen (650)
fentanyl (43)
indomethacin (25)
cefepime (2)
ceftriaxone (1000)
metronidazole (Flagyl) (500)
vancomycin (1000)
Zosyn (piperacillin-tazobactam) (4)
phenylephrine (167)
furosemide (20)
dextrose 50% (50)
regular insulin (4; 14)
phenytoin (143)
ascorbic acid (Vitamin C) (500)
bisacodyl (10)
docusate sodium (250)
Gastrografin (400)
KCl (12)
KPO4 (18)
magnesium sulfate (2)
NaCl (3)
NaPO4 (2)
ProMod (3)
retinol (Vitamin A) (25000)
sodium phosphate (22)
vecuronium (10)
ZnSO4 (220)
famotidine (20)
heparin (5000)
001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023

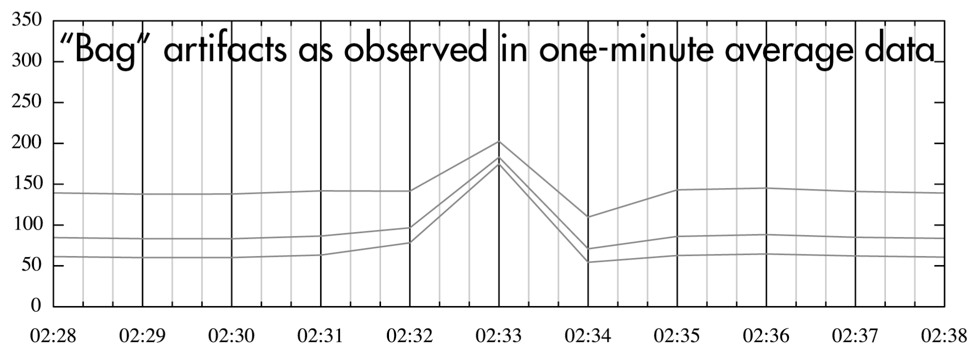
Blood pressure measurement

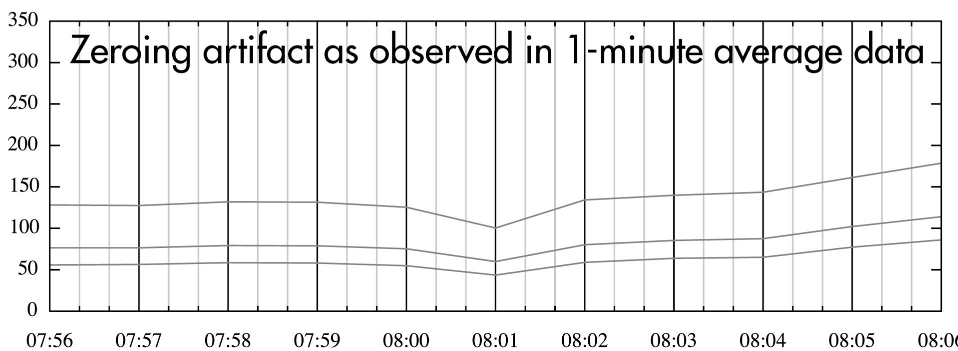


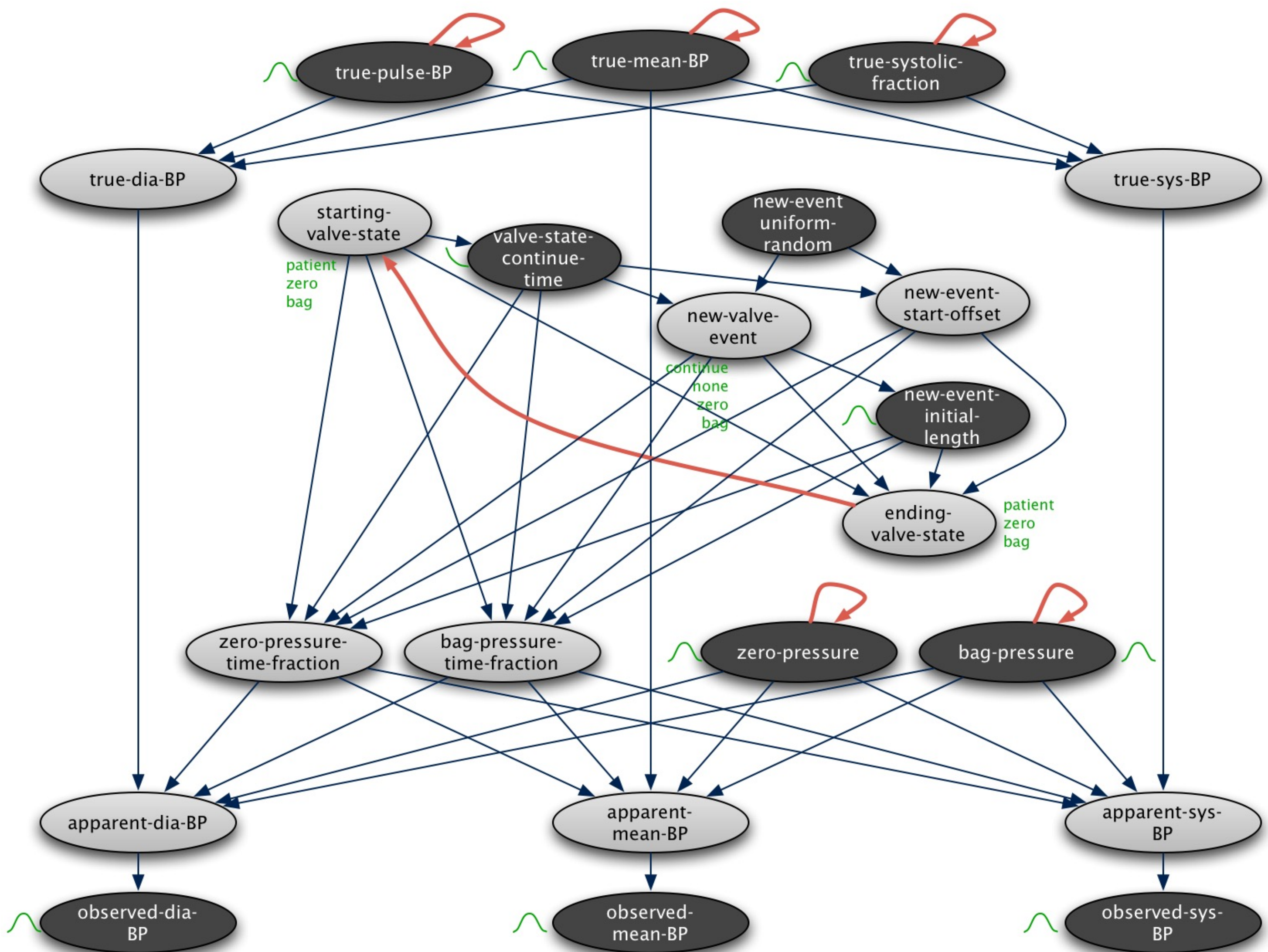
One-second vs one-minute data

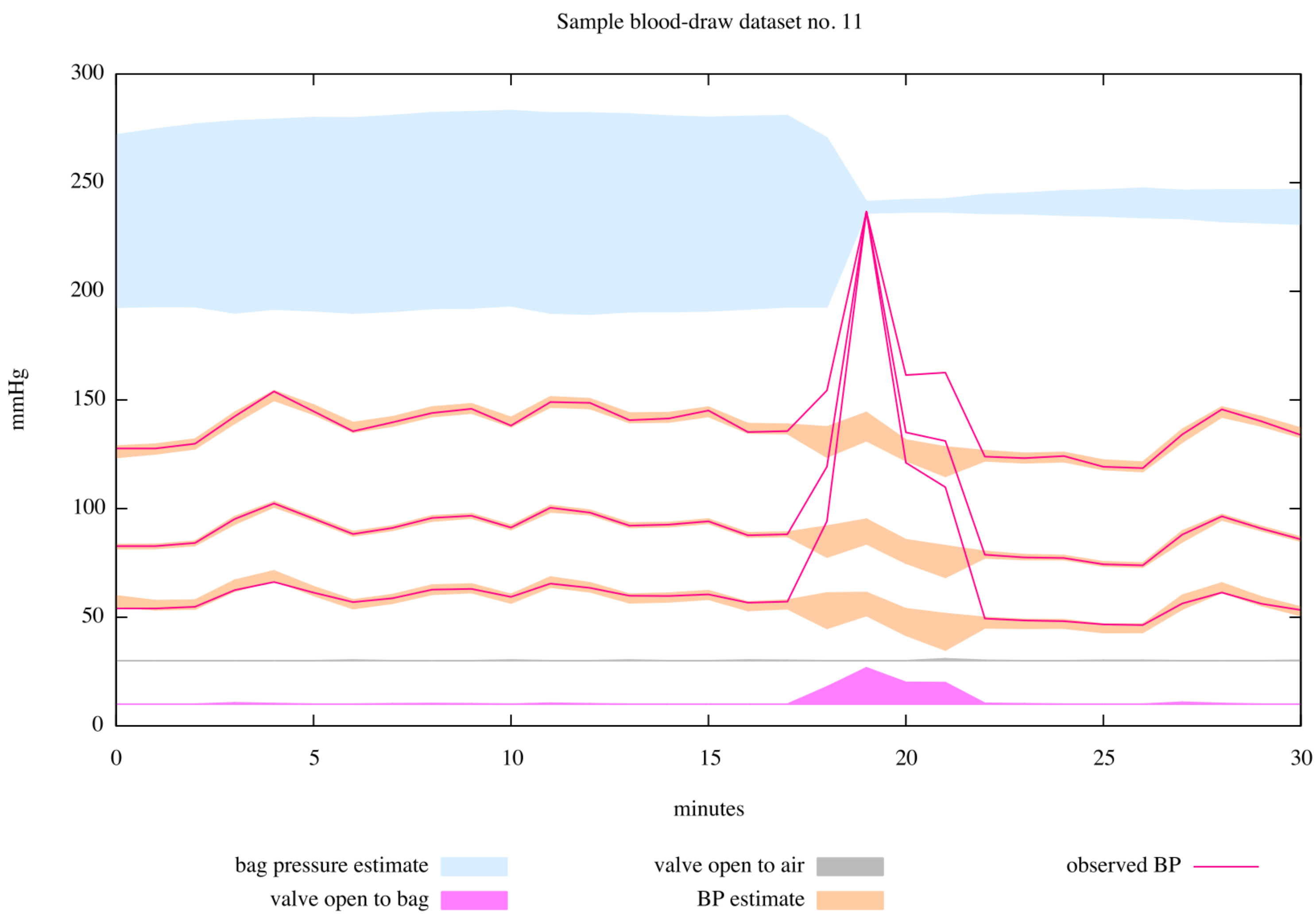


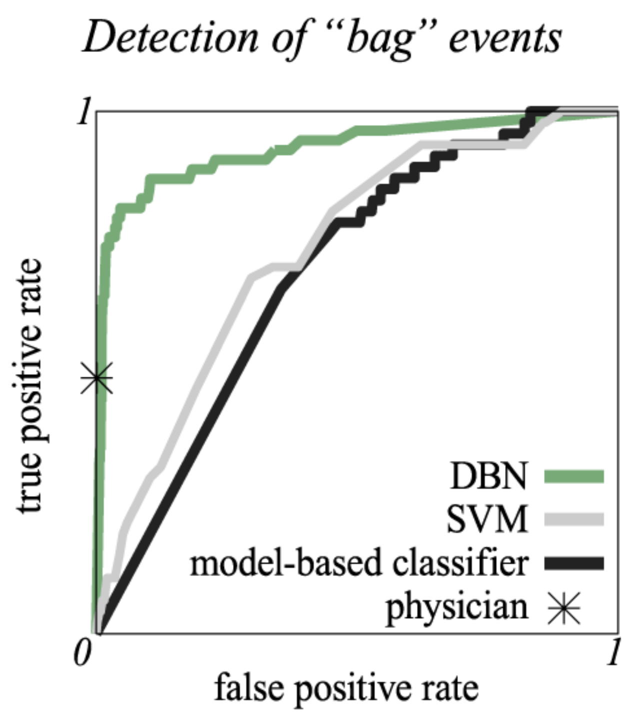




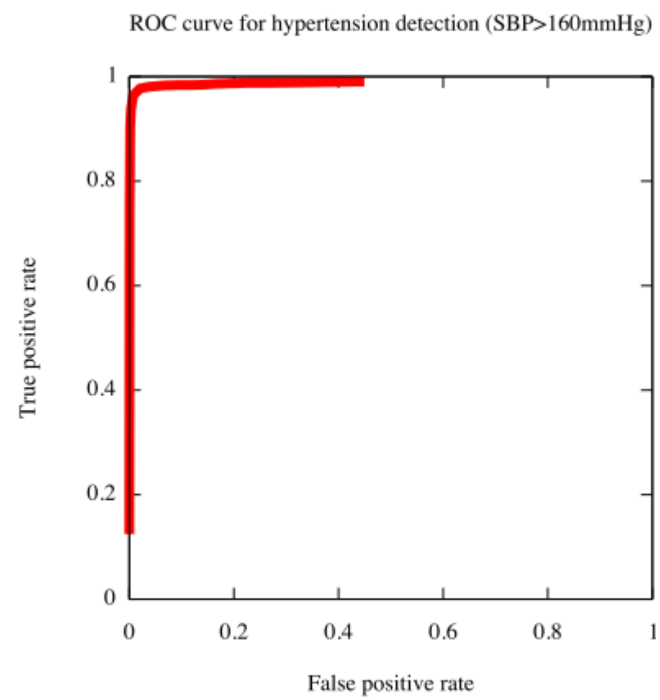



true-pulse-BP
true-mean-BP
true-systolic-fraction
true-dia-BP
true-sys-BP
starting-valve-state
patient zero bag
valve-state-continue-time
new-event uniform-random
new-event-start-offset
new-valve-event
continue none zero bag
new-event-initial-length
ending-valve-state
patient zero bag
zero-pressure-time-fraction
bag-pressure-time-fraction
zero-pressure
bag-pressure
apparent-dia-BP
apparent-mean-BP
apparent-sys-BP
observed-dia-BP
observed-mean-BP
observed-sys-BP

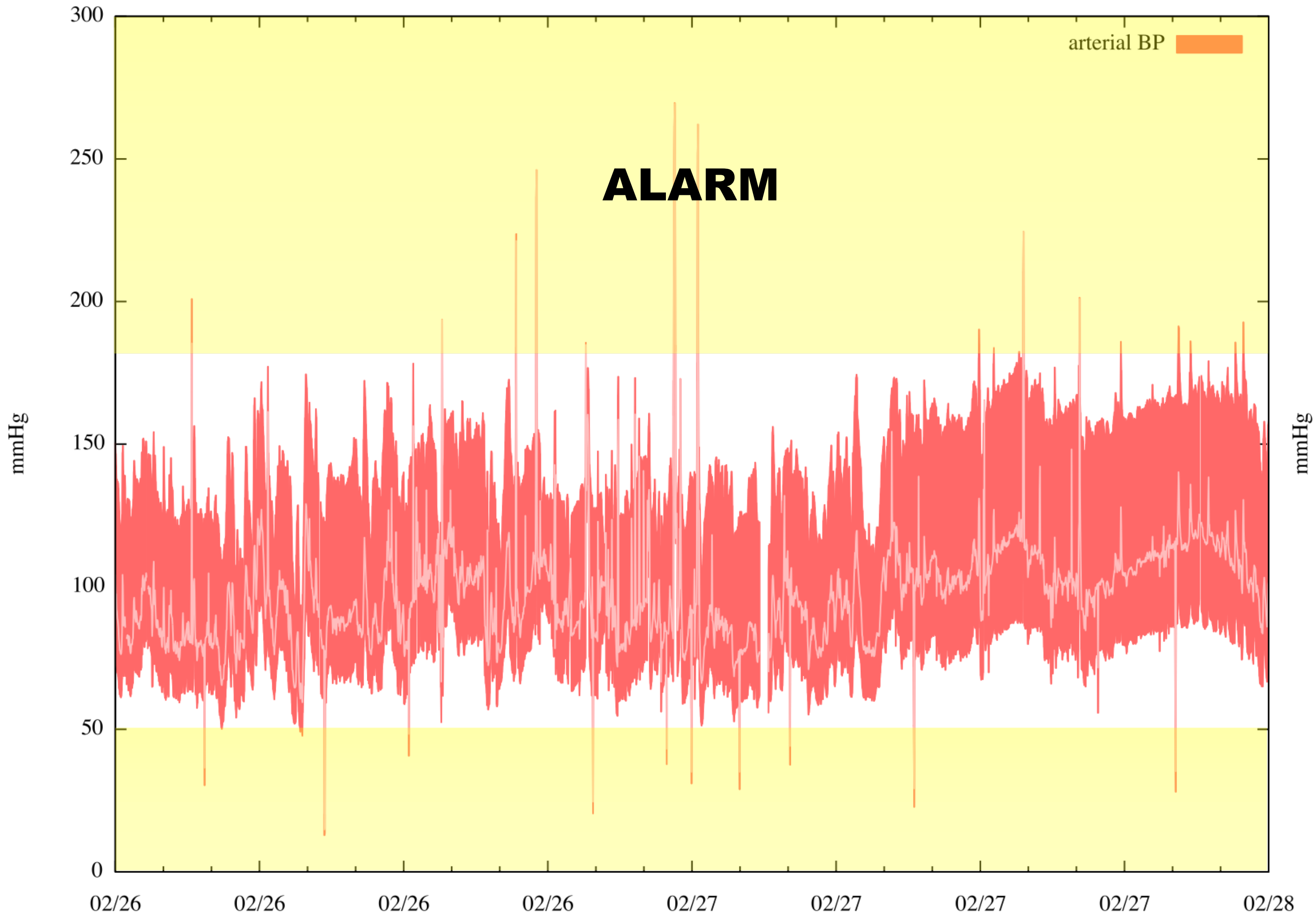

Sample blood-draw dataset no. 11
mmHg
0
50
100
150
200
250
300
0
5
10
15
20
25
30
minutes
bag pressure estimate
valve open to bag
valve open to air
BP estimate
observed BP



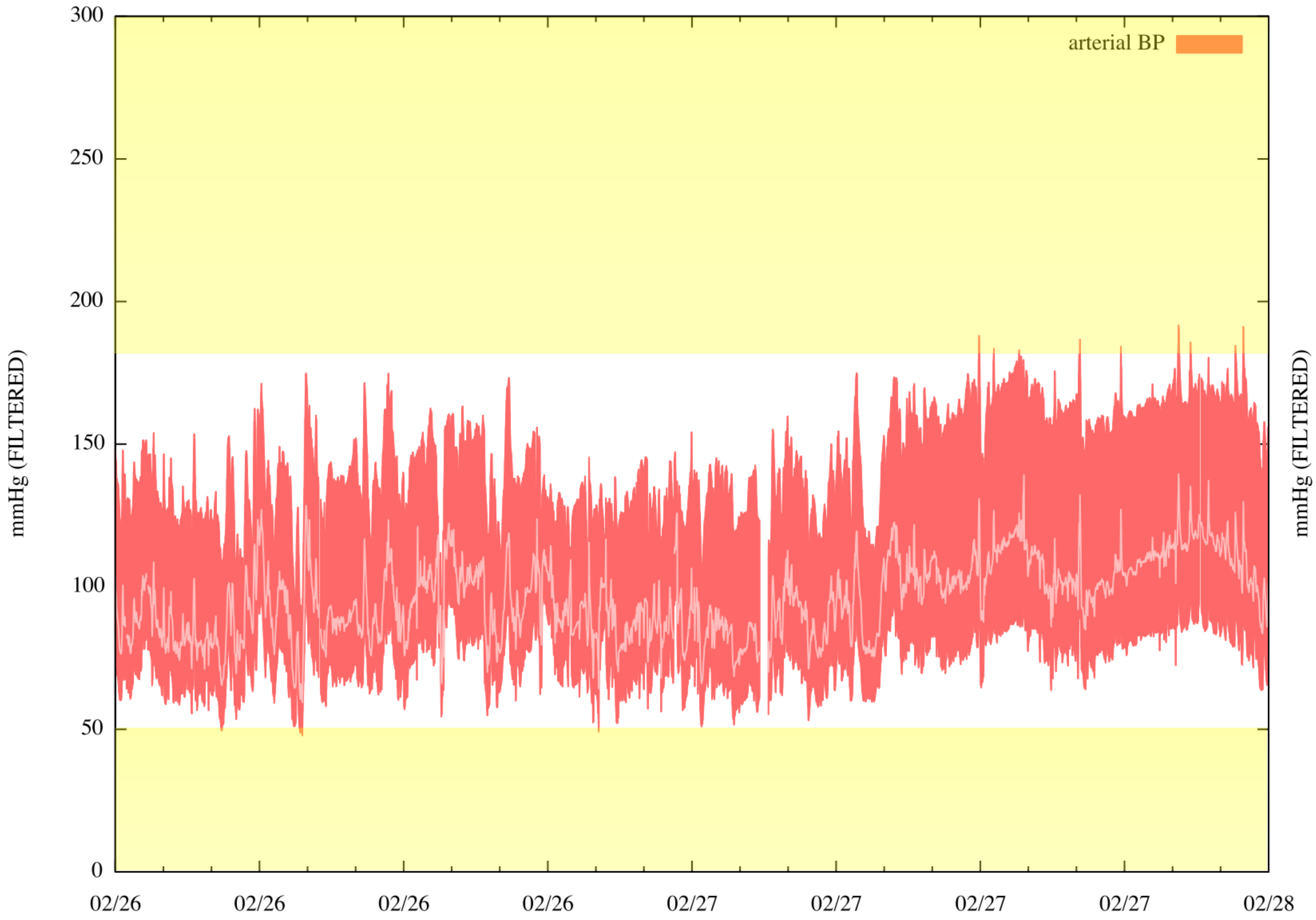
Detection of "bag" events
1
0
1
true positive rate
false positive rate
DBN
SVM
model-based classifier
physician



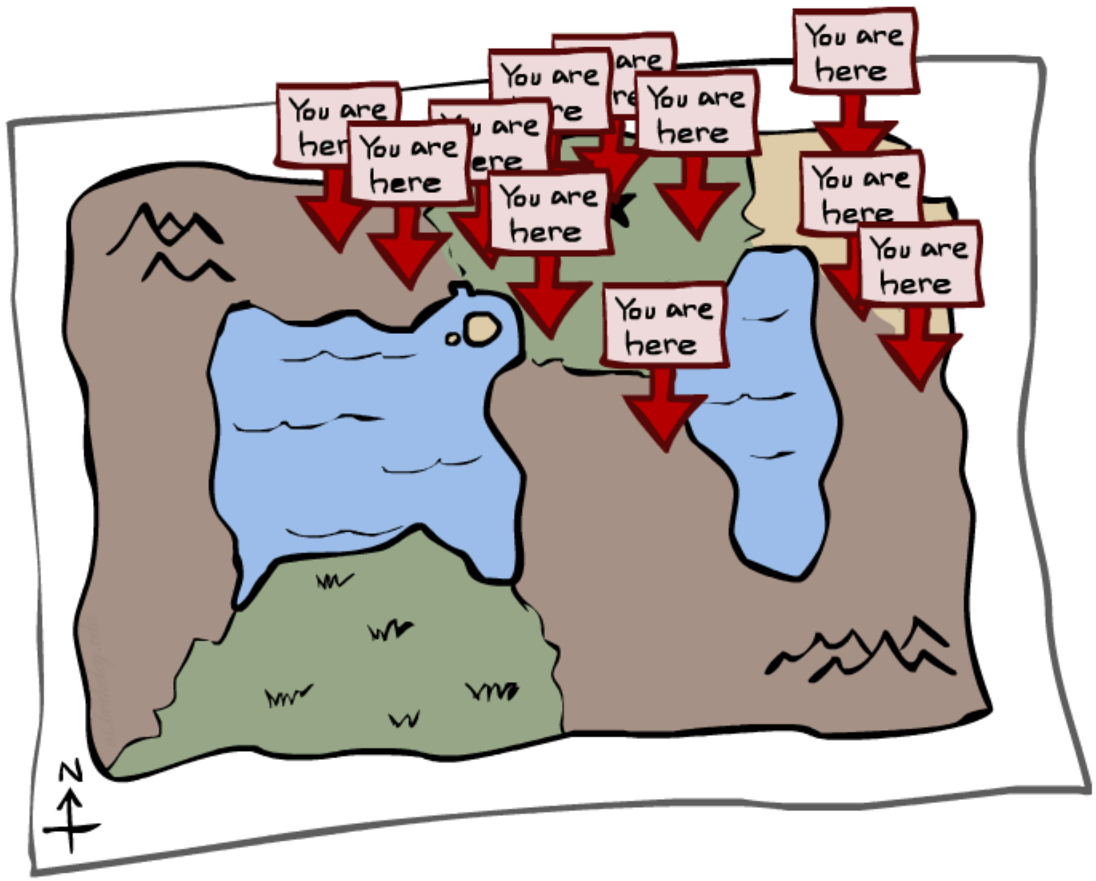
ROC curve for hypertension detection (SBP>160mmHg)
True positive rate
False positive rate
0
0.2
0.4
0.6
0.8
1



300
250
200
150
100
50
0
mmHg
mmHg
arterial BP
ALARM
02/26
02/26
02/26
02/26
02/27
02/27
02/27
02/27
02/28

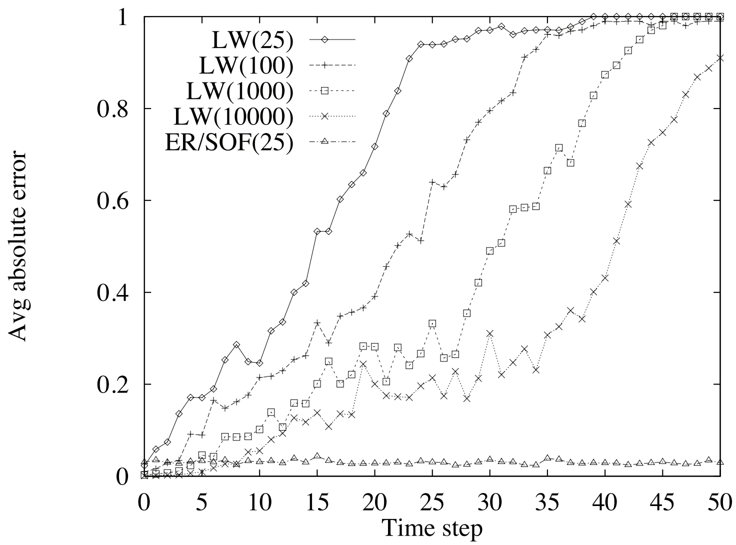

arterial BP
mmHg (FILTERED)
mmHg (FILTERED)
300
250
200
150
100
50
0
02/26
02/26
02/26
02/26
02/27
02/27
02/27
02/27
02/28

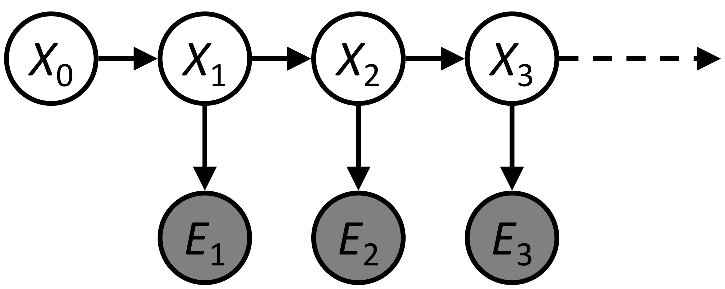
Particle Filtering



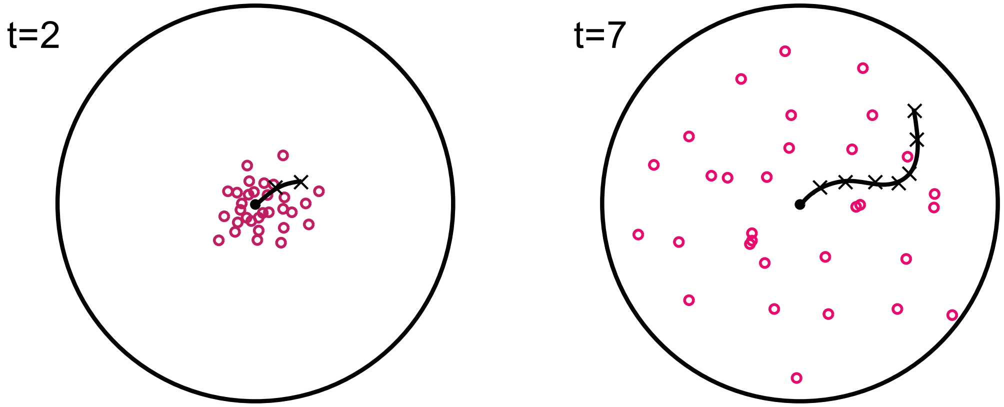
We need a new algorithm!

- When $|X|$ is more than 10^6 or so (e.g., 3 ghosts in a 10x20 world), exact inference becomes infeasible
- Likelihood weighting fails completely – number of samples needed grows ***exponentially*** with T





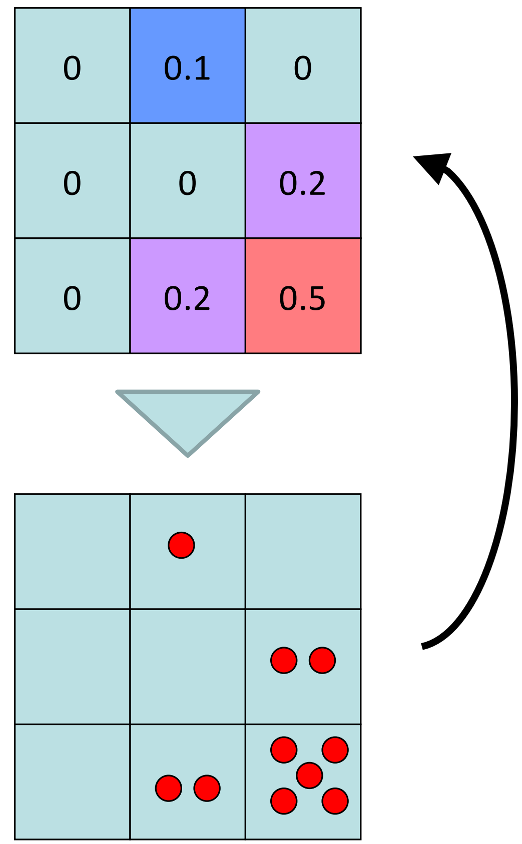
We need a new idea!



- The problem: sample state trajectories go off into low-probability regions, ignoring the evidence; too few “reasonable” samples
- Solution: kill the bad ones, make more of the good ones
- This way the population of samples stays in the high-probability region
- This is called ***resampling*** or survival of the fittest

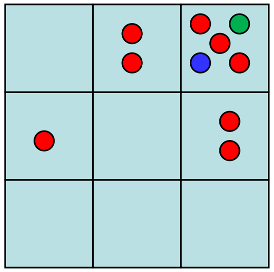
Particle Filtering

- Represent belief state by a set of samples
  - Samples are called ***particles***
  - Time per step is linear in the number of samples
  - But: number needed may be large
- A particle is a possible world state
  - (i.e. a possible assignment of values for each variable at a single timestep)
- This is how robot localization works in practice



Representation: Particles

- Our representation of $P(X)$ is now a list of N << $|X|$ particles
- $P(x)$ approximated by number of particles with value x
  - So, many x may have $P(x) = 0$!
  - More particles => more accuracy (cf. frequency histograms)
  - Usually we want a ***low-dimensional*** marginal
    - E.g., “Where is ghost 1?” rather than “Are ghosts 1,2,3 in [2,6], [5,6], and [8,11]?”
  - Estimates of low-dimensional marginals more accurate
    - (in log space; equally accurate in absolute terms)

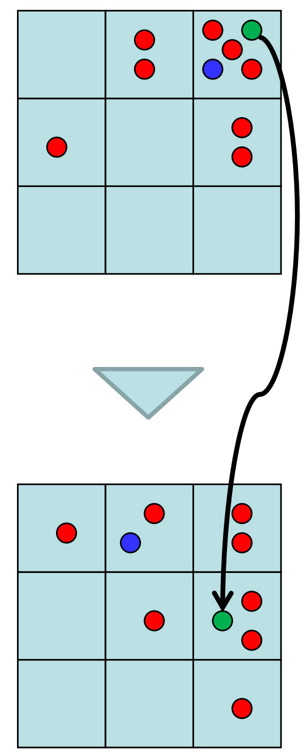


Particles:
(3,3)
(2,3)
(3,3)
(3,2)
(3,3)
(3,2)
(1,2)
(3,3)
(3,3)
(2,3)

Particle Filtering: Prediction step

- Particle j in state $x_t^{(j)}$ samples a new state directly from the transition model:
  - $x_{t+1}^{(j)} \sim P(X_{t+1} \mid x_t^{(j)})$
  - Here, most samples move clockwise, but some move in another direction or stay in place

Particles:
(3,3)
(2,3)
(3,3)
(3,2)
(3,3)
(3,2)
(1,2)
(3,3)
(3,3)
(2,3)

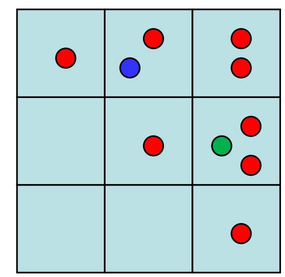


Particles:
(3,2)
(2,3)
(3,2)
(3,1)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(2,2)

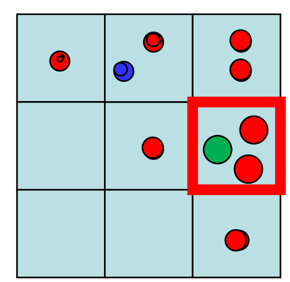
Particle Filtering: Update step

- After observing e_{t+1} :
  - As in likelihood weighting, weight each sample based on the evidence
    - $w^{(j)} = P(e_{t+1} \mid x_{t+1}^{(j)})$
  - Particles that fit the data better get higher weights, others get lower weights
  - Normalize the weights across all particles

Particles:
(3,2)
(2,3)
(3,2)
(3,1)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(2,2)



Particles:
(3,2) w=~~.9~~ .17
(2,3) w=~~.2~~ .04
(3,2) w=~~.9~~ .17
(3,1) w=~~.4~~ .08
(3,3) w=~~.4~~ .08
(3,2) w=~~.9~~ .17
(1,3) w=~~.1~~ .02
(2,3) w=~~.2~~ .04
(3,2) w=~~.9~~ .17
(2,2) w=~~.4~~ .08

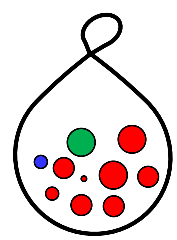


Particle Filtering: Resample

- Rather than tracking weighted samples, we ***resample***

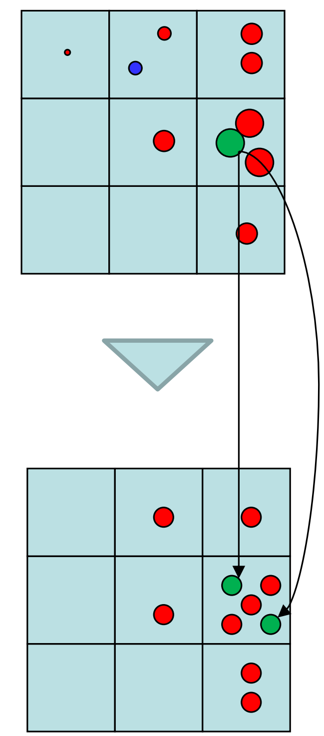
- *N* times, we choose from our weighted sample distribution
  (i.e., draw with replacement)

- Now the update is complete for this time step, continue with the next one
  (with weights reset to 1/N)



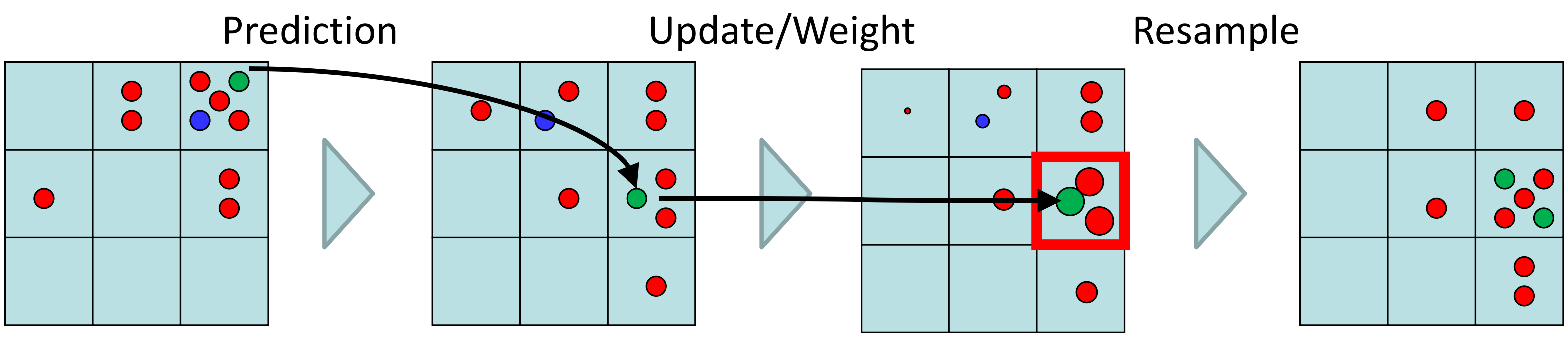
Particles:
(3,2) w=.17
(2,3) w=.04
(3,2) w=.17
(3,1) w=.08
(3,3) w=.08
(3,2) w=.17
(1,3) w=.02
(2,3) w=.04
(3,2) w=.17
(2,2) w=.08

(New) Particles:
(3,2)
(2,2)
(3,2)
(2,3)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(3,2)



Summary: Particle Filtering

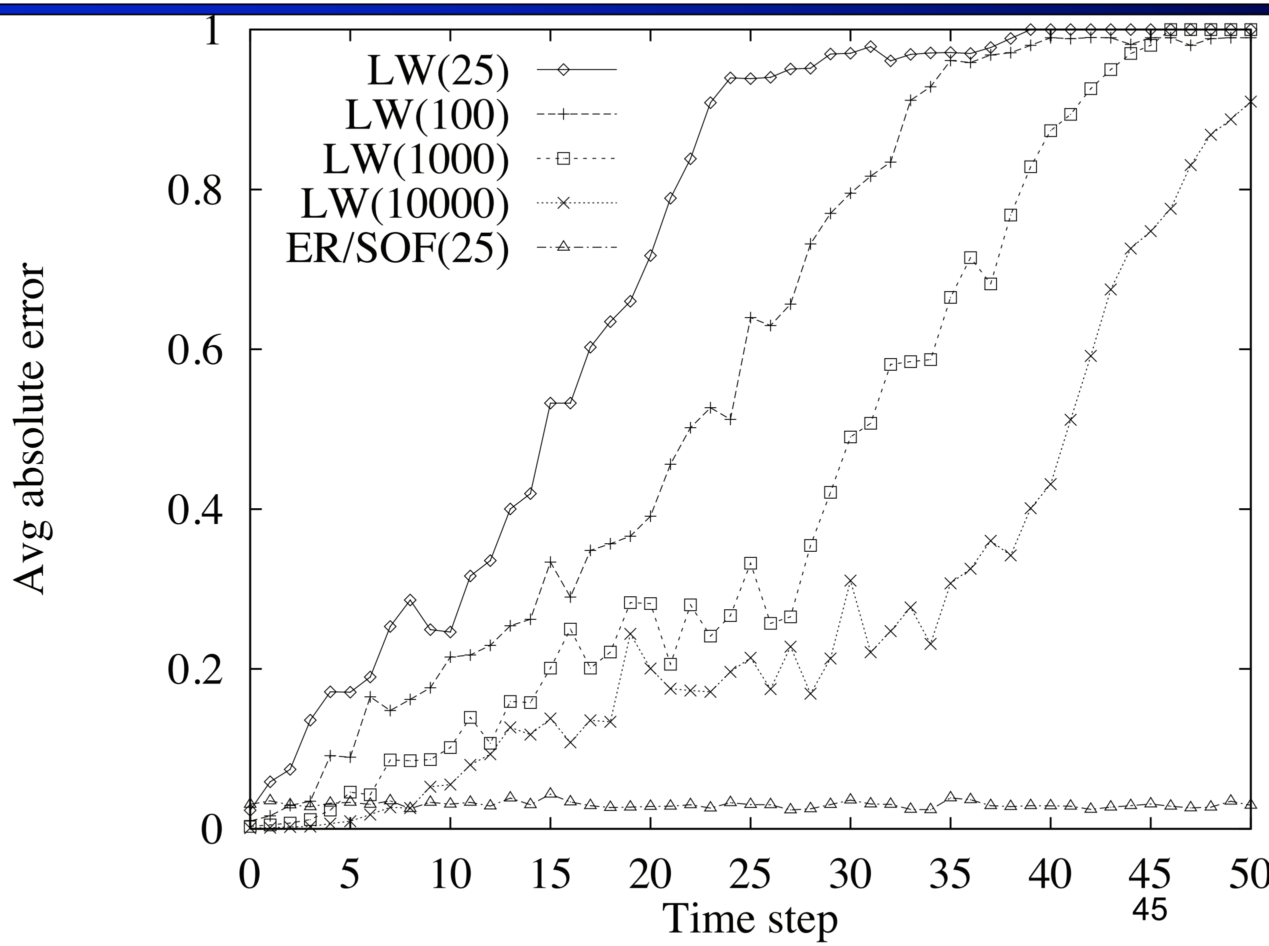
- Particles: track samples of states rather than an explicit distribution



Particles:	Particles:	Particles:	(New) Particles:
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,3)	(2,3) w=.2	(2,2)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(3,2)	(3,1)	(3,1) w=.4	(2,3)
(3,3)	(3,3)	(3,3) w=.4	(3,3)
(3,2)	(3,2)	(3,2) w=.9	(3,2)
(1,2)	(1,3)	(1,3) w=.1	(1,3)
(3,3)	(2,3)	(2,3) w=.2	(2,3)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,2)	(2,2) w=.4	(3,2)

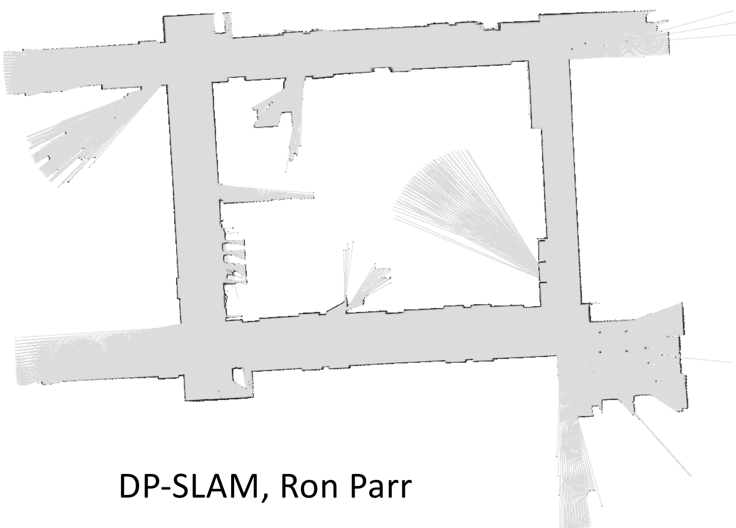
Consistency: see proof in AIMA Ch. 14 (requires DBN probabilities to be bounded away from 0)

Particle filtering on umbrella model

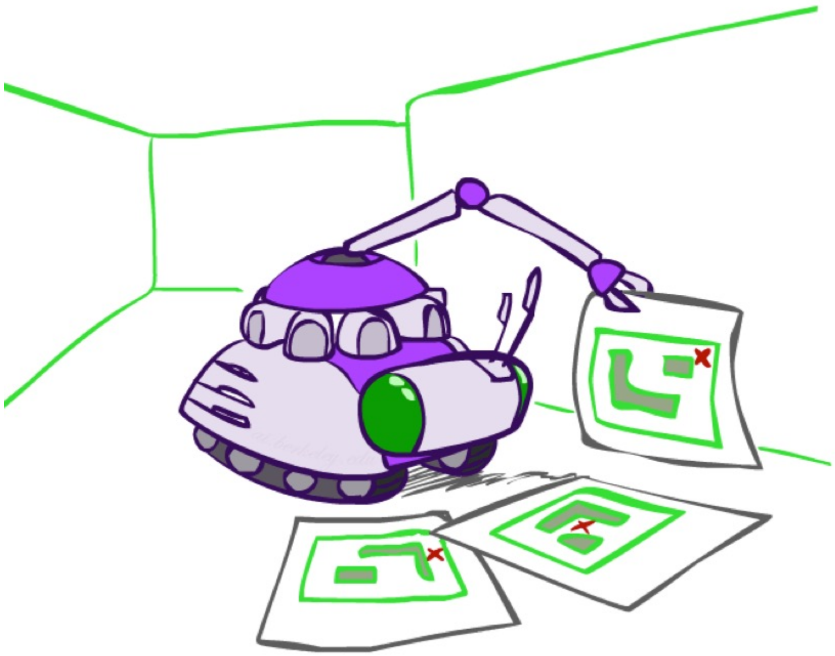


Robot Mapping

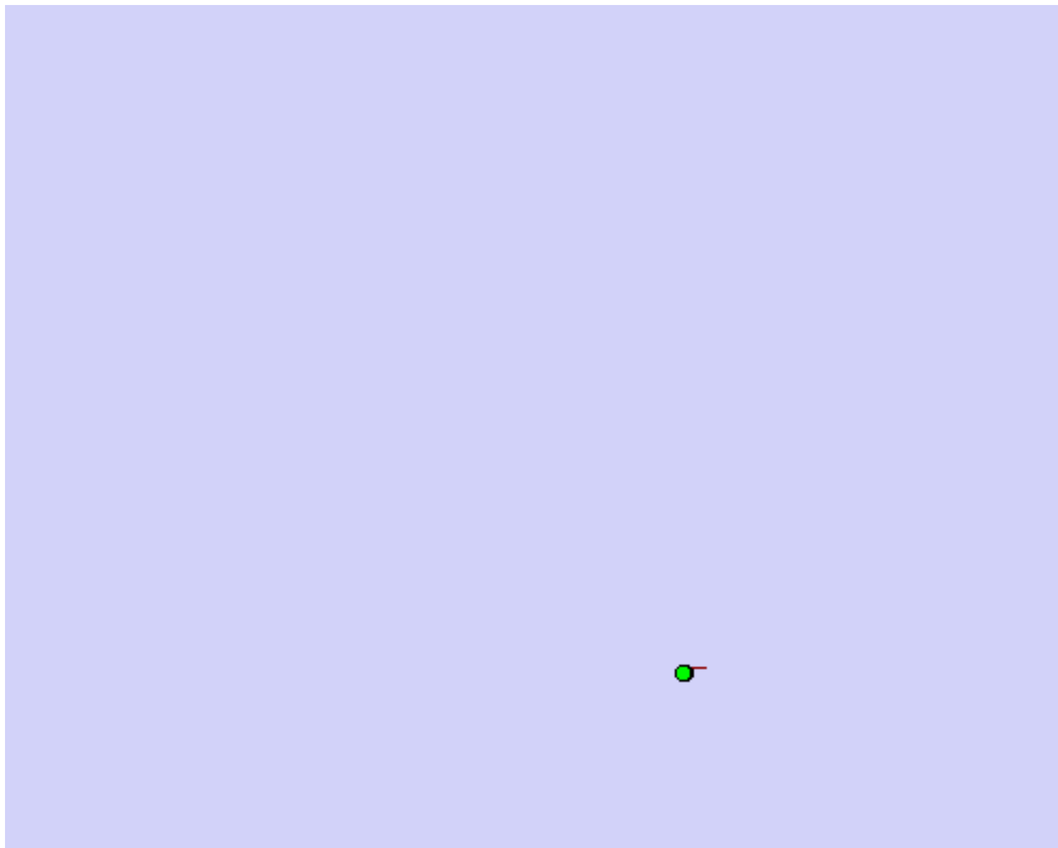
- SLAM: Simultaneous Localization And Mapping
  - Robot does not know map or location
  - State $x_t^{(j)}$ consists of position+orientation, map!
  - (Each map usually inferred exactly given sampled position+orientation sequence: RBPF)



DP-SLAM, Ron Parr



Particle Filter SLAM – Video



[Demo: PARTICLES-SLAM-fastslam.avi]