Graphical Models for Inference and Decision Making

Instructor: Kathryn Blackmond Laskey

Room 2214 ENGR

(703) 993-1644

Office Hours: Tuesday and Thursday 4:30-5:30 PM, or by appointment

Spring 2019

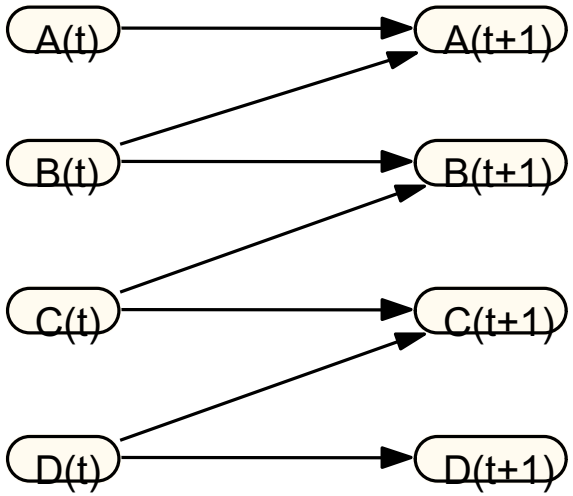
Inference in Dynamic Bayesian Networks

Objectives

- Define a state space model
- Define a dynamic Bayesian network and state why inference in DBNs tends to be computationally complex
  - Standard DBN
  - Partially Dynamic Bayesian Network (PDBN)
- Describe typical inference problems for DBNs:
  - Prediction
  - Filtering
  - Smoothing
  - Estimation
- Describe how the following DBN inference algorithms work:
  - Temporal rollup exact inference
  - Boyen-Koller approximate inference
  - Particle filter approximate inference
- Describe key research challenges for inference in state space models

Dynamic Bayesian Networks

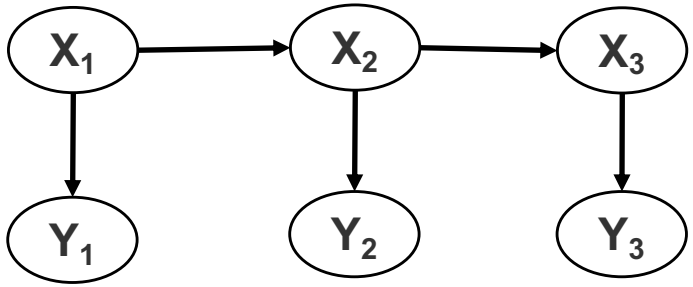
- **A dynamic Bayesian network (DBN) is a Bayesian network for which:**
  - **Nodes are indexed by time**
  - **Time is integer-valued and begins at zero (convenient assumption that loses no generality)**
  - **The local distribution for a variable can depend on**
    - » **Any variable that precedes it in time**
    - » **Variables at the same time that are prior to it in the node ordering**
  - **There is an integer *k*, called the *order* of the DBN such that the local distribution of X_t is the same for all t > k**
- **A DBN is an order *k* Markov chain**



MEBN fragment representation for an order 1 DBN

State Space Model

- A *state space model* is a representation for a dynamic system that satisfies the following conditions:
  - The behavior of the system depends on a *state* X_t which evolves in time
  - The state X_t at time t depends on the past only through its dependence on the immediate past state X_{t-1} (*Markov assumption*)
    - » $P(X_t \mid X_0, X_1, \ldots, X_{t-1}) = P(X_t \mid X_{t-1})$
  - The state is *hidden* and cannot be observed directly
  - We learn about the state through an *observation* Y_t which depends on X_t but not on past states or past observations
    - » $P(Y_t \mid X_0, X_1, \ldots, X_t, Y_0, Y_1, \ldots, Y_{t-1}) = P(Y_t \mid X_t)$
- State space models are a powerful and general way to model systems that evolve in time

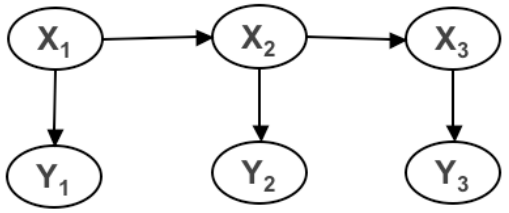


Typical Tasks for State Space Models

- **Prediction -**
  - Predict states of future variables given observations on past variables
- **Filtering -**
  - Infer values of current unobserved variables given current and past observed variables
- **Smoothing -**
  - Revise inferences about past unobserved variables given new observations
- **Estimation -**
  - Infer hidden static parameters from observations

These tasks may be performed offline in batch mode or online sequentially

Examples

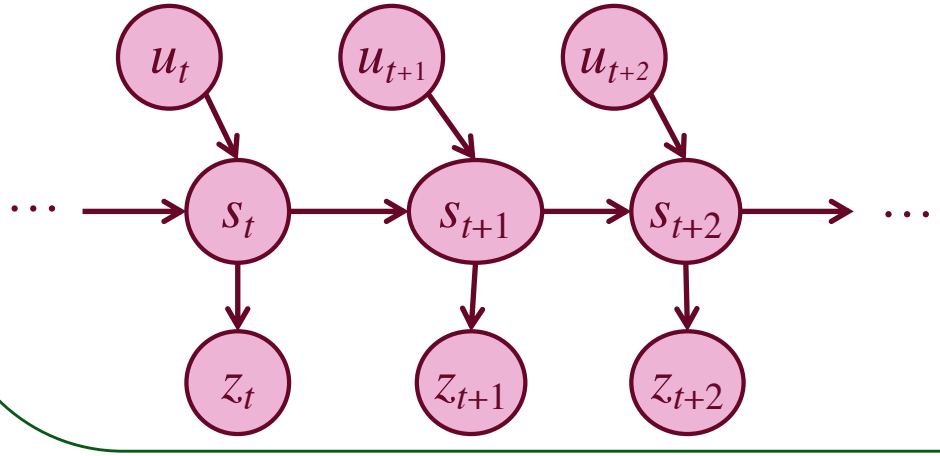


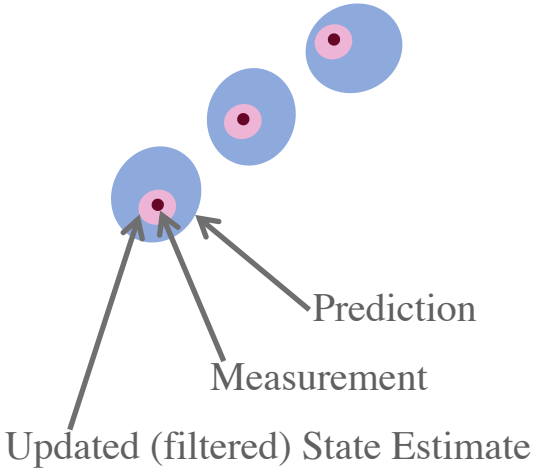
- **Hidden Markov model**
  - State is discrete, no internal structure
  - Observation is discrete
  - State transition is discrete Markov chain
  - Many applications in pattern recognition: speech recognition, language understanding, image understanding…
- **Kalman filter**
  - State is multivariate Gaussian
  - Observation is multivariate Gaussian
  - State transition is linear
  - Many applications in tracking and filtering
- **Dynamic Bayesian network**
  - State is usually discrete and multivariate
  - Observation is usually discrete, can be multivariate
  - Factored state transition probability distribution is represented as graph plus local distributions
  - Many applications in robotics, artificial intelligence, multi-source fusion
- **Partially dynamic Bayesian network**
  - State includes one or more static (not temporally changing) variables

If continuous nodes are allowed then all these examples are special cases of partially dynamic Bayesian networks

Kalman Filter

- Widely applied model for time-varying continuous process measured at regular time intervals
  - "Filter" noise to find best estimate of current state given measurements
  - Predict state at time of next measurement
- Unobservable system state $\underline{s}_t$ at time t is a real vector
- Measurement $\underline{z}_t$ at time t depends on system state
- Control input $\underline{u}_t$ at time t affects movement
- At each time step a recursive algorithm applies Bayesian inference with normal model to:
  - Estimate current state $\underline{s}_t$ given observations $\underline{z}_1, \ldots, \underline{z}_t$
  - Predict next state



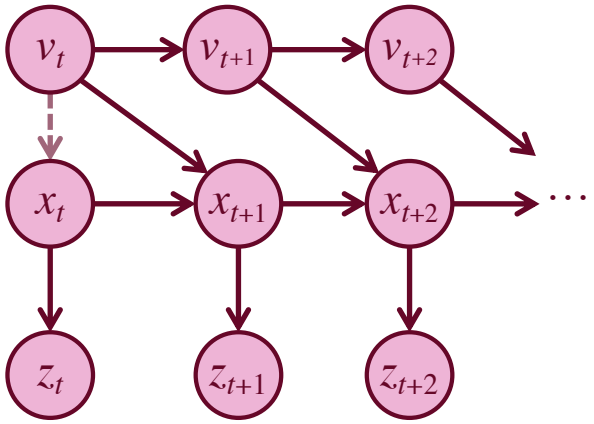


Applications of Kalman Filter

- Kalman filter is applied to a wide range of problems where we need to track moving objects
  - Fitting and predicting economic time series
  - Robot navigation
  - Tracking hands, faces, heads in video imagery
  - Tracking airplanes, missiles, ships, vehicles …
- There are many enhancements and extensions
  - Incorporating non-Gaussian error distributions
  - Incorporating non-linear movement equations
  - Handling maneuvering tracks
  - Tracking multiple objects
    - » Data association – which object goes with which track?
    - » Hypothesis management – which data association hypotheses have enough support to merit attention?
    - » Track initiation and deletion
    - » Spurious measurements not due to any tracks
    - » Incorporating information about object type
  - Missing data and non-regular time measurement intervals
  - Fusing information from multiple sensors

Details: Simple 1-Dimensional Kalman Filter with no Control

- State: position and velocity $\underline{s}_t = (x_t, v_t)$
- Initial state (x_1, v_1) is known
- Evolution equations:
  - $v_t \mid v_{t-1} \sim \mathrm{N}(v_{t-1}, \tau^2)$
  - $x_t \mid x_{t-1}, v_{t-1} \sim \mathrm{N}(x_{t-1} + v_{t-1}, \sigma^2)$
  - $z_t \mid x_t \sim \mathrm{N}(x_t, \xi^2)$
- At time $t > 1$, conditional on $z_{1:t-1}$ the current state variables are normally distributed
  - $v_t \mid z_{1:t-1}$ has mean m_t and variance φ_t^2
  - $x_t \mid v_t, z_{1:t-1}$ has mean $a_t + b_t v_t$ and variance ψ_t^2
- We can use normal-normal conjugate updating to develop recursive updating equations for m_t, a_t, b_t, φ_t^2 and ψ_t^2



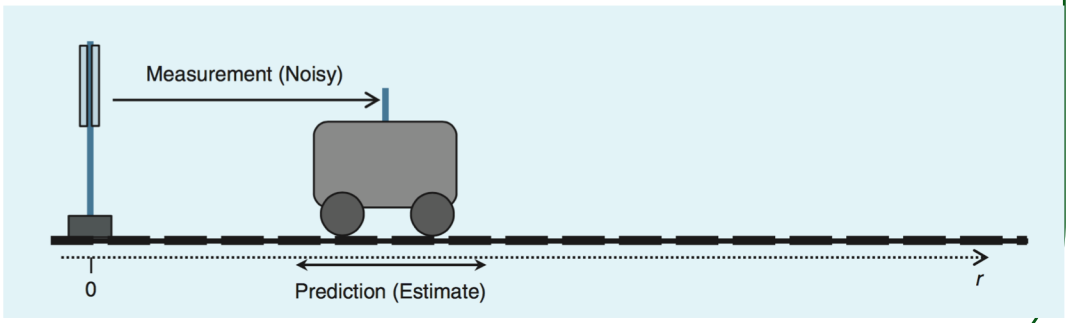
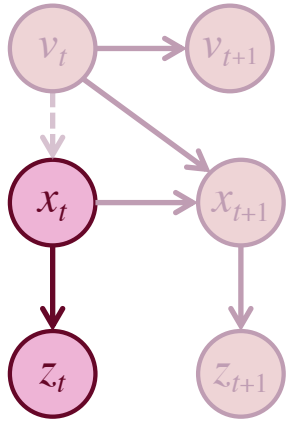


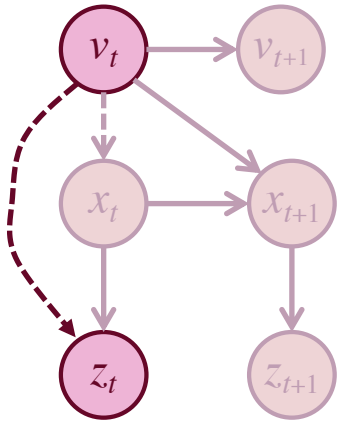
Diagram source: Faragher, Understanding the Basis of the Kalman Filter, *IEEE Signal Processing*, 128, 2012

Bayesian Updating: Position



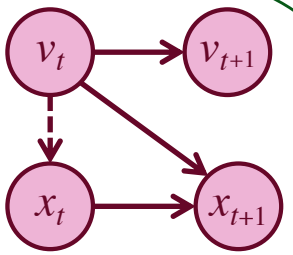
- Reminder of evolution equations:
  - $v_t \mid z_{1:t-1} \sim \mathrm{N}(m_t, \varphi_t^2)$
  - $x_t \mid v_t, z_{1:t-1} \sim \mathrm{N}(a_t + b_t v_t, \psi_t^2)$
  - $z_t \mid x_t \sim \mathrm{N}(x_t, \xi^2)$
- Use Bayesian conjugate updating to find new conditional distribution for position given velocity
  - Distribution of x_t given $z_{1:t}$, v_t is normal
    - Mean $\dfrac{\dfrac{a_t + b_t v_t}{\psi_t^2} + \dfrac{z_t}{\xi_t^2}}{\dfrac{1}{\psi_t^2} + \dfrac{1}{\xi_t^2}} = a_t^* + b_t^* v_t$, where $a_t^* = \dfrac{\dfrac{a_t}{\psi_t^2} + \dfrac{z_t}{\xi_t^2}}{\dfrac{1}{\psi_t^2} + \dfrac{1}{\xi_t^2}}$ and $b_t^* = \dfrac{\dfrac{b_t}{\psi_t^2}}{\dfrac{1}{\psi_t^2} + \dfrac{1}{\xi_t^2}}$
    - Standard deviation $\psi_t^* = \left(\dfrac{1}{\psi_t^2} + \dfrac{1}{\xi_t^2} \right)^{-1/2}$

Bayesian Updating: Velocity



- Reminder of evolution equations:
  - $v_t \mid z_{1:t\text{-}1} \sim \mathrm{N}(m_t, \varphi_t^2)$
  - $x_t \mid v_t, z_{1:t\text{-}1} \sim \mathrm{N}(a_t + b_t v_t, \psi_t^2)$
  - $z_t \mid x_t \sim \mathrm{N}(x_t, \xi^2)$
- Use Bayesian conjugate updating to find new distribution for velocity
  - Integrate out x_t to get distribution of z_t given v_t and $z_{1:t\text{-}1}$
    - $z_t \mid v_t, z_{1:t\text{-}1} \sim \mathrm{N}(a_t + b_t v_t, \xi^2 + \psi_t^2)$
  - Reformulate measurement as $y_t = (z_t - a_t)/b_t$
  - Conditional distribution of y_t given v_t is
    - $y_t \mid v_t, z_{1:t\text{-}1} \sim \mathrm{N}(v_t, (\xi^2 + \psi_t^2)/b_t^2)$
  - Distribution of v_t given $z_{1:t\text{-}1}, y_t$ is normal
    - Mean $m_t^* = \dfrac{\dfrac{m_t}{\varphi_t^2} + \dfrac{y_t b_t^2}{\xi_t^2 + \psi_t^2}}{\dfrac{1}{\varphi_t^2} + \dfrac{b_t^2}{\xi_t^2 + \psi_t^2}}$, Standard deviation $\varphi_t^* = \left(\dfrac{1}{\varphi_t^2} + \dfrac{b_t^2}{\xi_t^2 + \psi_t^2} \right)^{-1/2}$
  - This is also the distribution of v_t given $z_{1:t}$

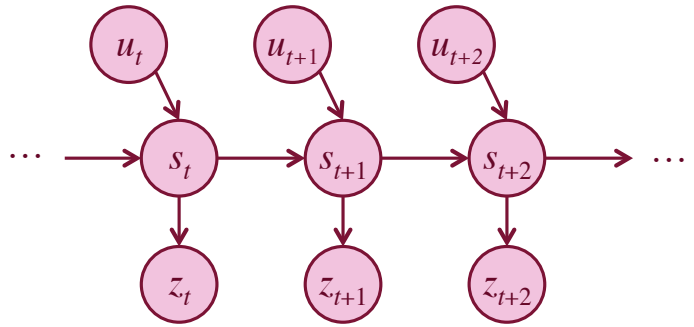
Predicting the Next Step

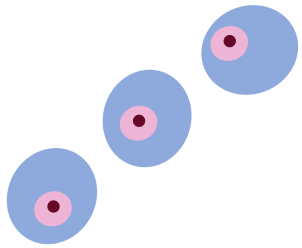


- Distribution of $\underline{s}_t = (x_t, v_t)$ given $z_{1:t}$ is normal
  - $v_t \mid z_{1:t} \sim \mathrm{N}(m_t^*, (\varphi_t^*)^2)$
  - $x_t \mid v_t, z_{1:t-1} \sim \mathrm{N}(a_t^* + b_t^* v_t, (\psi_t^*)^2)$
- Distribution of $\underline{s}_{t+1} = (x_{t+1}, v_{t+1})$ given v_t, x_t is independent of $z_{1:t}$ and normal
  - $v_{t+1} \mid v_t \sim \mathrm{N}(v_t, \tau^2)$
  - $x_{t+1} \mid x_t, v_t \sim \mathrm{N}(x_t + v_t, \sigma^2)$
- Marginalizing out v_t and x_t gives the distribution for (x_{t+1}, v_{t+1}) given $z_{1:t}$
  - $v_{t+1} \mid z_{1:t} \sim \mathrm{N}(m_{t+1}, \varphi_{t+1}^2)$, where $m_{t+1} = m_t^*, \;\; \varphi_{t+1}^2 = (\varphi_t^*)^2 + \tau^2$
  - $x_{t+1} \mid v_t, z_{1:t} \sim \mathrm{N}(a_{t+1} + b_{t+1} v_{t+1}, \psi_{t+1}^2)$, where $a_{t+1} = a_t^*, \;\; b_{t+1} = b_t^*, \;\; \psi_{t+1}^2 = (\psi_t^*)^2 + \sigma^2$

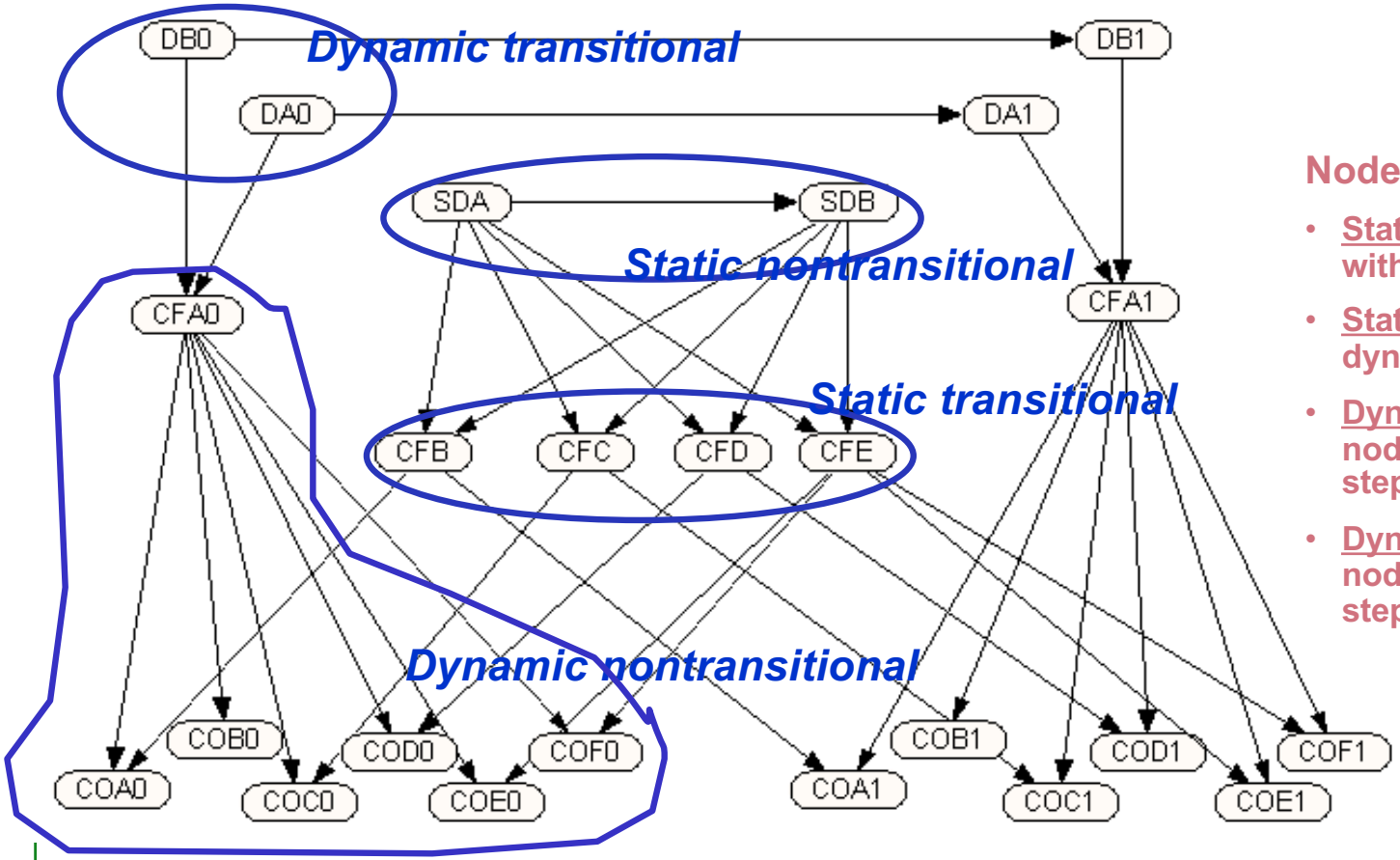
Summary: Kalman Filter

- The Kalman filter was invented by Rudolf Kalman in 1960-61
- It is widely applied to model time-varying real-valued process measured at regular time intervals
  - "Filter" noise to find best estimate of current state given measurements
  - Predict state at time of next measurement
- We examined a simple 1-dimensional problem with no control input
- The algorithm operates recursively as follows
  - Filtering: From prediction of current state (prior given measurements prior to current time) and measurement (likelihood) use conjugate Bayesian updating to find posterior distribution given measurements up to and including current time
  - Prediction: Use marginalization to find predictive distribution of next state given measurements up to and including current state





Partially Dynamic Bayesian Networks



Node types:

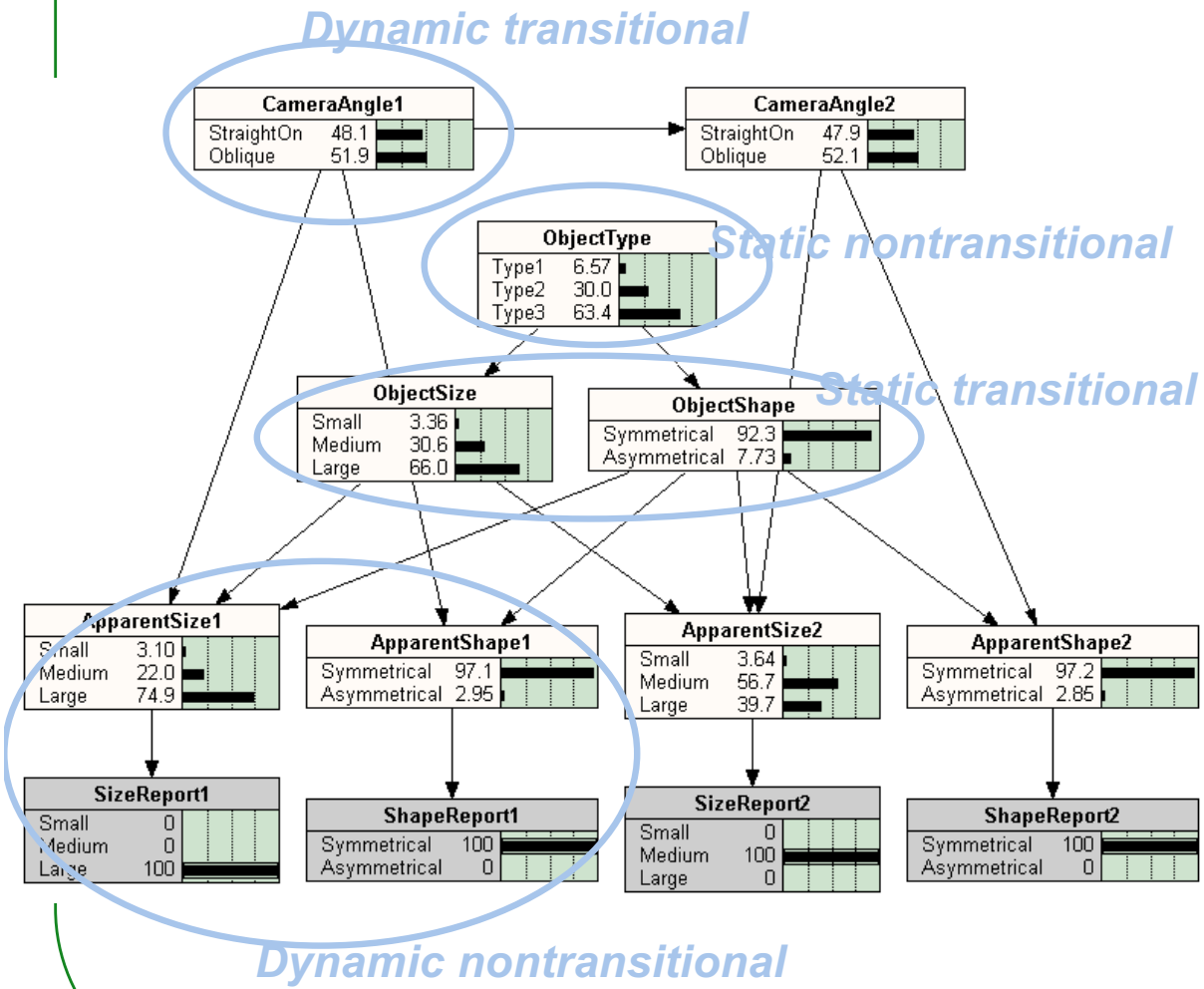
- Static nontransitional - static nodes with no dynamic children
- Static transitional - static nodes with dynamic children
- Dynamic nontransitional - dynamic nodes with no children in next time step
- Dynamic transitional - dynamic nodes with children in next time step

- **PDBN has static as well as dynamic nodes**
- **Most DBN theory and algorithms assume all variables are dynamic**
- **Static nodes can be exploited for efficiency but may degrade accuracy in algorithm not specialized to handle static nodes**

PDBN Inference - Exact Rollup

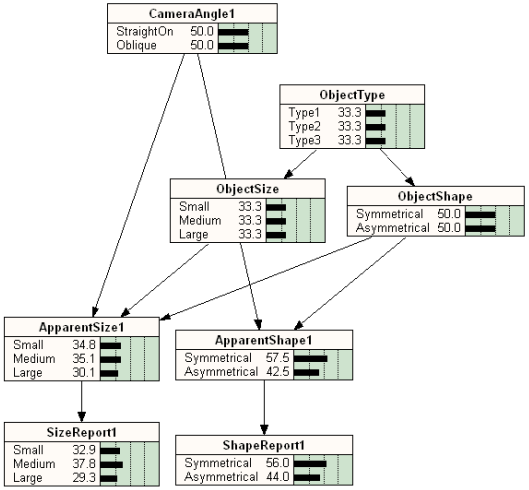
- Filtering in PDBNs works similarly to a Kalman filter
- We keep information on 2 timesteps
  - All the current dynamic nodes
  - All the current static nodes
  - The dynamic transitional nodes
- Two-stage rolling inference:
  1. Initialize
     - Set up BN at first timestep
     - Set current timestep to 1
  2. Apply evidence at current timestep
  3. Update beliefs
  4. Absorb all nodes except transitional nodes
  5. Roll BN forward to next timestep
     - Add next time step dynamic nodes
  6. Return to Step 2

Example PDBN

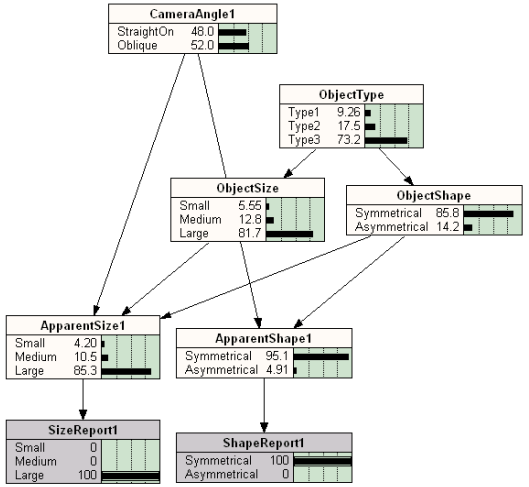


- There are 3 types of objects
  - Type 1 objects are usually small and asymmetrical
  - Type 2 objects are usually medium-sized and may be symmetrical or asymmetrical
  - Type 3 objects are usually large and symmetrical
- Objects can be viewed straight on or obliquely
  - Camera angle is unknown and changes with time
- Asymmetrical objects viewed obliquely may look smaller than actual size and may appear symmetrical

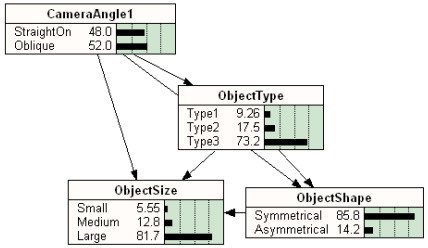
Example: Exact Rollup
(First Evidence)



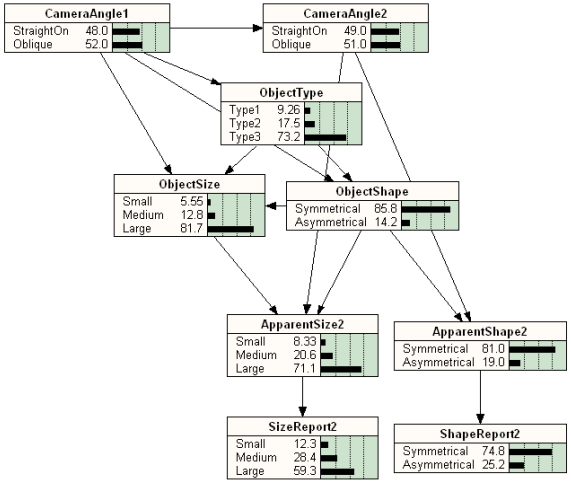
Initialize PDBN at Time Step 1



1. **Apply evidence at current time step**
2. **Update beliefs on all unobserved RVs**

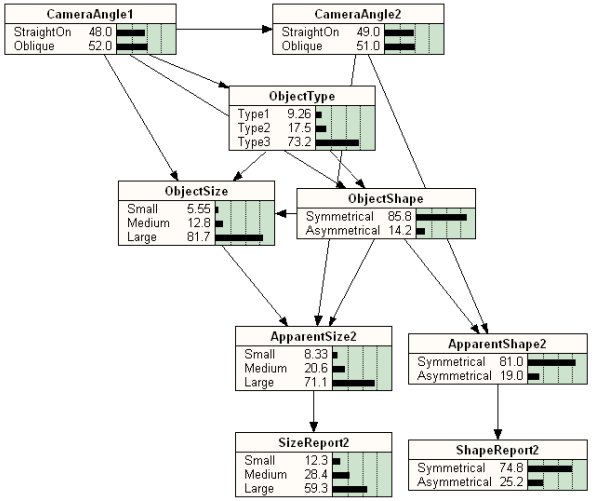


3. **Absorb all RVs not part of past expression**

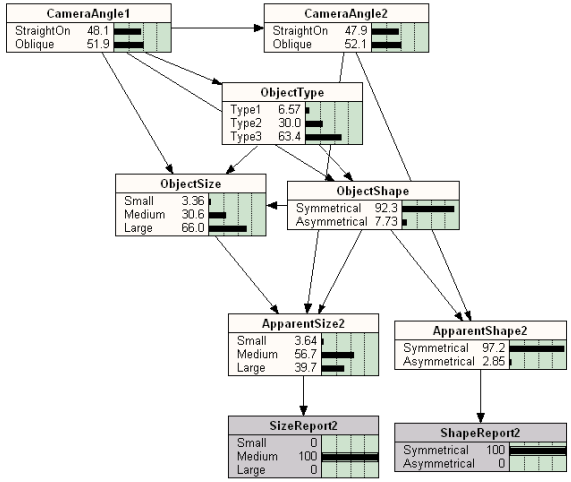


4. **Extend to next time step**

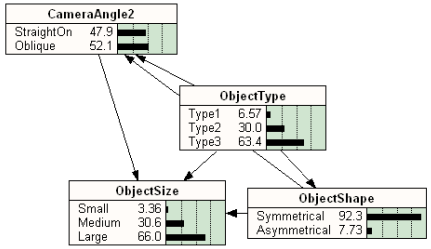
Example: Exact Rollup
(Next Evidence)



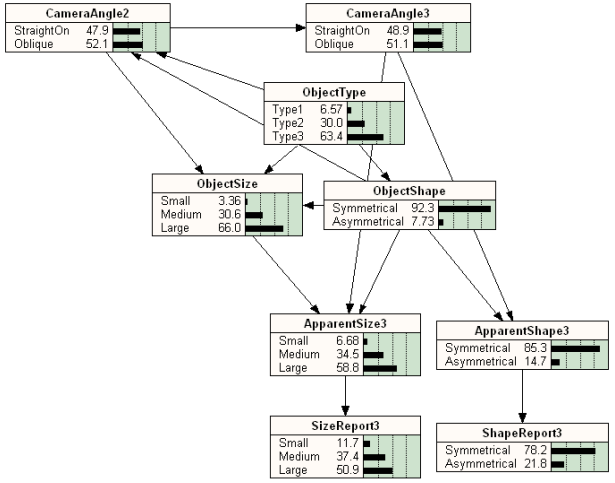
Begin with current 2-PDBN



1. **Apply evidence at current time step**
2. **Update beliefs on all unobserved RVs**

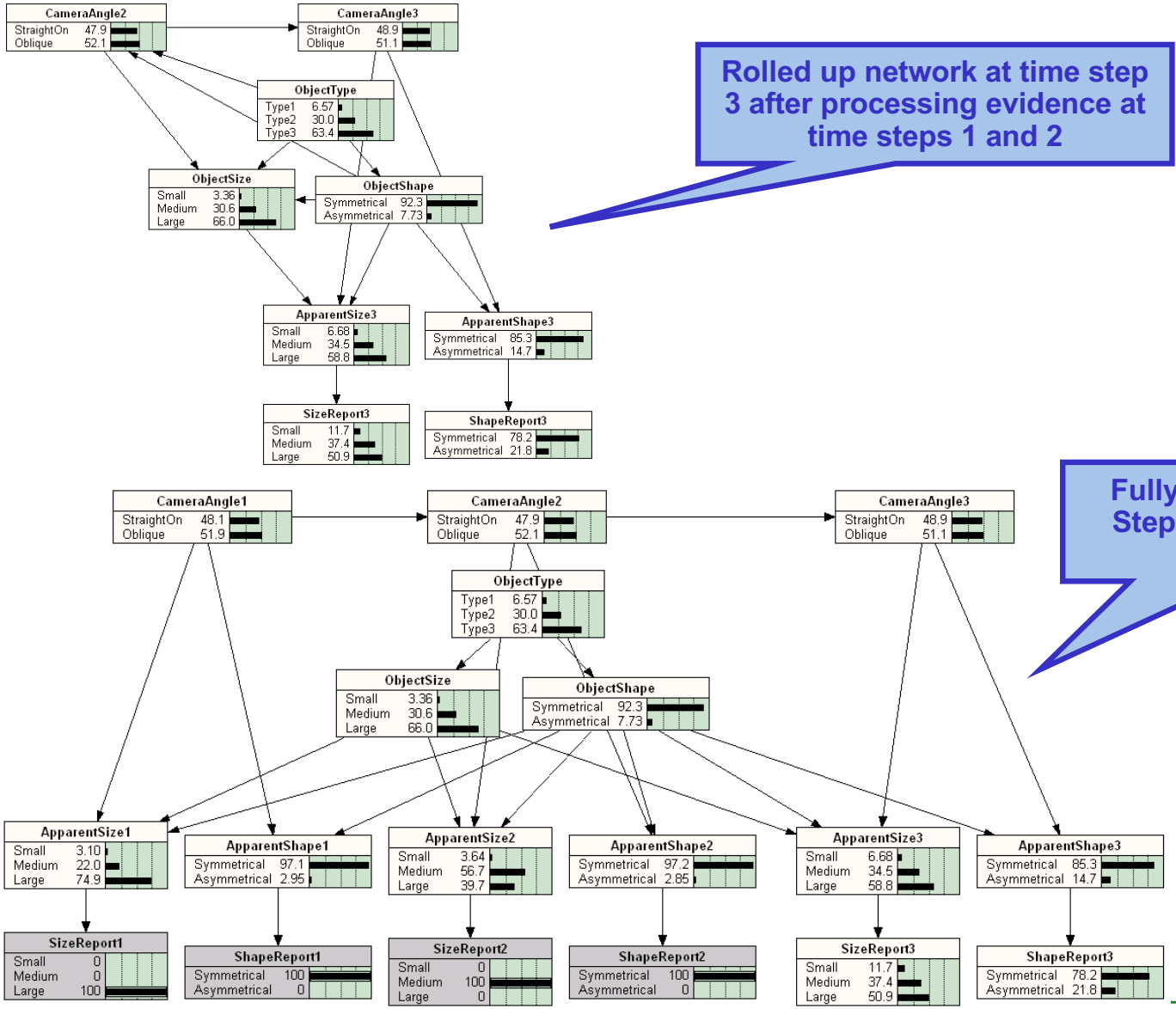


3. **Absorb all RVs not part of past expression**



4. **Extend to next time step**

Comparison



Rolled up network at time step 3 after processing evidence at time steps 1 and 2

Fully Extended 3 Time Step PDBN (answer is same)

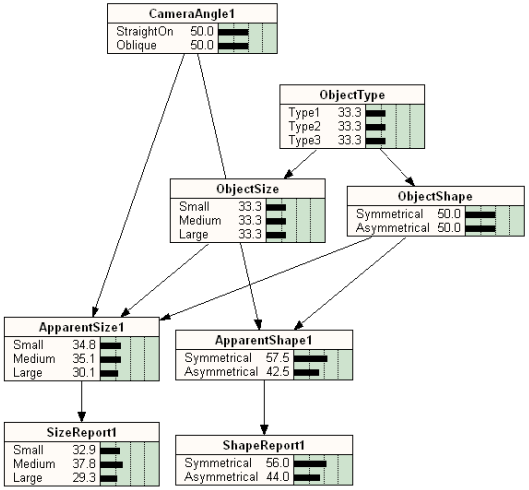
Approximate PDBN Inference

- **Exact rollup is intractable for all but the most simple PDBNs**
  - **Even for sparsely connected DBN usually cannot construct sparse junction tree**
  - **When there are many variables inference is intractable**
- **Approximation methods**
  - **Boyen-Koller**
    - » **Deterministic approximation**
    - » **Replaces exact past expression with simplified past expression**
  - **Particle filter**
    - » **Stochastic approximation**
    - » **Approximates past expression with sample of "particles"**
  - **Additional algorithms under development**

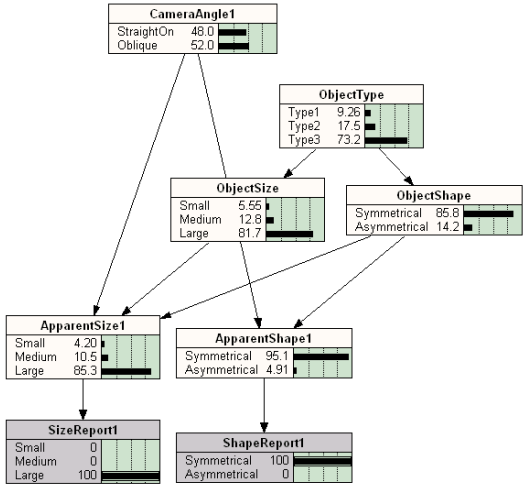
Boyen-Koller Approximation

- **Works like rollup except we approximate marginal distribution of X_t given past and current evidence by simpler structure**
- **Keep information on 2 timesteps**
  - **All the current dynamic nodes**
  - **An approximate "past expression" that summarizes all previous evidence**
  - **The approximate past expression contains static and dynamic transitional nodes but structure is simpler than exact rollup past expression**
- **The Boyen-Koller algorithm**
  1. **Get observations on dynamic nontransitional evidence nodes**
  2. **Use Bayesian inference to update beliefs on unobserved nodes**
  3. **Compute new approximate past expression**
     - **Use same structure as approximate past expression from previous timestep**
     - **Replace beliefs with updated marginal beliefs**
  4. **Roll BN forward to next timestep, keeping only the new past expression**
  5. **Return to Step 1**

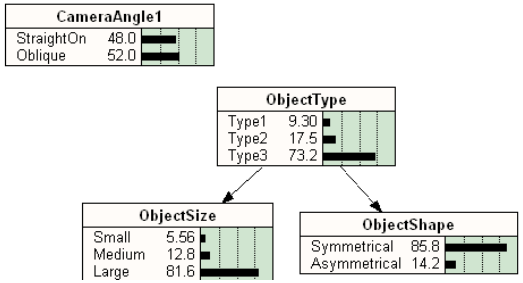
Example: Fully Factored BK
(First Evidence)



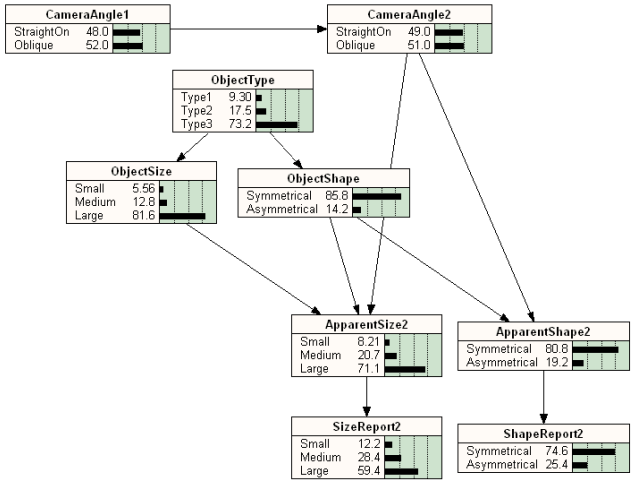
Initialize PDBN at Time Step 1



1. Apply evidence at current time step
2. Update beliefs on all unobserved RVs

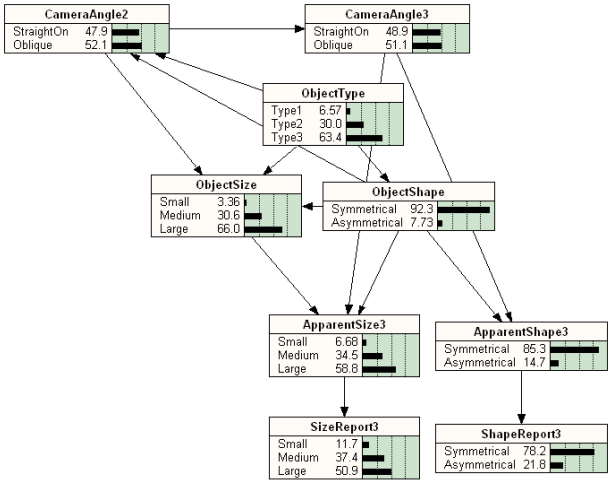


3. Remove RVs not part of past expression & adjust belief tables in past expression to conform to current conditional probabilities

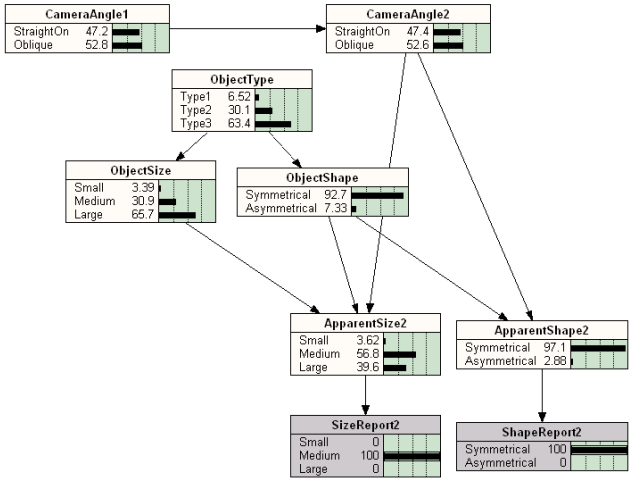


4. Extend to next time step

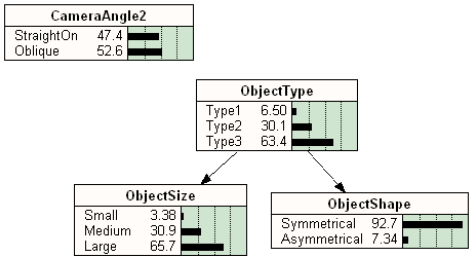
Example: Fully Factored BK
(Next Evidence)



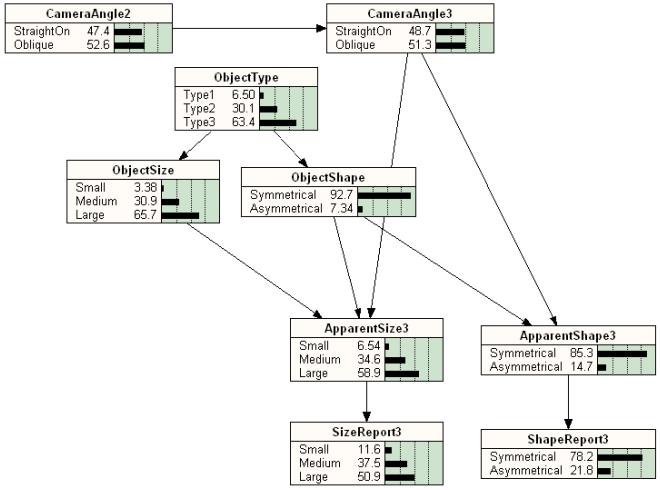
Begin with current 2-PDBN



1. Apply evidence at current time step
2. Update beliefs on all unobserved RVs

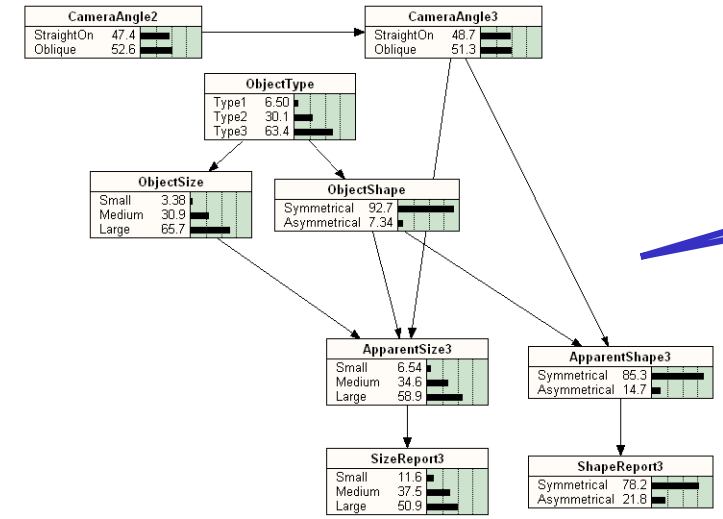


3. Absorb all RVs not part of past expression

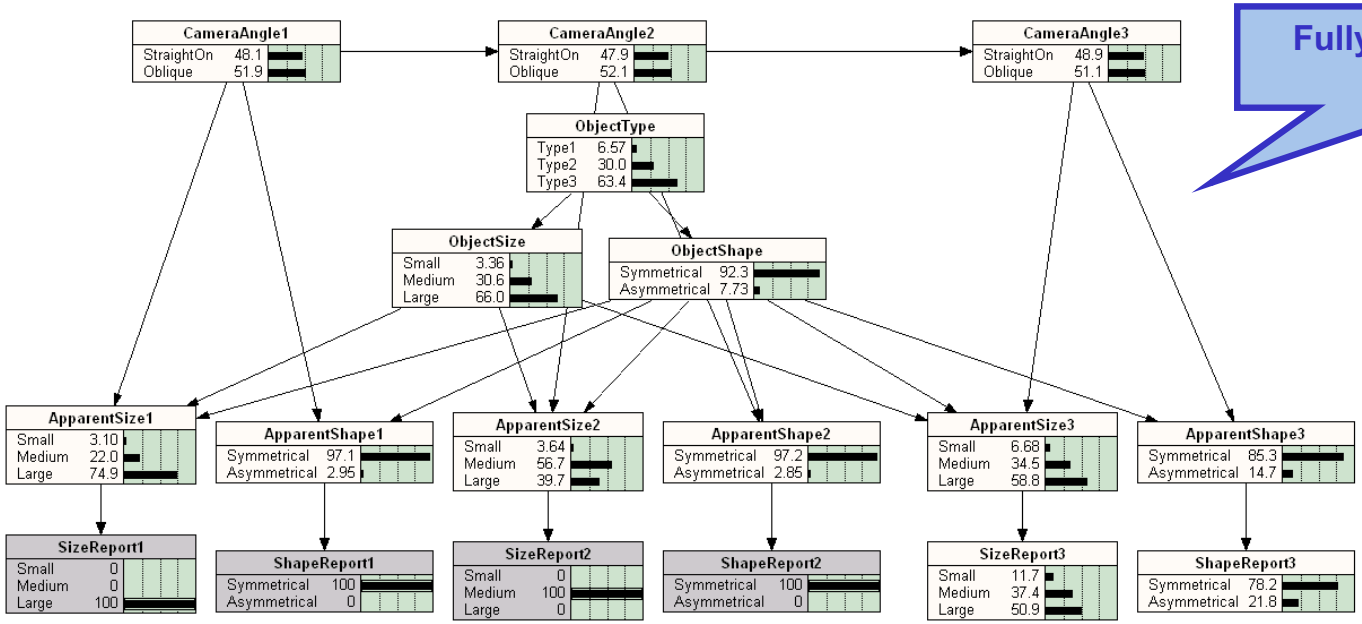


4. Extend to next time step

Comparison



BK approximate network at time step 3 after processing evidence at time steps 1 and 2



Fully Extended 3 Time Step PDBN

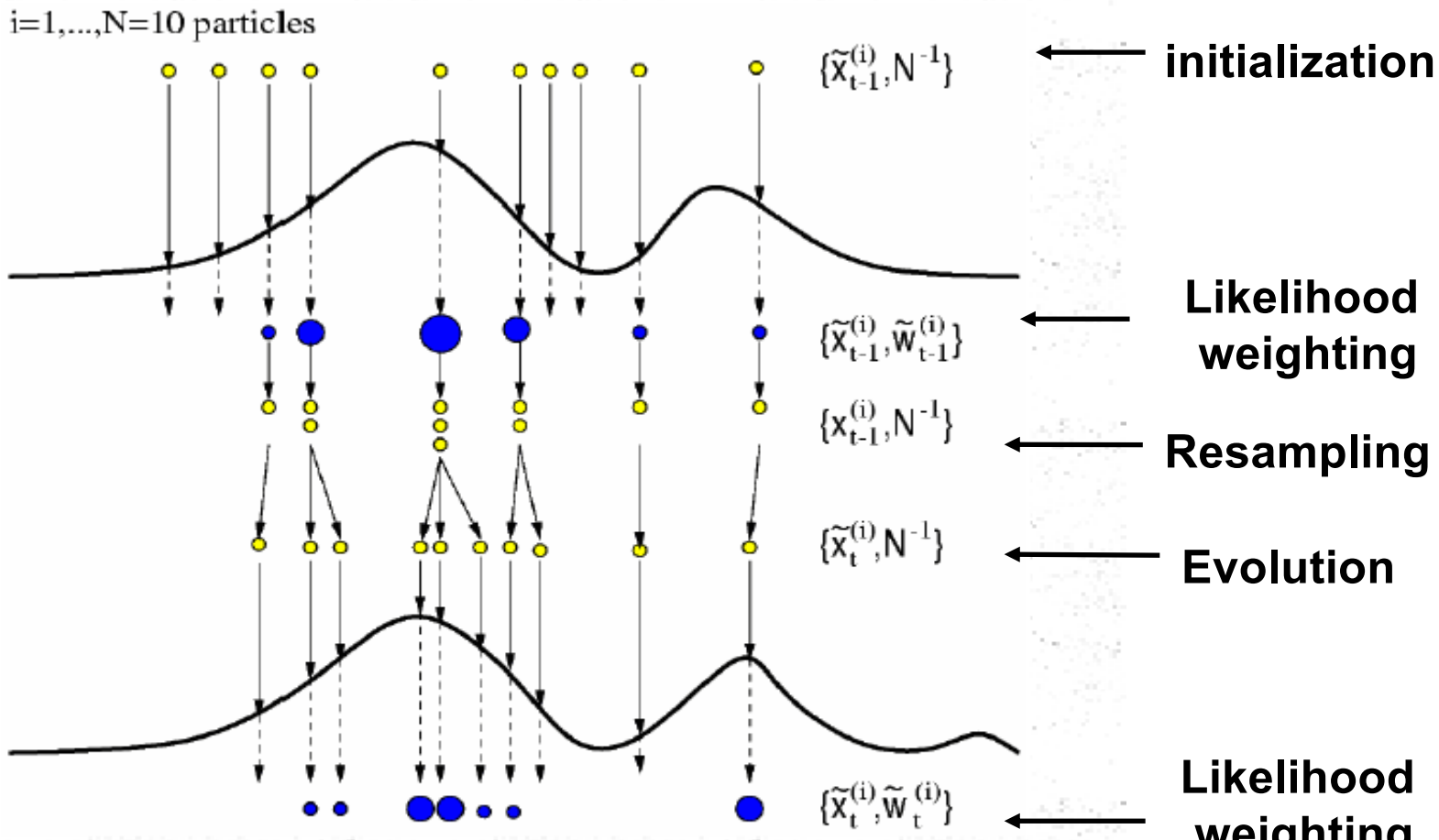
Boyen-Koller Theory

- If DBN has a unique stationary distribution, the forward prediction operation "shrinks" both the approximate and exact belief states toward the stationary distribution of the DBN (if it has one) and closer to each other
- Effect of errors due to previous approximations decreases exponentially
- Overall error remains bounded indefinitely
- Approximation error depends on how well the approximate belief state matches the exact belief state
- Important issues
  - How to choose the structure for the approximate belief state
    - » Tractable
    - » Closely approximates exact state
  - Approximation bounds do not apply when there are static nodes!

Factored Frontier

- Similar to Boyen-Koller
  - Boyen-Koller does exact prediction followed by marginalization
- Factored frontier approximates the prediction step using factored distributions
  - This amounts to applying junction tree propagation to "loopy" tree but ignoring the multiple connections
- Factored frontier is equivalent to single iteration of "loopy belief propagation"
- Can be improved by iterating loopy belief propagation

Particle Filter Basic Idea



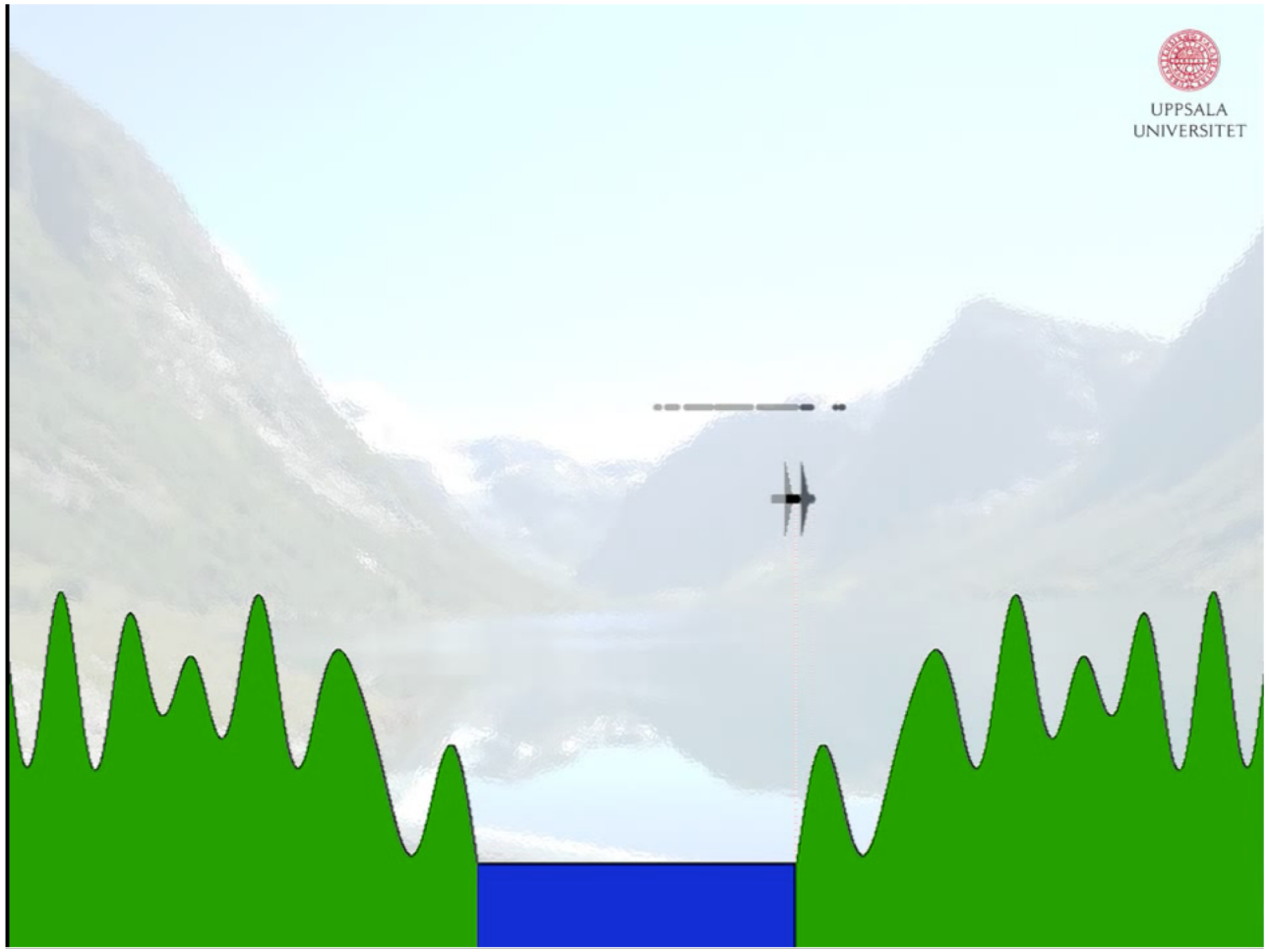
(graphics taken from Van der Merwe et al.)

Particle Filter Algorithm

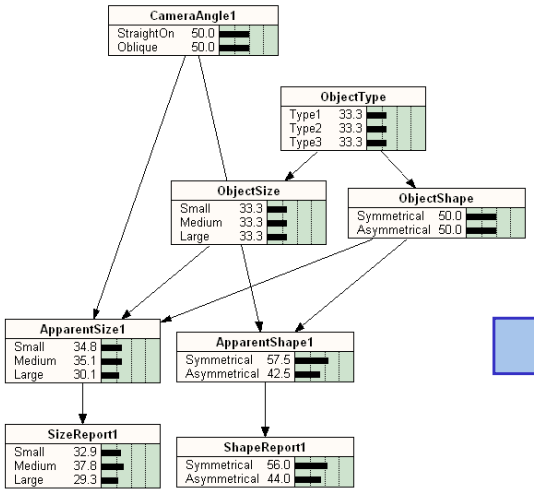
- **Notation:**
  - $X_t = (X_{1t}, X_{2t}, \ldots, X_{kt})$ is time step t of k-variable DBN
  - There are N particles x_{it}, i=1,…N
  - Each particle assigns a value x_{ijt} to each random variable X_{jt} at time step t
  - ev_t is evidence at time t
- **Basic particle filter:**
  - Begin with a sample (equally weighted) of particles $x_{1t}, x_{2t}, \ldots x_{Nt}$
  - For each particle $x_{it} = (x_{i1t}, x_{i2t}, \ldots, x_{ikt})$:
    - » Project forward to obtain a distribution $P(X_{ij(t+1)} \mid x_{i,pa(j)t})$ for $X_{ij(t+1)}$ at time t+1
    - » Sample trial value $x'_{ij(t+1)}$ from $P(X_{ij(t+1)} \mid x_{i,pa(j)t})$
  - Reweight particles based on evidence
    - » $w_{i(t+1)} \propto P(ev_{t+1} \mid x_{i(t+1)})$
    - » Normalize to sum to 1
  - Resample (this step keeps the weights from getting too skewed)
    - » Sample N particles with replacement from the collection of trial values
    - » Use weights as probabilities

Visualizing the Particle Filter

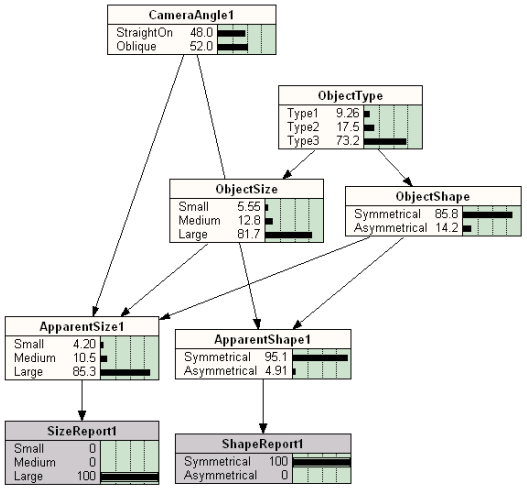
- A nice intuitive explanation of the particle filter (with no math):
  - https://www.youtube.com/watch?v=aUkBa1zMKv4



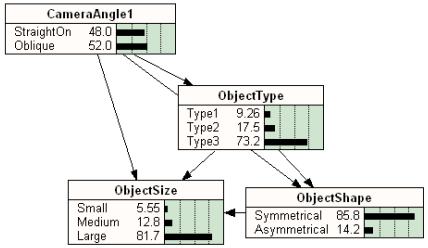
PDBN Particle Filter



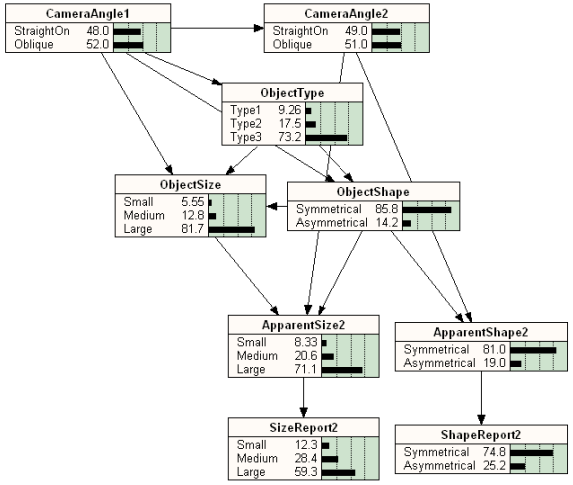
Generate initial sample of static and first timestep dynamic nodes



1. Apply evidence at current time step
2. Calculate weights of particles



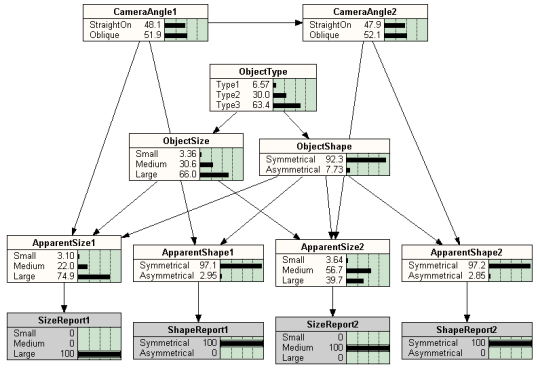
3. Sample particles with replacement; probability proportional to weights
4. Only past expression nodes are needed



4. Sample next timestep for each particle
5. Go to 1

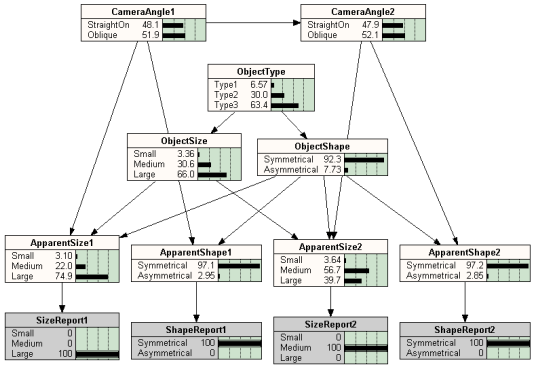
- PF maintains collection of weighted particles
- Each particle is a single realization of all nodes in the BN
- Belief bars should be interpreted as histograms of weighted particle counts

Step-by-Step Example
(1 of 3)



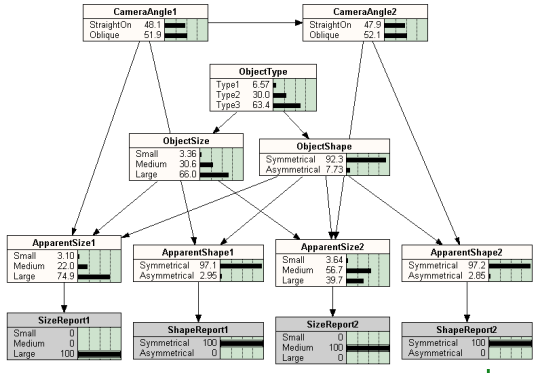
1. Sample values for Static Nodes
    a. For i=1:numParticles
        i. Randomly draw ObjectType(i) from Pr(ObjectType)
        ii. Randomly draw ObjectSize(i) from Pr(ObjectSize | ObjectType)
        iii. Randomly draw ObjectShape(i) from Pr(ObjectShape | ObjectType)
    end
    b. Set Particle(i) <- [ObjectType(i), ObjectSize(i), ObjectShape(i)]
2. Sample values for non-evidence dynamic nodes, using initial distributions for dynamic transitional nodes
    a. For i=1:numParticles
        i. Randomly draw CameraAngle1(i) according to initial distribution Pr(CameraAngle1)
        ii. Randomly draw ApparentSize1(i) and ApparentShape1(i) according to
            Pr(ApparentSize | CameraAngle1(i), ObjectSize(i), ObjectShape(i)) and
            Pr(ApparentShape | CameraAngle1(i), ObjectShape(i))
    end
    b. Set Particle(i) <- Particle(i) + [CameraAngle1(i), ApparentSize1(i), ApparentShape1(i)]
3. Set T=1

Step-by-Step Example
(2 of 3)



4. Calculate weights
    a. Set evidence values szrT and shprT for SizeReportT and ShapeReportT. That is:
        i. szr1=Large and shpr1=Symmetrical;
        ii. szr2 = Medium and shpr2=Symmetrical;
        iii. Evidence values at times greater than 2 were not given in the example.
    b. For i=1:numParticles
        i. Calculate unnormalized weight for each particle:
            Set $w_{Unnorm}(i)$ <- Pr(szrT | ApparentSizeT(i)) * Pr(shprT | ApparentShapeT(i))
    end
    c. For i=1:numParticles
        i. Calculate normalized weight: Set $w(i)$ <- $w_{Unnorm}(i)/SumOverj(w_{Unnorm}(j))$
5. Resample
    a. For i=1:numParticles
        i. Draw ResampleIndex(i) between 1 and numParticles with probability w(i) of drawing i
        ii. Set NewParticle(i) <- Particle(ResampleIndex(i))
    end
    b. For i=1:numParticles
        i. Set Particle(i)=NewParticle(i)
    end

Step-by-Step Example
(3 of 3)



6. If T is last time step, exit. Else roll forward to T <- T+1
7. Sample values for non-evidence dynamic nodes, using DBN distributions for dynamic transitional nodes
    a. For i=1:numParticles
        i. Randomly draw CameraAngleT(i) according to Pr(CameraAngleT | CameraAngleT-1)
        ii. Randomly draw ApparentSizeT(i) and ApparentShapeT(i) according to
            Pr(ApparentSize | CameraAngleT-1(i), ObjectSize(i), ObjectShape(i)) and
            Pr(ApparentShape | CameraAngleT-1(i), ObjectShape(i))
    end
    b. Set Particle(i) <- Particle(i) + [CameraAngleT(i), ApparentSizeT(i), ApparentShapeT(i)]
8. Go to Step 3.

Weighted Monte Carlo: BasicTheory

- **Our goal is to estimate a target integral (or sum):***

$$t = \int_x a(x) d\mu(x)$$

- **We can calculate (or estimate) *a*(*x*) but not the integral**
- **We can easily simulate observations from probability distribution *q*(*x*)**
- **Re-express our target integral (or sum):***

$$t = \int_x \frac{a(x)}{q(x)} q(x) d\mu(x)$$

- **Sample observations $x_1, \ldots, x_n$ from *q*(*x*)**
- **Estimate *t* by:**

$$\hat{t} = \frac{1}{n} \sum_{i=1}^{n} \frac{a(x_i)}{q(x_i)}$$

** This is generic notation that applies to sums and/or integrals over arbitrary sets. The notation $d\mu(x)$ stands for the "unit of measure" over which we are integrating or summing. Read it as dx in a univariate integral, $dx_1 \ldots dx_n$ in a multivariate integral, or a "point mass" at each element of a discrete set (in the latter case the integral symbol should be read as Σ_x).*

See also: approximate inference lecture

Application: A Simple Particle Filter

- **Assume:**
  - State space model has state transition distribution $f(x_t \mid x_{t-1})$ and observation distribution $g(y_t \mid x_t)$:
  - Collection of equally weighted particles $\{x^{(i)}_{t-1}\}_{i=1,..n}$ provides an estimate of the $f_{est}(x_{t-1} \mid y_{1:t-1})$ distribution of states given past evidence
  - We want to estimate $f_{est}(x_{t-1} \mid y_{1:t})$
- **Sample $\tilde{x}_t^{(i)}$ from the state transition distribution $f(x_t \mid x^{(i)}_{t-1})$ and weight each particle by $w^{(i)}_t \propto g(y_t \mid \tilde{x}_t^{(i)})$ (normalize weights to sum to 1)**
- **Use weighted particles for estimating functions of state**
- **Resample to move to the next time step:**
  - Sample $\tilde{x}_t^{(i)}$ (with replacement) with probability $w^{(i)}_t$ to obtain a collection $\{x^{(i)}_t\}_{i=1,..n}$ of equally weighted particles
- ***Why does this work?***

Particle Filter: Estimation

- **Goal: estimate expected value of function of current state:**

$$\bar{t} = \int t(x_t) f(x_t \mid y_{1:t})$$

- **Re-express the integral:** $\bar{t} = \int_{x_t} \int_{x_{t-1}} \frac{t(x_t) \Pr(x_{t-1} \mid y_{1:t-1}) f(x_t \mid x_{t-1}) g(y_t \mid x_t)}{g(y_t)} dx_{t-1} dx_t$

- **Approximate numerator and denominator:**

$$\int_{x_t} \int_{x_{t-1}} t(x_t) \Pr(x_{t-1} \mid y_{1:t-1}) f(x_t \mid x_{t-1}) g(y_t \mid x_t) dx_{t-1} dx_t \approx \frac{1}{n} \sum_i t(\tilde{x}_t^{(i)}) g(y_t \mid \tilde{x}_t^{(i)})$$

$$\int_{x_t} \int_{x_{t-1}} \Pr(x_{t-1} \mid y_{1:t-1}) f(x_t \mid x_{t-1}) g(y_t \mid x_t) dx_{t-1} dx_t \approx \frac{1}{n} \sum_i g(y_t \mid \tilde{x}_t^{(i)})$$

- **Take ratio:**

$$\bar{t} \approx \frac{\sum_i t(\tilde{x}_t^{(i)}) g(y_t \mid \tilde{x}_t^{(i)})}{\sum_i g(y_t \mid \tilde{x}_t^{(i)})} = \sum_i w_t^{(i)} t(\tilde{x}_t^{(i)})$$

$a(x_t) = t(x_t) f(x_t \mid x_{t-1}^{(i)}) g(y_t \mid x_t)$
$q(x_t) = f(x_t \mid x_{t-1}^{(i)})$

$a(x_t) = f(x_t \mid x_{t-1}^{(i)}) g(y_t \mid x_t)$
$q(x_t) = f(x_t \mid x_{t-1}^{(i)})$

Estimate before resampling

Particle Filter: Resampling

- We could avoid resampling and carry weighted particles.
- Weight update without resampling is:
  - $w^{(i)}_t \propto w^{(i)}_{t-1}\, g(y_t \mid x^{(i)}_t)$
- In practice the weights become very unbalanced after a few iterations: most are very near zero and a few have larger values
- Resampling preserves expected values but avoids unbalanced weights

Particle Filter: Limiting Behavior

- Under reasonable conditions on the importance distribution, weighted Monte Carlo estimates satisfy a central limit theorem
- As number of samples becomes large
  - Estimate converges with probability 1 to the integral being estimated
  - Deviations of estimate from true value have approximately a Gaussian distribution
- These results apply as number of particles becomes large *while holding the number of time steps fixed*
- We are often interested in holding number of samples fixed and letting number of time steps become large
- *There are no large-sample results for this case!*

Modifications of Basic Algorithm

- **Importance sampling**
  - Sample trial value $x'_{ij(t+1)}$ from distribution other than $P(x_{ij(t+1)} \mid x_{ipaji)t})$ and weight the estimate appropriately
  - The optimal sampling distribution would be $P(x_{ij(t+1)} \mid x_{ipa(i)t}, ev_t)$ but this may be intractable
  - Poorly chosen importance distribution can make things worse
- **Rao-Blackwellization**
  - Sometimes we can disconnect the BN into tractable sub-parts by sampling only some of the variables
  - Use temporal rollup for variables not sampled
  - Theory: replacing sampled value by expectation lowers variance of estimator
- **Auxiliary particle filter**
  - Goal is to increase number of particles in useful regions of space but avoid duplicated particles from resampling
  - Resample *before* propagating forward rather than afterward
  - Sample auxiliary value $x^a_{ij(t+1)}$ given x_{ijt}, compute weights given evidence, resample
  - For each $x^a_{ij(t+1)}$ in the resampled collection sample "real" particle $x_{ij(t+1)}$

Importance Sampling

- Instead of sampling from transition distribution $f(x_t \mid x^{(i)}_{t-1})$ we can sample from an an *importance distribution* $q(x_t)$
- To retain correct expected values we adjust the weights:

$$w_t^{(i)} \propto \frac{f(\tilde{x}_t^{(i)} \mid x_{t-1}^{(i)}) g(y_t \mid \tilde{x}_t^{(i)})}{q(\tilde{x}_t^{(i)})}$$

- Performance depends on a good importance distribution
  - The optimal importance distribution is a probability density function proportional to $f(\tilde{x}_t^{(i)} \mid x_{t-1}^{(i)}) g(y_t \mid \tilde{x}_t^{(i)})$
  - Standard particle filter uses $f(\tilde{x}_t^{(i)} \mid x_{t-1}^{(i)})$
  - This can be a poor choice
- Adaptive importance sampling repeats importance sampling, improving the importance distribution on each step

Dynamic Iterative Importance Sampling for PDBNs (Chang)

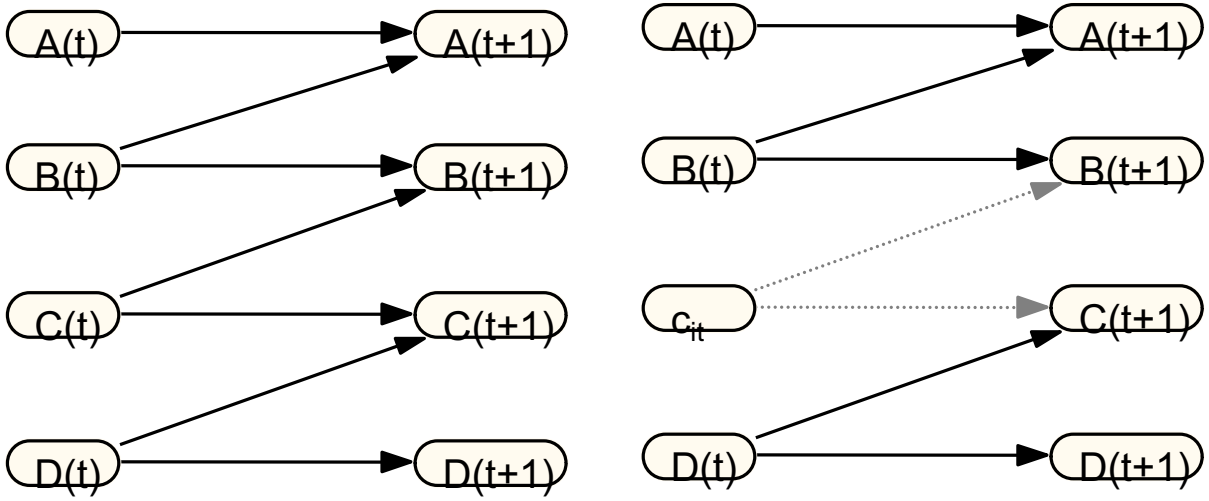
- **Importance distribution:**
  - Same factored structure as original BN
  - Replace original CPTs with *importance CPTs* (ICPTs) estimated from importance samples
  - ICPTs are updated on each time step
- **The algorithm:**
  1. Initialize importance distribution
  2. For number of adaptation steps:
     - Generate samples using current importance distribution
     - Compute weights and update scores
     - Update ICPTs
  3. Roll over to next time step
     - Replace CPTs for transitional nodes with ICPTs
- **Product of ICPTs is good approximation to joint PDF**
- **Empirical results show method is effective except when evidence has extremely small probabilities**

Adaptive Importance Sampling: Caveat

- Good importance distribution appears to be the single most effective way to improve weighted Monte Carlo
- Optimal importance function is proportional to target distribution
- Adaptive importance sampling:
  - Estimate optimal importance function from previous samples
  - Use estimate to generate the next set of samples
- Caution: Estimates from adaptive importance sampling have correct expectation *but may not satisfy a Central Limit Theorem*!
  - Intuition: overfitted estimates can cause extreme weights (cases for which denominator of weight is very small relative to numerator)
  - There are ways to ensure that estimates satisfy CLT (e.g., put small *fixed* weight on uniform CPT in ICPT estimate for DIIS)

Rao-Blackwell Particle Filter

- **We can reduce variance either by:**
  - **Increasing the number of particles**
  - **Replacing a sampling step by an exact computation with the same expectation**
- **When exact computation on part of the problem is cheap, then inserting an exact computation can improve accuracy**



Example:

- **If we know C(t) then we can decompose the computation into two simpler sub-problems, which we may be able to afford to do exactly**
- **We can simulate C(t) and compute distribution of A(t), B(t) and D(t) exactly given C(t)**
- **Variance will be less for a given number of particles than simulating all variables**

Filtering and Estimation

- Inferences are more accurate if we compute distributions before sampling than after
- To infer distribution of jth variable at time t based on Monte Carlo sampling:
  - Roll each equally weighted particle x_{it-1} forward through the transition model to obtain a distribution $P(X_{ji(t-1)}|ev_{[0,..t-1]})$
  - Compute weight $w_{it} \propto P(ev_t \mid x_{it})$
  - Approximate the marginal distribution of jth variable given evidence up to and including t as:

$$P(X_{jt}, ev_{[0,t]}) = \sum_i w_{jt} P(X_{jit} \mid x_{i(t-1)}, ev_{[0,t-1]})$$

- Delayed inference (infer static or unobserved dynamic variable at time t using data up to and including time t+k) is generally more accurate than concurrent inference
  - More information is used
  - May be less accurate if resampling is used because resampling reduces distinct past samples

Challenge: Particle Impoverishment

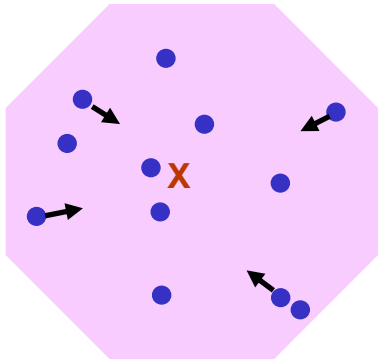
- After several rounds of resampling, typically all particles are descended from a single initial particle
- When there are static nodes or near-deterministic transitions this can cause very poor performance
- Particle impoverishment can result in convergence to local minima of likelihood function and very poor estimates
- There is no asymptotic theory for particle filter as number of timesteps becomes large -- asymptotic theory relates to number of particles becoming large
- There are no guarantees of convergence!!

Dealing with Impoverishment

- More particles (brute force)
  - Usually not very effective when particle impoverishment is severe (especially in case of static variables)
- Regularized particle filter
  - Ordinary particle filter uses discrete approximation to state density
  - Regularized particle filter
    - » Approximate state density at past time step with continuous distribution (often mixture of Gaussians with small standard deviation ("bandwidth")
    - » Resample from approximate density before propagating to next timestep
- Adaptive importance sampling
  - A good importance distribution is the best solution to particle impoverishment
  - Bad importance distribution can make impoverishment much worse
  - Adaptive importance sampling iteratively improves approximation
  - Computation cost is worth it if good importance distribution is found

Particle Impoverishment with Static Parameters

- **Standard PF *cannot recover* from impoverishment of static parameter**
- **Suggested approaches:**
  - **Artificial evolution of static parameter**
    - » **Ad hoc; no justification for amount of perturbation; information loss over time**
  - **Shrinkage (Liu & West)**
    - » **Combines ideas from artificial evolution & kernel smoothing**
    - » **Perturbation "shrinks" static parameter for each particle toward weighted sample mean**
      - **Perturbation holds variance of set of particles constant**
      - **Correlation in disturbances compensates for information loss**
  - **Resample-Move (Gilks & Berzuini)**
    - » **Metropolis-Hastings step corrects for particle impoverishment**
    - » **MH sampling of static parameter involves entire trajectory but is performed less frequently as runs become longer**
- **There is not much literature on empirical performance of these approaches in applications**



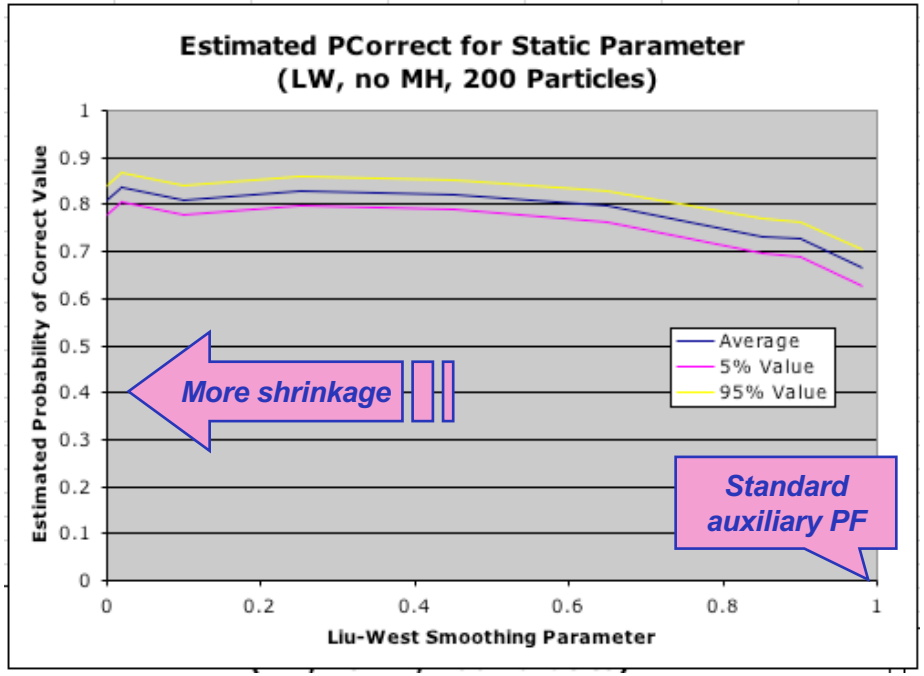
Static Parameter Shrinkage

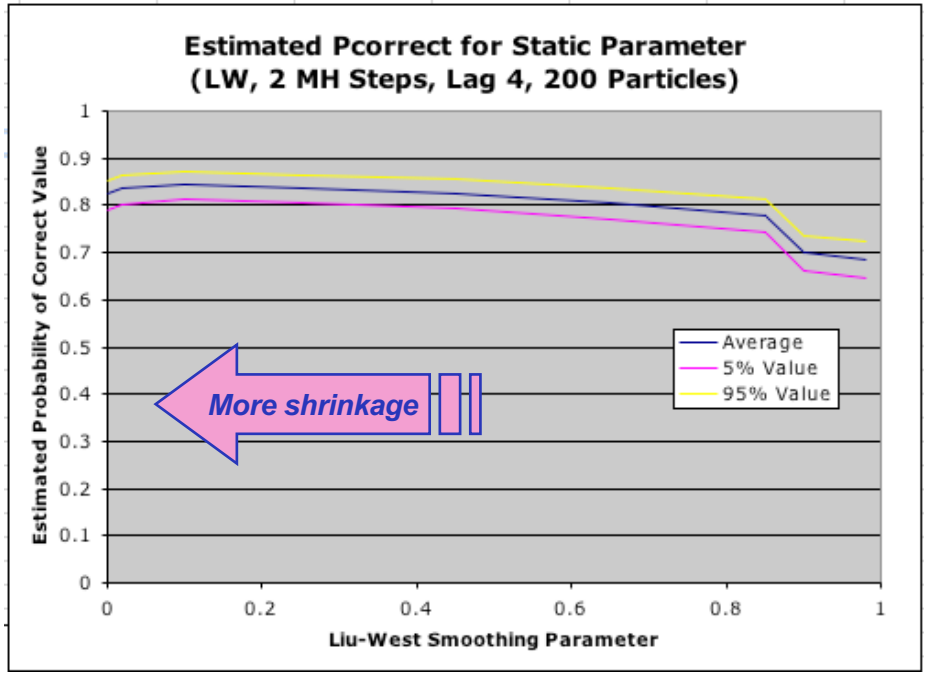
- **Justification:**
  - Ad hoc "jiggling" of static parameter increases variance of estimator
  - This can compensate for reduction in variance due to impoverishment
  - There is no theory to justify how much to jiggle or to evaluate how well the compensation works
- **Shrinkage algorithm - Insert after resampling step of PF:**
  - Estimate posterior distribution of static parameter
  - "Jiggle" static parameter randomly as follows:
    - » Hold static parameter fixed with probability p
    - » Sample from estimated posterior distribution with probability 1-p
- **Avoids overdispersion from ad hoc jiggling**
  - If PF estimate is accurate estimate of posterior distribution then L-W shrinkage will maintain accurate mean & variance
  - If PF estimate gets stuck in local optimum L-W may not help much

Resample-Move Algorithm

- **Keep record of *entire trajectory* of each particle**
  - Current static parameter value
  - Current and past values of all dynamic state variables
  - Current and past observations
- **Insert "move" step after resampling step in PF algorithm**
  - For each particle:
    - » Use a *proposal distribution* to suggest a random change in the particle trajectory (static parameter, present state, and/or past state(s)
    - » Evaluate:
      - Likelihood of new & old trajectories
      - Probability of proposing new from old and old from new
    - » Decide whether to accept or reject change
      - "Better" (more likely) new state increases chance of acceptance
      - "Easy" transition back from new state increases chance of acceptance
- **"Move" step is a Markov process with unique stationary distribution equal to distribution we want to estimate**
- **Computationally more challenging than standard PF**

Some Illustrative Results





- Tested algorithms on one-dimensional problem from literature
- Shrinkage and resample-move improve estimate
- Need to explore relative improvement against computation cost of:
  - More particles
  - Shrinkage step
  - MH step
- Need to extend to multi-dimensional PDBNs

Some General Points

- Online inference in state space models with static parameters is *hard*
- Find parts of computation where sampling can be replaced by cheap exact computation
- Find good heuristic approximations to use as proposal distribution
- Good proposal distribution is typically more important than more samples
- Offline estimation of static parameters may be necessary

Summary and Synthesis

- **Dynamic Bayesian networks pose a challenge for BN methods**
  - Even sparsely connected DBNs give rise to intractable junction trees
  - Approximation is necessary for problems of any size
- **Standard tasks for DBN inference**
  - Filtering
  - Prediction
  - Smoothing
  - Estimation
- **Approximation approaches**
  - Project current belief to lower-dimensional approximation that can be rolled forward in time tractably (Boyen-Koller and factored frontier)
  - Monte Carlo simulation (particle filter)
  - Hybrid approaches
- **We discussed several ways to improve approximate methods**

References

General Tracking and Fusion

- Bar-Shalom, Y. and X. Li. *Estimation and Tracking: Principles, Techniques, and Software*. Storrs, CT: YBS, 1995.
- Stone, L.D., Barlow, C.A. and Corwin, T.L. *Bayesian Multiple Target Tracking*, Boston, MA: Artech House, 1999.

Bayesian networks for Tracking and Fusion

- Krieg, M.L. (2003) "Joint Multi-sensor Kinematic and Attribute Tracking using Bayesian Belief Networks," Proc.of Information Fusion' 2003, pp. 17-24.

Rollup

- Takikawa, M., d'Ambrosio, B. and Wright, E. Real-Time Inference with Large-Scale Temporal Bayes Nets. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2002.

Boyen-Koller and related approximations

- Boyen, X. and Koller, D. Tractable inference for complex stochastic processes. In *Proc. of the Conf. on Uncertainty in AI*, 1998. http://citeseer.nj.nec.com/boyen98tractable.html
- Murphy, K. and Weiss, Y. The factored frontier algorithm for approximate inference in DBNsIn *Proc. of the Conf. on Uncertainty in AI*, 2001.

Particle filters and related Monte Carlo methods

- Arulampalam, S., Maskell, S., Gordon, N.J., and Clapp, T. (2002) "A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking," *IEEE Tran on Signal Proc*., 50(2), pp. 174-188.
- Doucet, A. de Freitas, N., Murphy, K. and Russell. S. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 176-- 183, Stanford, 2000. http://citeseer.nj.nec.com/doucet00raoblackwellised.html
- Doucet, A., de Frietas, N., Gordon, N. and Smith, A. (eds) *Sequential Monte Carlo Methods in Practice*, Springer-Verlag, 2001.
- Gilks, W.R. and Berzuini, C. Following a Moving Target - Monte Carlo Inference for Dynamic Bayesian Models *Journal of the Royal Statisical Society B. 63*: 127-146

Some useful URLs

- http://sigwww.cs.tut.fi/TICSP/PubsSampsa/MSc_Thesis.pdf
- http://www.bookpool.com/sm/158053631X