Introduction to Machine Learning

Brown University CSCI 1950-F, Spring 2012
Prof. Erik Sudderth

Lecture 25:
Markov Chain Monte Carlo (MCMC)
Course Review and Advanced Topics

Many figures courtesy Kevin Murphy's textbook,
Machine Learning: A Probabilistic Perspective

Monte Carlo Methods

$$\mathbb{E}[f] = \int f(z)p(z)\, dz \approx \frac{1}{L}\sum_{\ell=1}^{L} f(z^{(\ell)}) \qquad z^{(\ell)} \sim p(z)$$

Estimation of expected model properties via simulation

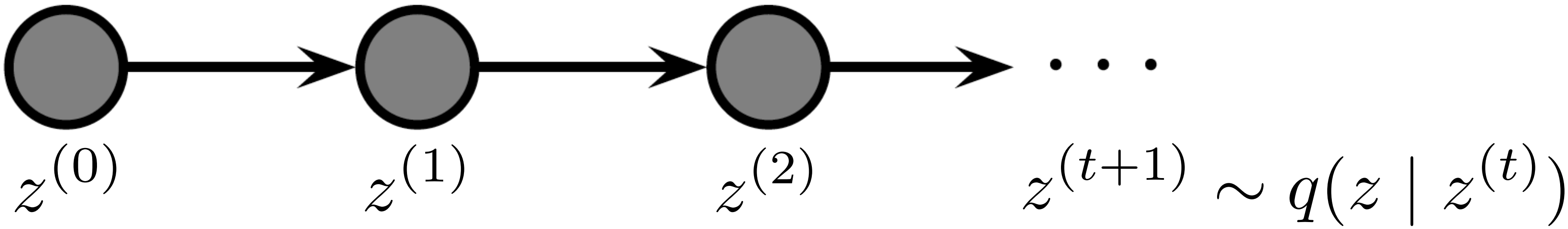
Provably good if *L* sufficiently large:

- Unbiased for any sample size
- Variance inversely proportional to sample size (and independent of dimension of space)
- Weak law of large numbers
- Strong law of large numbers
- **Problem:** Drawing samples from complex distributions…

Alternatives for hard problems:

- Importance sampling
- *Markov chain Monte Carlo (MCMC)*

Markov Chain Monte Carlo

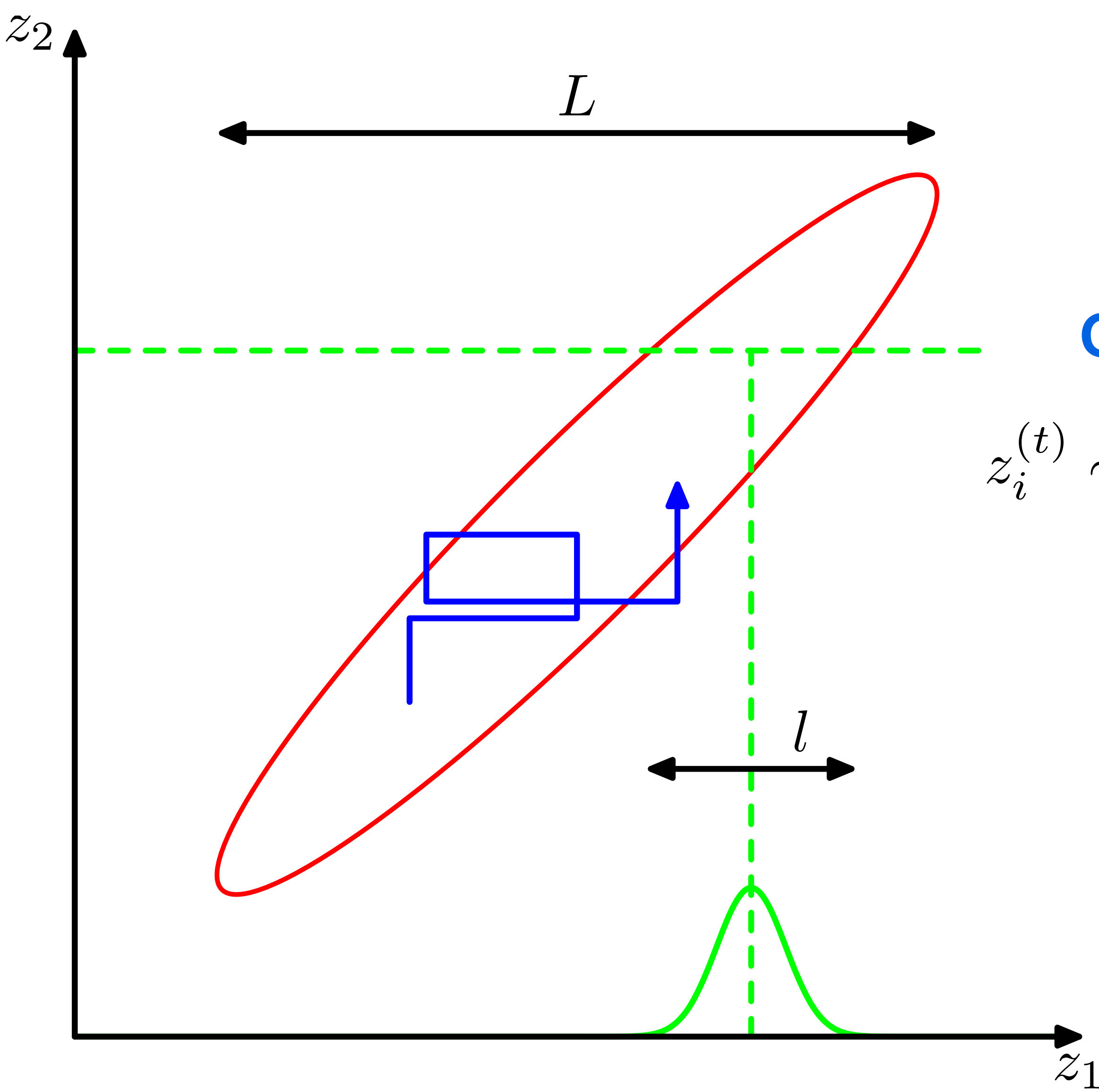


- At each time point, state $z^{(t)}$ is a configuration of *all the variables in the model:* parameters, hidden variables, etc.
- We design the transition distribution $q(z \mid z^{(t)})$ so that the chain is *irreducible* and *ergodic*, with a unique stationary distribution $p^*(z)$

$$p^*(z) = \int_{\mathcal{Z}} q(z \mid z')p^*(z')\, dz'$$

- For learning, the target equilibrium distribution is usually the posterior distribution given data *x*: $p^*(z) = p(z \mid x)$
- Popular recipes: *Metropolis-Hastings and Gibbs samplers*

Gibbs Sampler for a 2D Gaussian



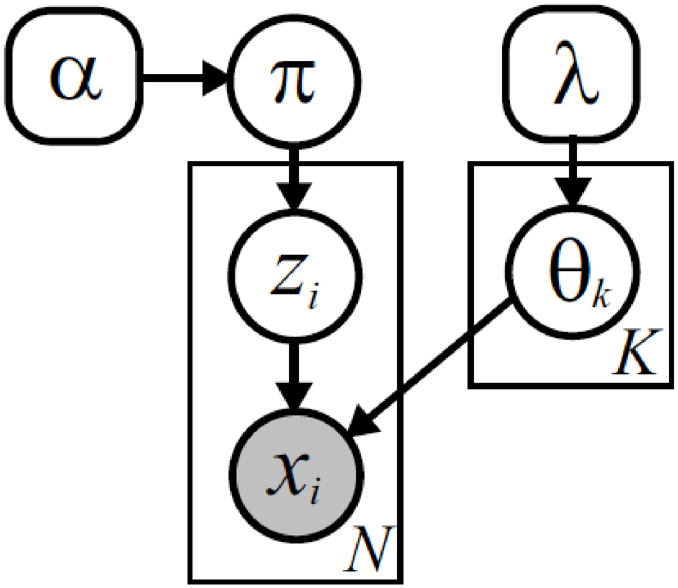
General Gibbs Sampler

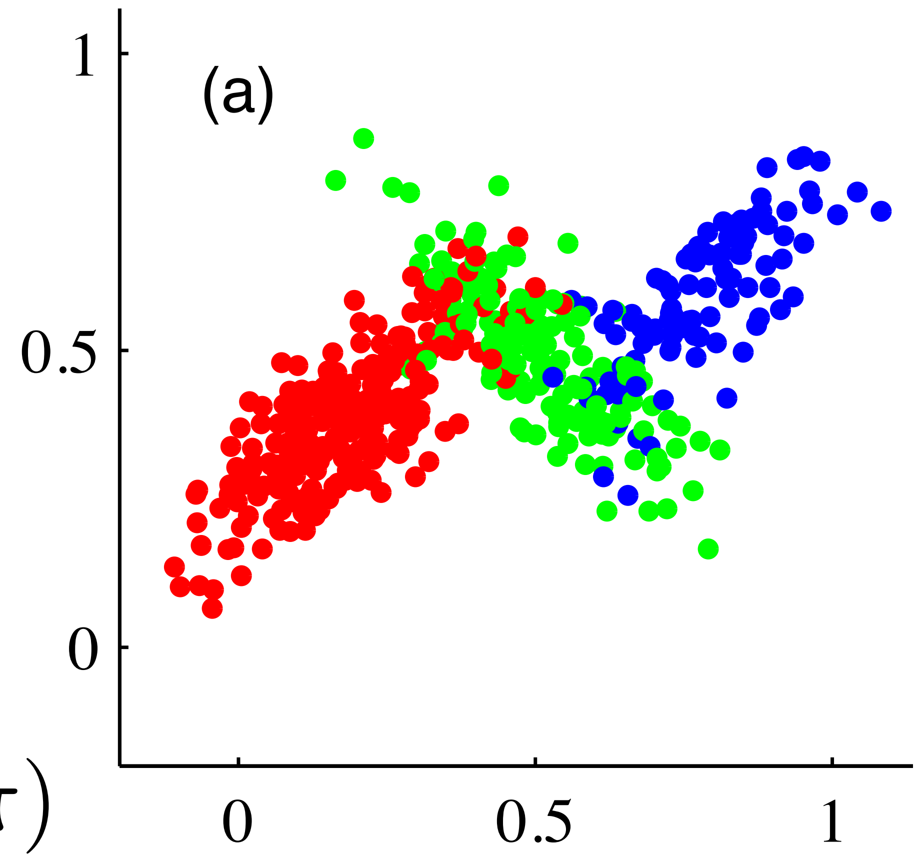
$$z_i^{(t)} \sim p(z_i \mid z_{\setminus i}^{(t-1)}) \qquad i = i(t)$$
$$z_j^{(t)} = z_j^{(t-1)} \qquad j \neq i(t)$$

Under mild conditions, converges assuming all variables are resampled infinitely often (order can be fixed or random)

C. Bishop, Pattern Recognition & Machine Learning

Probabilistic Mixture Models





$$\theta_k = \{\mu_k, \Sigma_k\}$$

$$p(z_i \mid \pi) = \mathrm{Cat}(z_i \mid \pi)$$

$$p(x_i \mid z_i, \mu, \Sigma) = \mathcal{N}(x_i \mid \mu_{z_i}, \Sigma_{z_i})$$

Mixture Sampler Pseudocode

Given mixture weights $\pi^{(t-1)}$ and cluster parameters $\{\theta_k^{(t-1)}\}_{k=1}^K$ from the previous iteration, sample a new set of mixture parameters as follows:

1. Independently assign each of the N data points x_i to one of the K clusters by sampling the indicator variables $z = \{z_i\}_{i=1}^N$ from the following multinomial distributions:

$$z_i^{(t)} \sim \frac{1}{Z_i} \sum_{k=1}^{K} \pi_k^{(t-1)} f(x_i \mid \theta_k^{(t-1)})\, \delta(z_i, k) \qquad Z_i = \sum_{k=1}^{K} \pi_k^{(t-1)} f(x_i \mid \theta_k^{(t-1)})$$

2. Sample new mixture weights according to the following Dirichlet distribution:

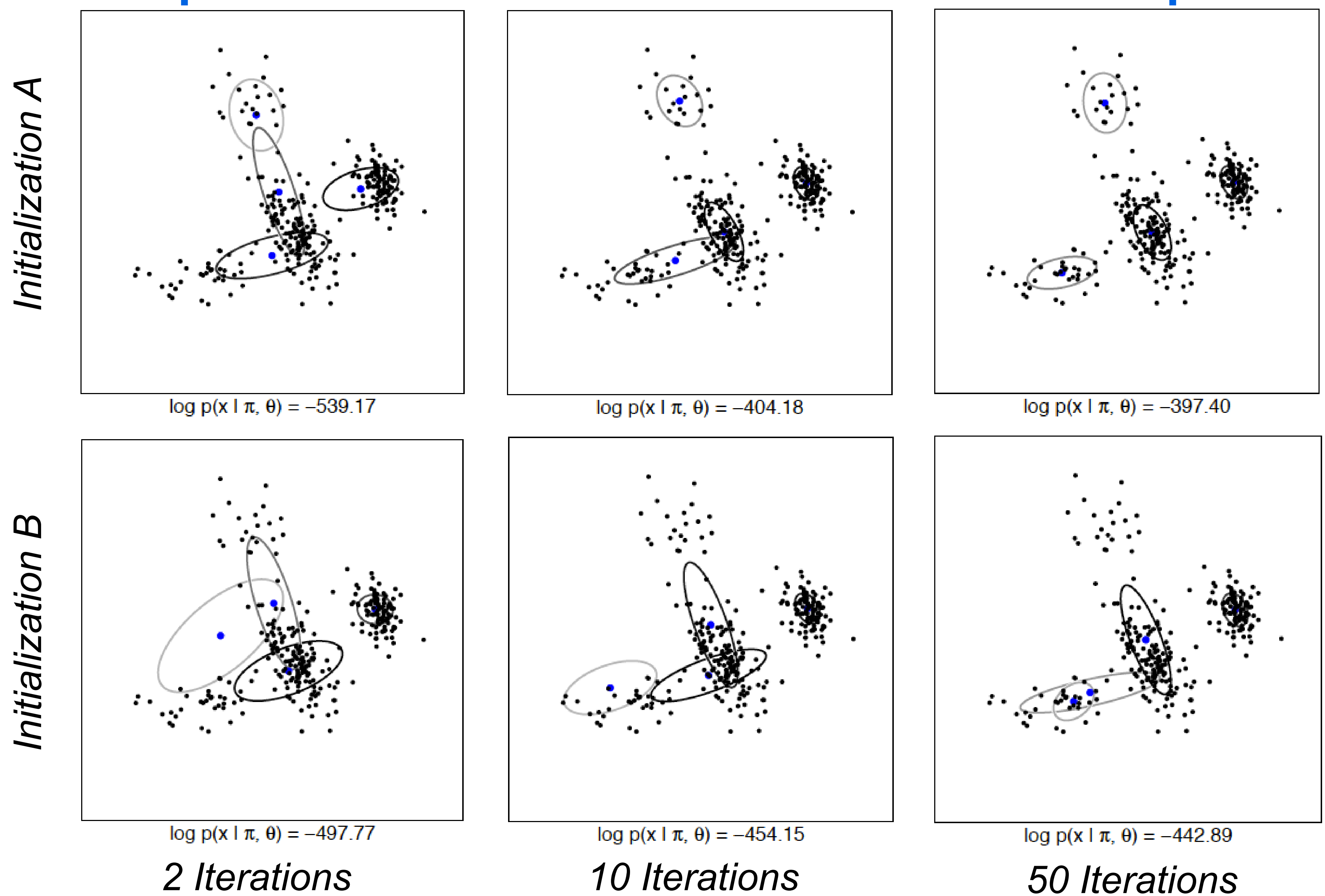
$$\pi^{(t)} \sim \mathrm{Dir}(N_1 + \alpha/K, \ldots, N_K + \alpha/K) \qquad N_k = \sum_{i=1}^{N} \delta(z_i^{(t)}, k)$$

3. For each of the K clusters, independently sample new parameters from the conditional distribution implied by those observations currently assigned to that cluster:

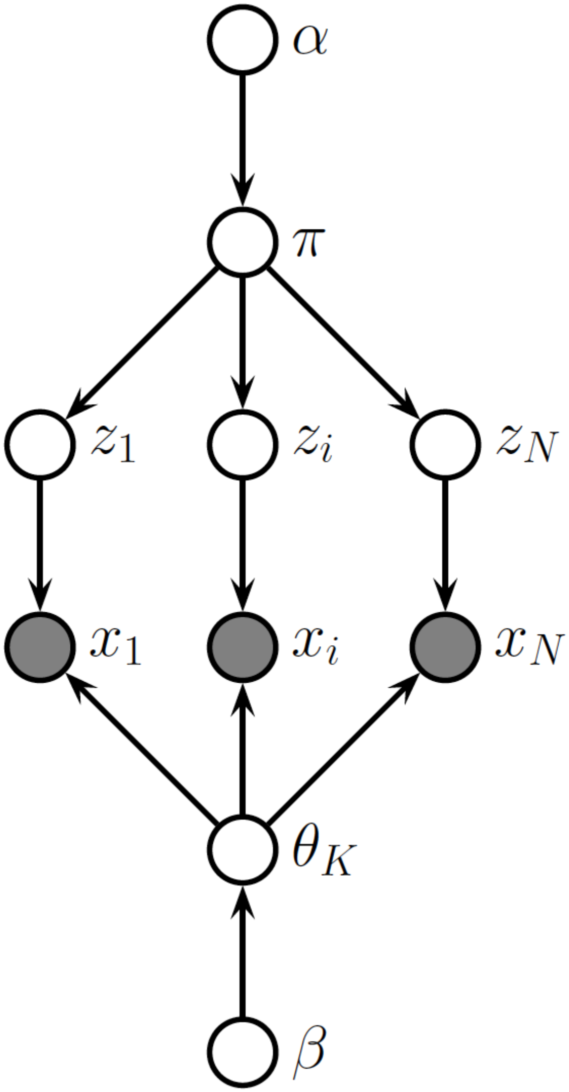
$$\theta_k^{(t)} \sim p(\theta_k \mid \{x_i \mid z_i^{(t)} = k\}, \lambda)$$

When λ defines a conjugate prior, this posterior distribution is given by Prop. 2.1.4.

Snapshots of Mixture Gibbs Sampler



Collapsed Sampling Algorithms

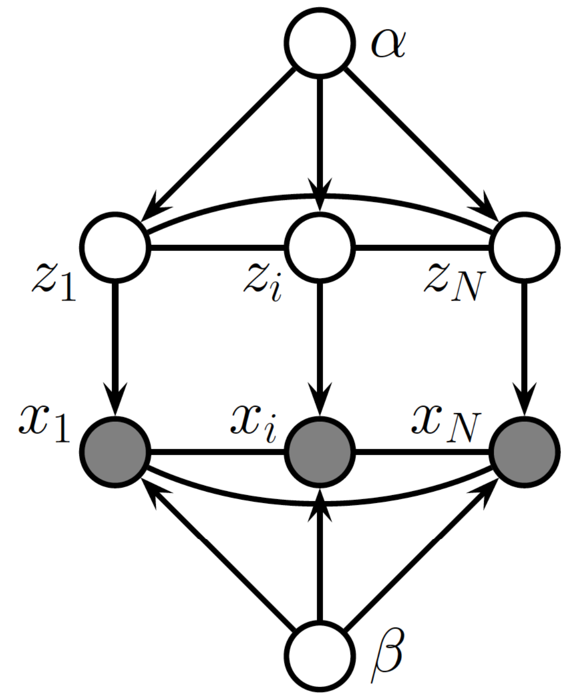


$$\pi \sim \mathrm{Dir}(\alpha)$$

$$z_i \sim \mathrm{Cat}(\pi)$$

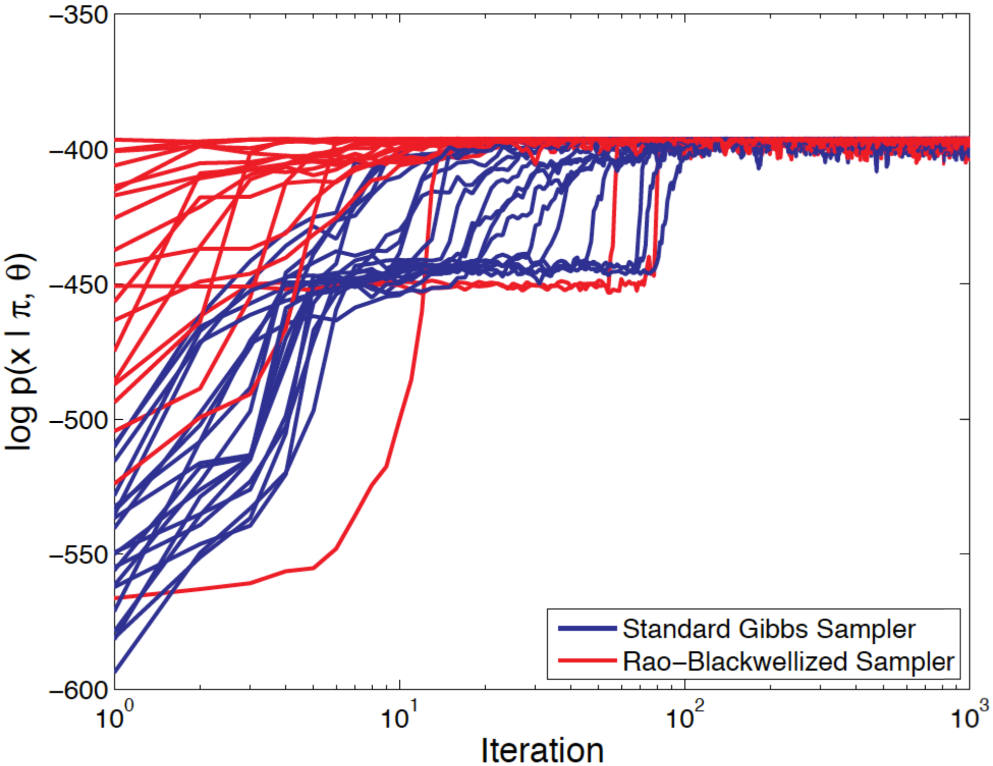
$$x_i \sim F(\theta_{z_i})$$

$$\theta_k \sim G(\beta)$$

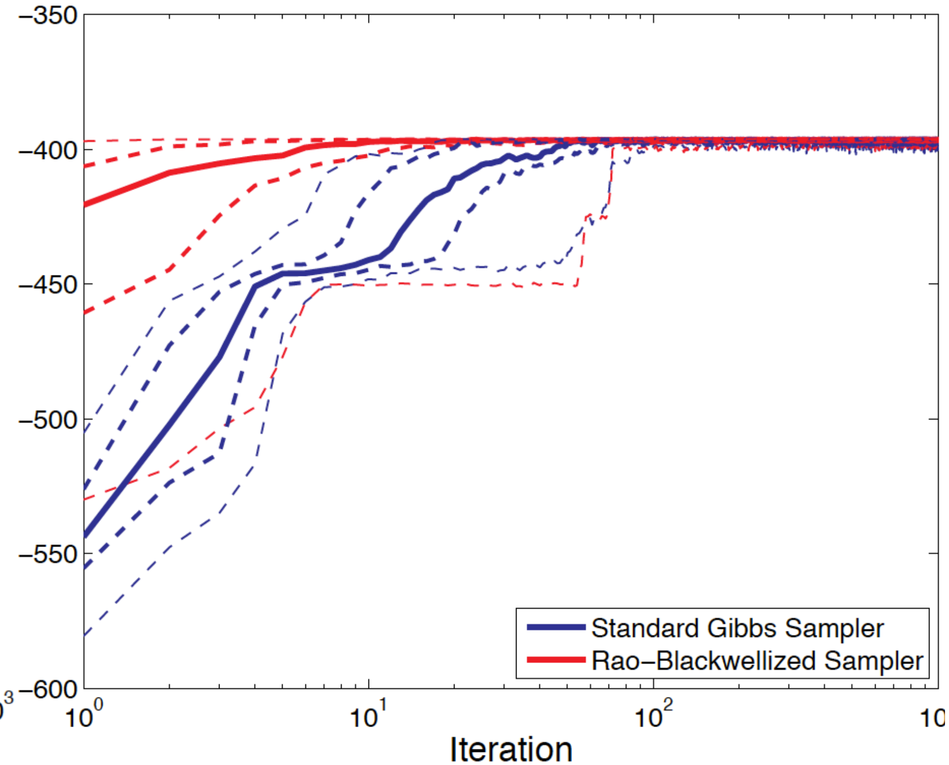


Conjugate priors allow exact marginalization of parameters, to make an equivalent model with fewer variables

Gibbs: Representation and Mixing



Multiple Initializations

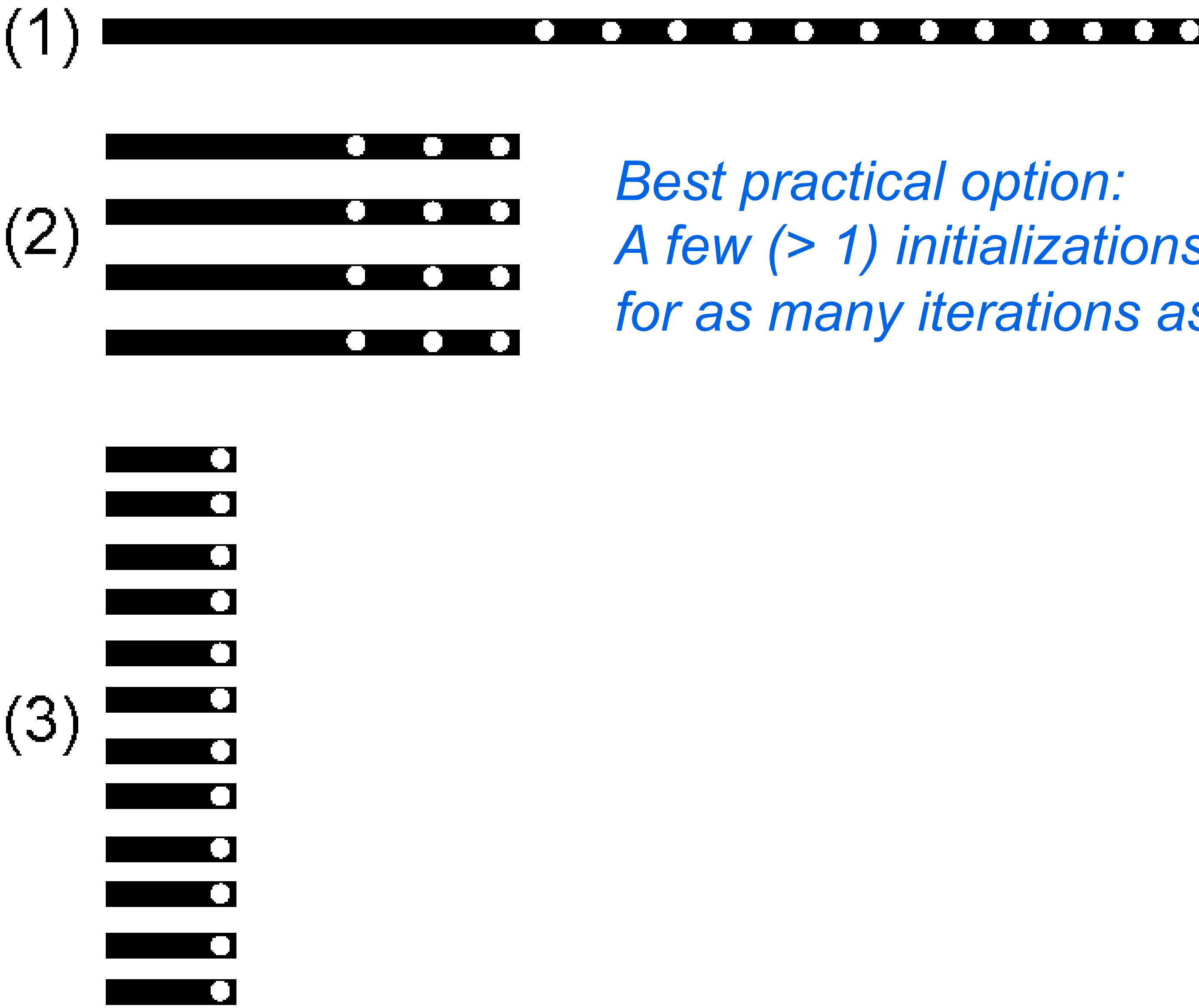


Quantiles of 100 Chains

Standard Gibbs: Alternatively sample assignments, parameters

Collapsed Gibbs: Marginalize parameters, sample assignments

MCMC & Computational Resources



Best practical option:
A few (> 1) initializations
for as many iterations as possible

End of New Material

Next Slides: Some review and some advertisement of advanced topics

The Main Learning Problems

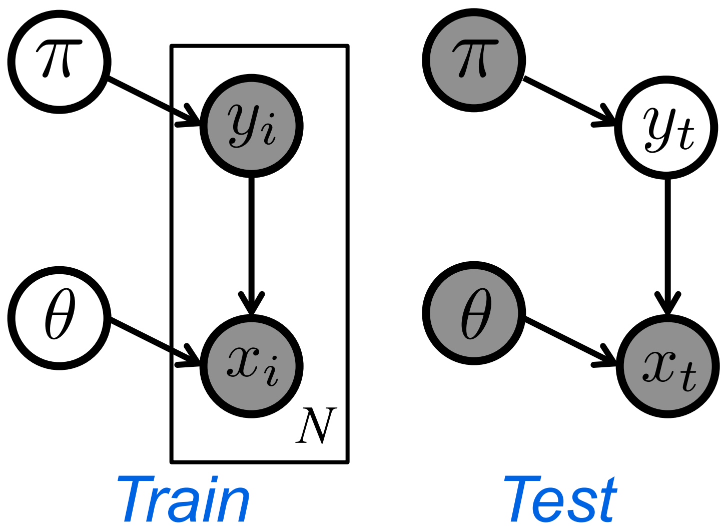
	Supervised Learning	***Unsupervised Learning***
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

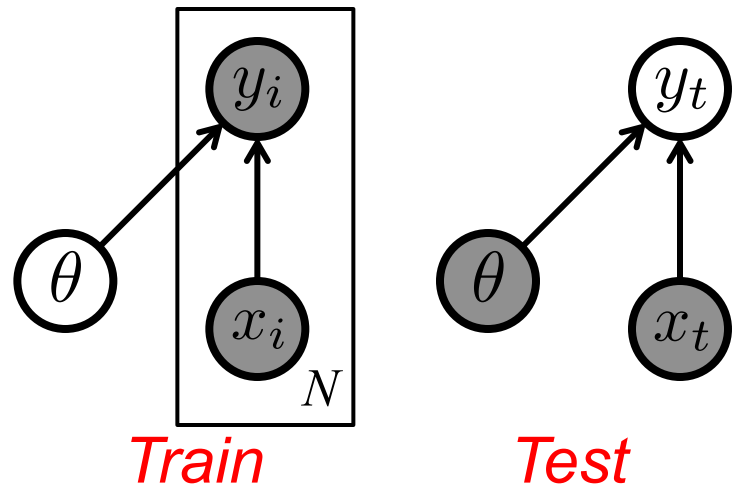
- **Supervised:** Learn to approximate a function from examples
- **Unsupervised:** Learn a representation which compresses data
- **Probabilistic learning:** Learn by maximizing probability, or minimizing an expected loss

Supervised Learning

Generative ML or MAP Learning: *Naïve Bayes*

$$\max_{\pi,\theta} \log p(\pi) + \log p(\theta) + \sum_{i=1}^{N} [\log p(y_i \mid \pi) + \log p(x_i \mid y_i, \theta)]$$





Discriminative ML or MAP Learning: *Logistic regression*

$$\max_{\theta} \log p(\theta) + \sum_{i=1}^{N} \log p(y_i \mid x_i, \theta)$$

Learning via Optimization

ML Estimate: $$\hat{w} = \arg\min_w \; -\sum_i \log p(y_i \mid x_i, w)$$

MAP Estimate: $$\hat{w} = \arg\min_w \; -\log p(w) - \sum_i \log p(y_i \mid x_i, w)$$

Gradient vectors:

$$f : \mathbb{R}^M \rightarrow \mathbb{R}$$
$$\nabla_w f : \mathbb{R}^M \rightarrow \mathbb{R}^M$$
$$(\nabla_w f(w))_k = \frac{\partial f(w)}{\partial w_k}$$

Hessian matrices:

$$\nabla_w^2 f : \mathbb{R}^M \rightarrow \mathbb{R}^{M \times M}$$
$$(\nabla_w f(w))_{k,\ell} = \frac{\partial^2 f(w)}{\partial w_k \partial w_\ell}$$

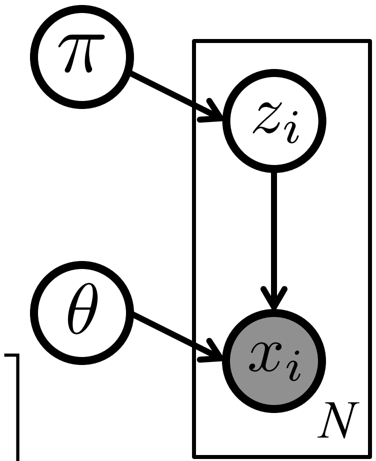
Optimization of Smooth Functions:

- *Closed form:* Find zero gradient points, check curvature
- *Iterative:* Initialize somewhere, use gradients to take steps towards better (by convention, smaller) values

Unsupervised Learning

Clustering:

$$\max_{\pi,\theta} \; \log p(\pi) + \log p(\theta) + \sum_{i=1}^{N} \log \left[\sum_{z_i} p(z_i \mid \pi) p(x_i \mid z_i, \theta) \right]$$

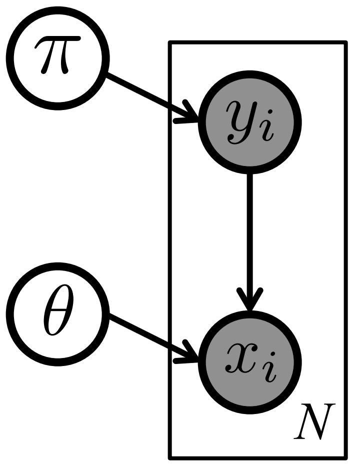


Dimensionality Reduction:

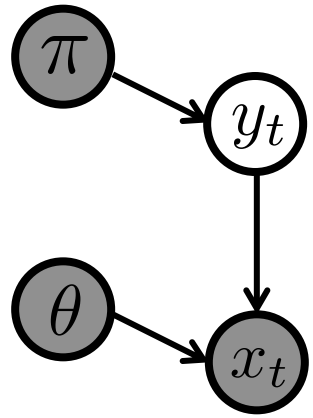
$$\max_{\pi,\theta} \; \log p(\pi) + \log p(\theta) + \sum_{i=1}^{N} \log \left[\int_{z_i} p(z_i \mid \pi) p(x_i \mid z_i, \theta) \; dz_i \right]$$

- No notion of training and test data: labels are *never* observed
- As before, *maximize* posterior probability of model parameters
- For hidden variables associated with each observation, we *marginalize* over possible values rather than estimating
  - Fully accounts for uncertainty in these variables
  - There is one hidden variable per observation, so cannot perfectly estimate even with infinite data
- Must use generative model (discriminative degenerates)

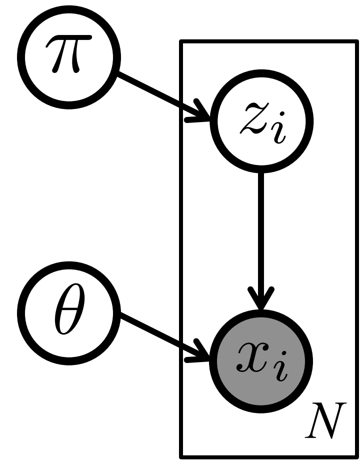
Expectation Maximization (EM)



Supervised Training



Supervised Testing



Unsupervised Learning

$\pi, \theta \longrightarrow$ *parameters (define low-dimensional manifold)*

$z_1, \ldots, z_N \longrightarrow$ *hidden data (locate observations on manifold)*

- **Initialization:** Randomly select starting parameters
- **E-Step:** Given parameters, find posterior of hidden data
  - Equivalent to test inference of full posterior distribution
- **M-Step:** Given posterior distributions, find likely parameters
  - Similar to supervised ML/MAP training
- **Iteration:** Alternate E-step & M-step until convergence

EM as Lower Bound Maximization

$$\ln p(x \mid \theta) = \ln \left(\int_z p(x, z \mid \theta) \, dz \right)$$

$$\ln p(x \mid \theta) \geq \int_z q(z) \ln p(x, z \mid \theta) \, dz - \int_z q(z) \ln q(z) \, dz \triangleq \mathcal{L}(q, \theta)$$

- **Initialization:** Randomly select starting parameters $\theta^{(0)}$
- **E-Step:** Given parameters, find posterior of hidden data

$$q^{(t)} = \arg\max_q \mathcal{L}(q, \theta^{(t-1)})$$

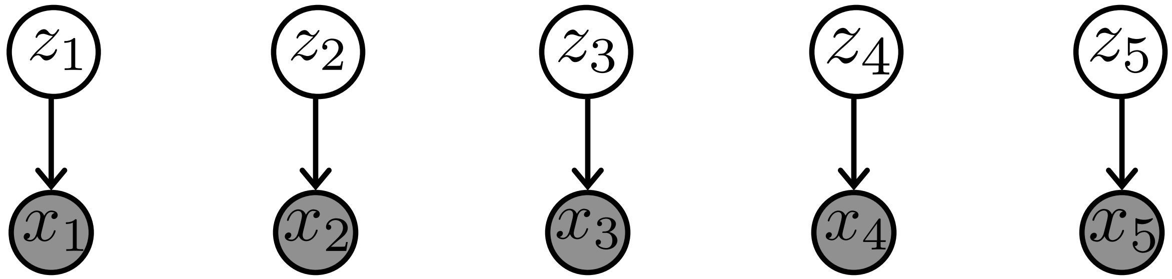
- **M-Step:** Given posterior distributions, find likely parameters

$$\theta^{(t)} = \arg\max_\theta \mathcal{L}(q^{(t)}, \theta)$$

- **Iteration:** Alternate E-step & M-step until convergence

Gaussian Mixture Models vs. HMMs

Mixture Model

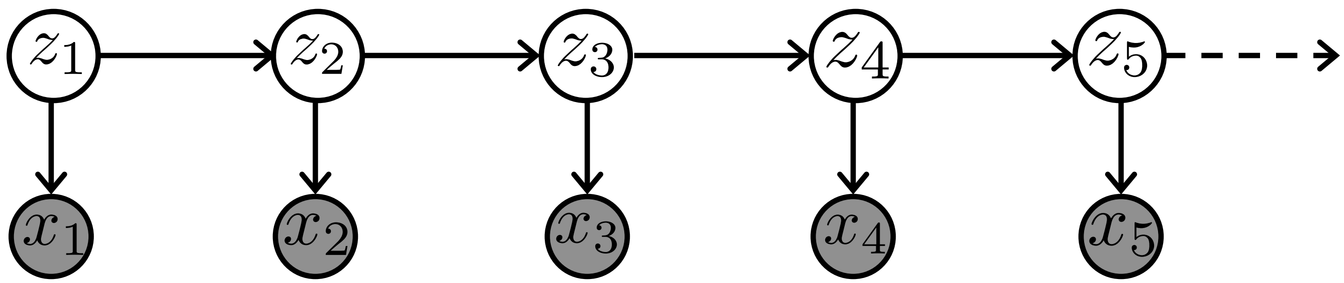


$z_i \in \{1, \ldots, K\}$

$$p(z_i \mid \pi, \mu, \Sigma) = \mathrm{Cat}(z_i \mid \pi)$$

$$p(x_i \mid z_i, \pi, \mu, \Sigma) = \mathrm{Norm}(x_i \mid \mu_{z_i}, \Sigma_{z_i})$$

Hidden Markov Model



$$p(z_t \mid \pi, \mu, \Sigma, z_{t-1}, z_{t-2}, \ldots) = \mathrm{Cat}(z_t \mid \pi_{z_{t-1}})$$

$$p(x_t \mid z_t, \pi, \mu, \Sigma) = \mathrm{Norm}(x_t \mid \mu_{z_t}, \Sigma_{z_t})$$

Recover mixture model when all rows of state transition matrix are equal.

Probabilistic PCA & Factor Analysis

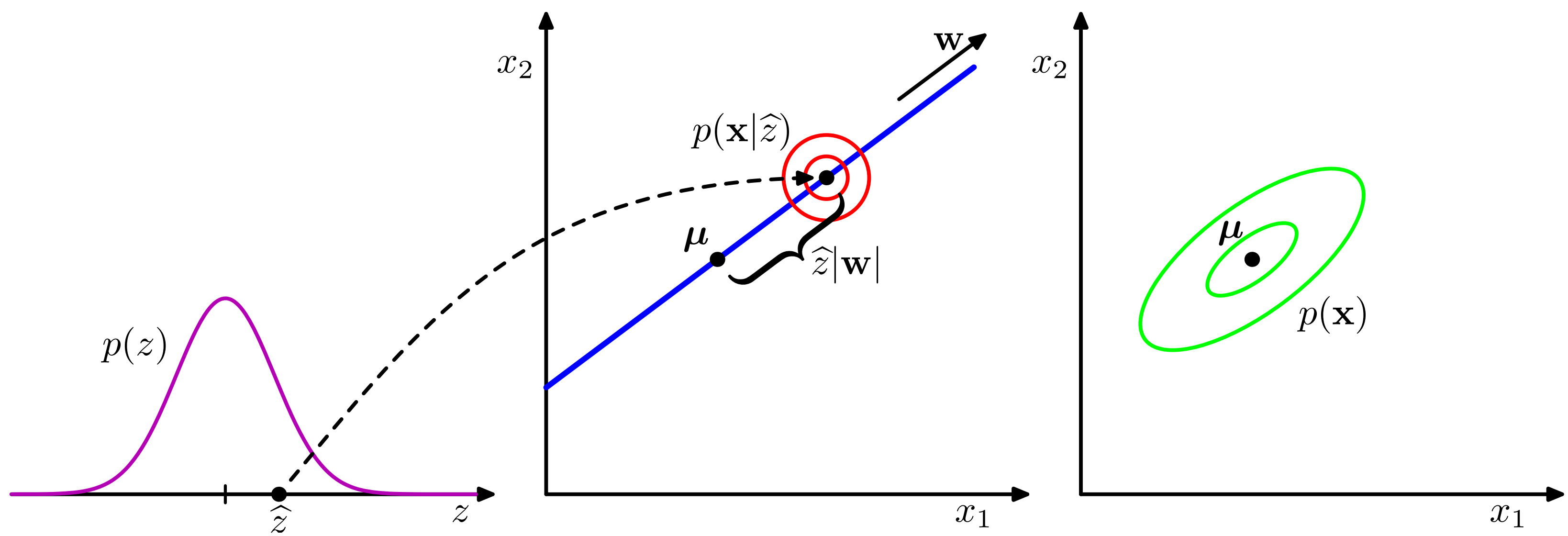
- **Both Models:** Data is a linear function of low-dimensional latent coordinates, plus Gaussian noise

$$p(x_i \mid z_i, \theta) = \mathcal{N}(x_i \mid W z_i + \mu, \Psi) \qquad p(z_i \mid \theta) = \mathcal{N}(z_i \mid 0, I)$$

$$p(x_i \mid \theta) = \mathcal{N}(x_i \mid \mu, W W^T + \Psi)$$

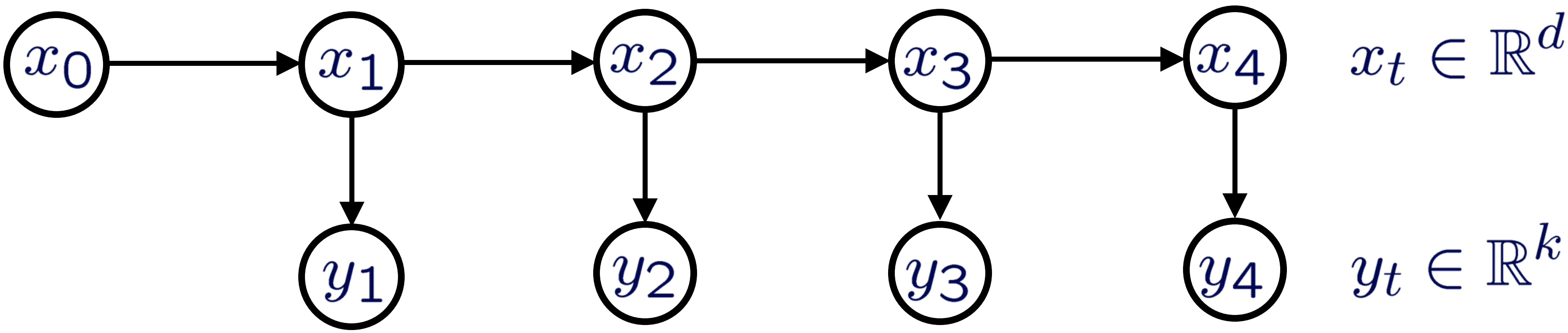
low rank covariance parameterization

- **Factor analysis:** Ψ is a general diagonal matrix
- **Probabilistic PCA:** $\Psi = \sigma^2 I$ is a multiple of identity matrix



C. Bishop, Pattern Recognition & Machine Learning

Linear State Space Models



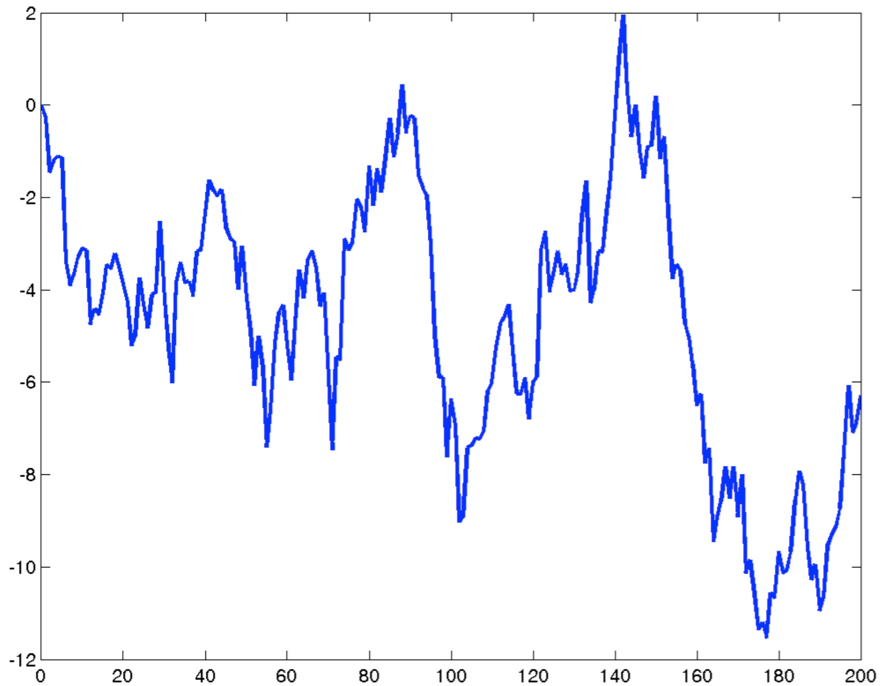
$$x_{t+1} = Ax_t + w_t \qquad w_t \sim \mathcal{N}(0, Q)$$

$$y_t = Cx_t + v_t \qquad v_t \sim \mathcal{N}(0, R)$$

- States & observations jointly Gaussian:
  - All marginals & conditionals Gaussian
  - Linear transformations remain Gaussian

Simple Linear Dynamics

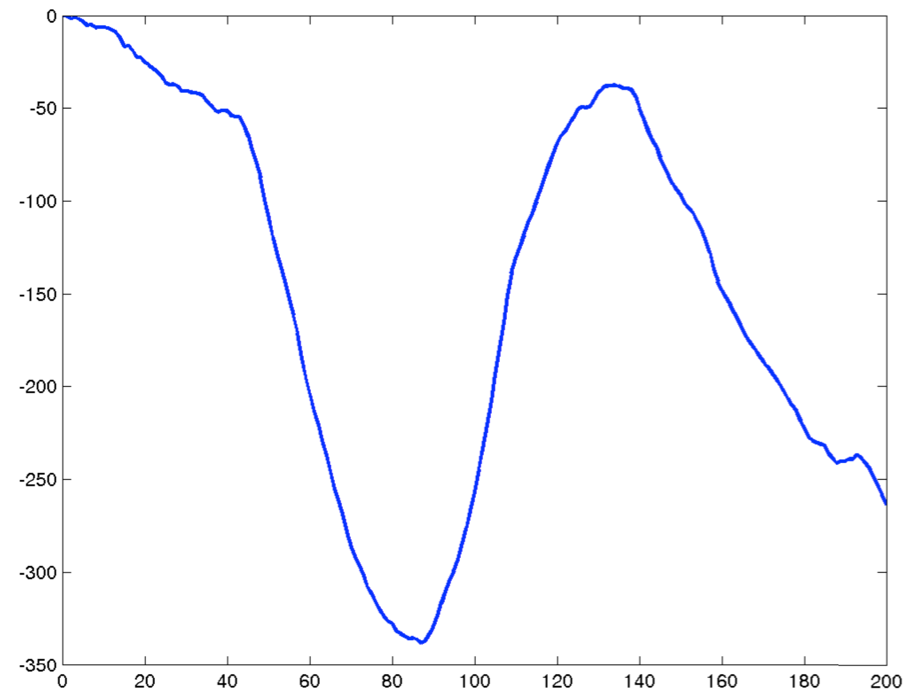
Brownian Motion



Time

$$x_{t+1} = x_t + w_t$$

Constant Velocity



Time

$$\begin{bmatrix} x_{t+1} \\ \delta_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ \delta_t \end{bmatrix} + w_t$$

Kalman Filter

$$x_{t+1} = Ax_t + w_t \qquad w_t \sim \mathcal{N}(0, Q)$$

$$y_t = Cx_t + v_t \qquad v_t \sim \mathcal{N}(0, R)$$

- Represent Gaussians by mean & covariance:

$$p(x_t \mid y_1, \ldots, y_{t-1}) = \mathcal{N}(x; \tilde{\mu}_t, \tilde{\Lambda}_t)$$

$$p(x_t \mid y_1, \ldots, y_t) = \mathcal{N}(x; \mu_t, \Lambda_t)$$

Prediction:

$$\boxed{\begin{aligned} \tilde{\mu}_t &= A\mu_{t-1} \\ \tilde{\Lambda}_t &= A\Lambda_{t-1}A^T + Q \end{aligned}}$$

Kalman Gain:

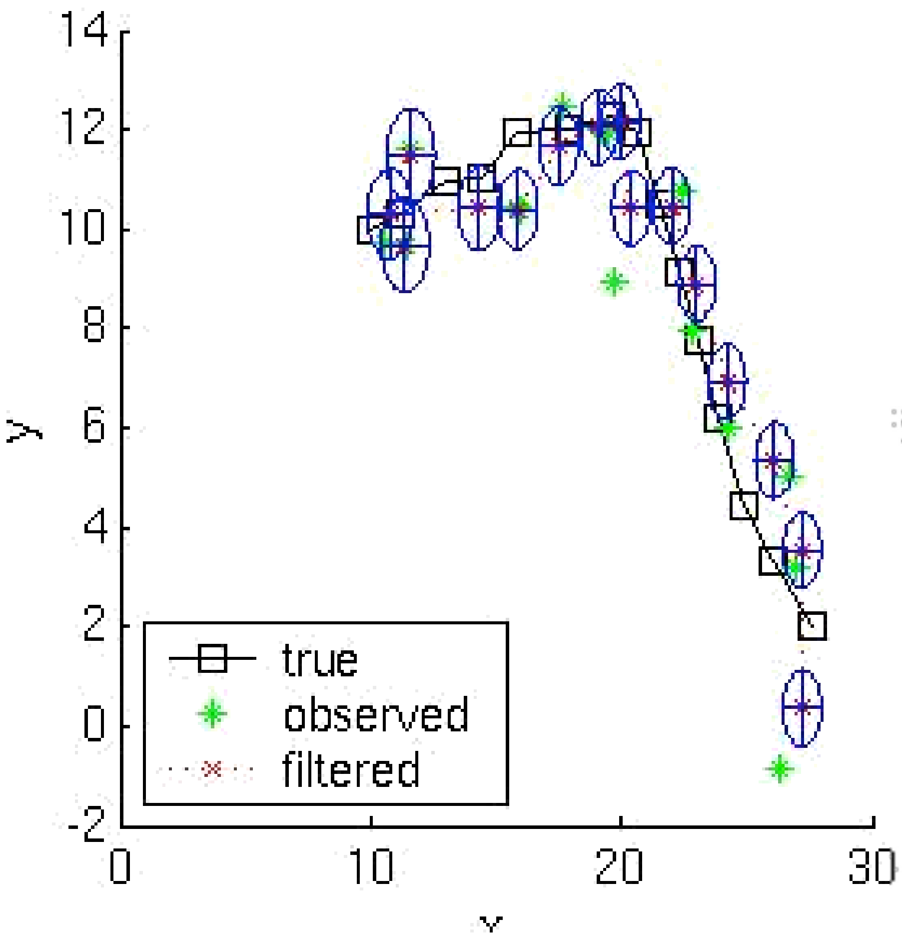
$$K_t = \tilde{\Lambda}_t C^T (C\tilde{\Lambda}_t C^T + R)^{-1}$$

Update:

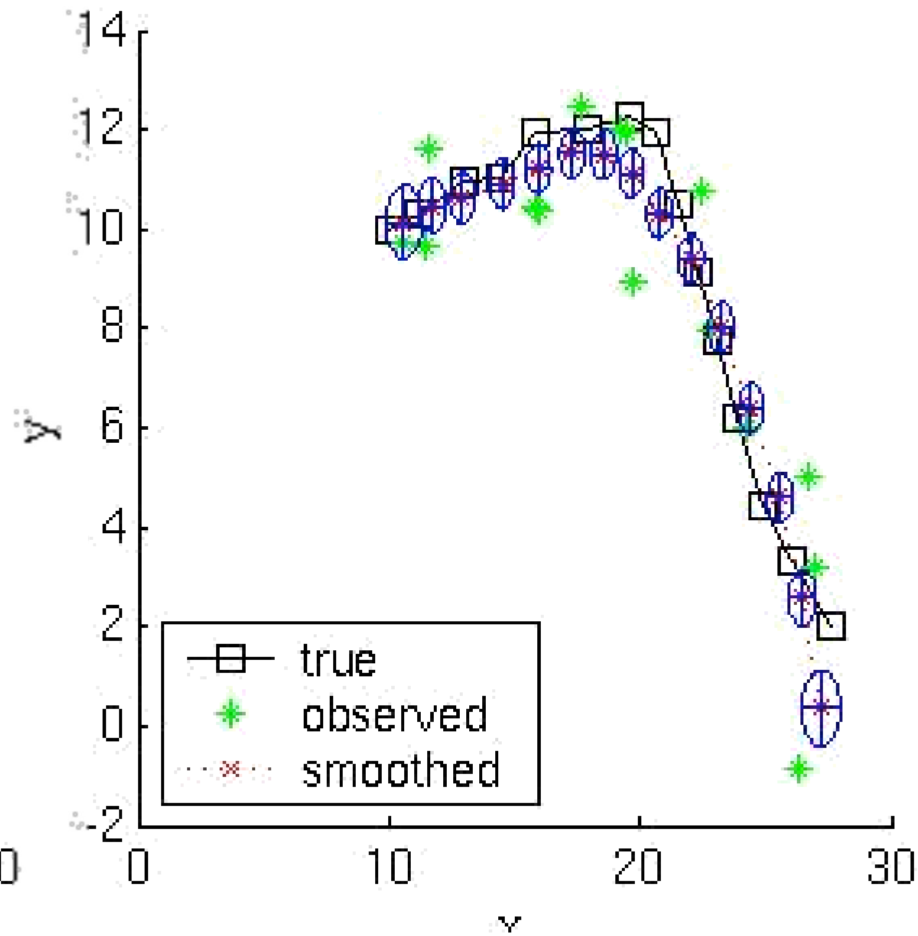
$$\boxed{\begin{aligned} \mu_t &= \tilde{\mu}_t + K_t(y_t - C\tilde{\mu}_t) \\ \Lambda_t &= \tilde{\Lambda}_t - K_t C\tilde{\Lambda}_t \end{aligned}}$$

Constant Velocity Tracking

Kalman Filter

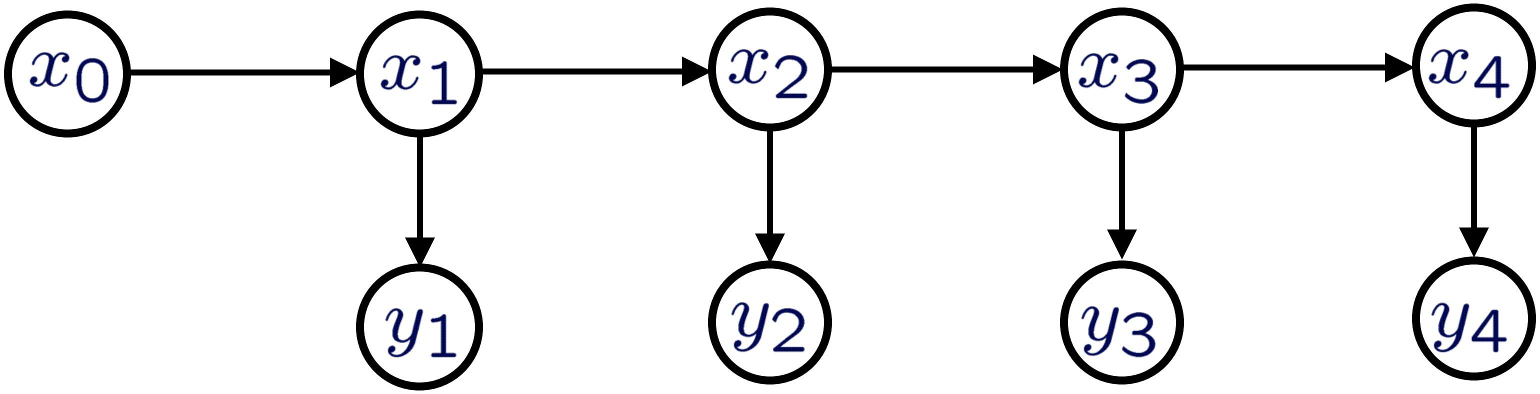


Kalman Smoother



(K. Murphy, 1998)

Nonlinear State Space Models



$x_t \in \mathbb{R}^d$

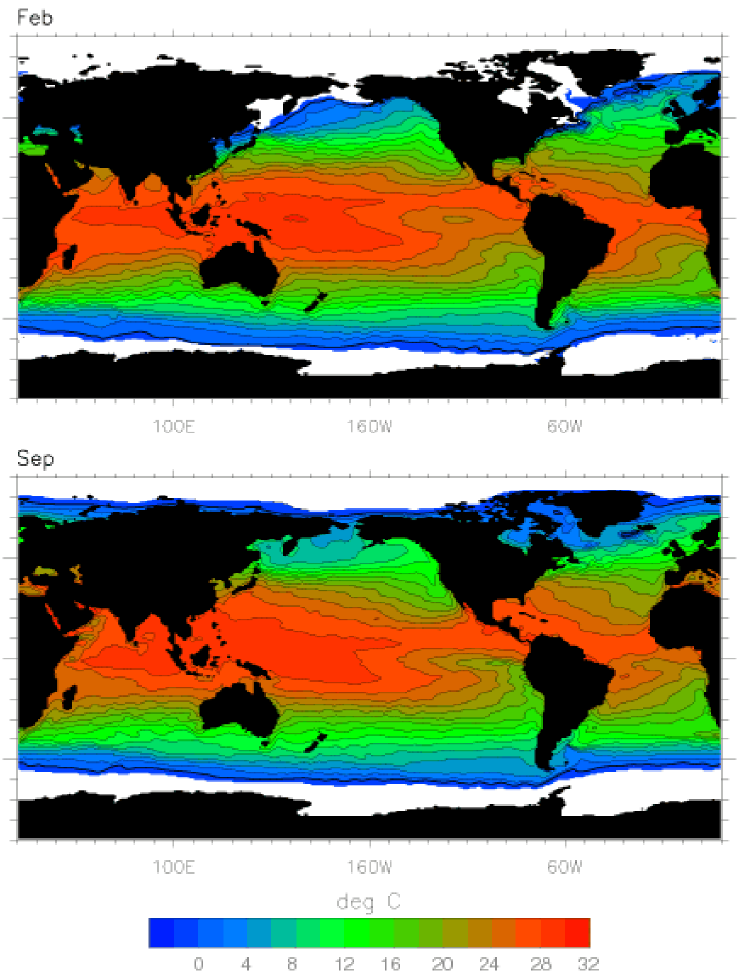
$y_t \in \mathbb{R}^k$

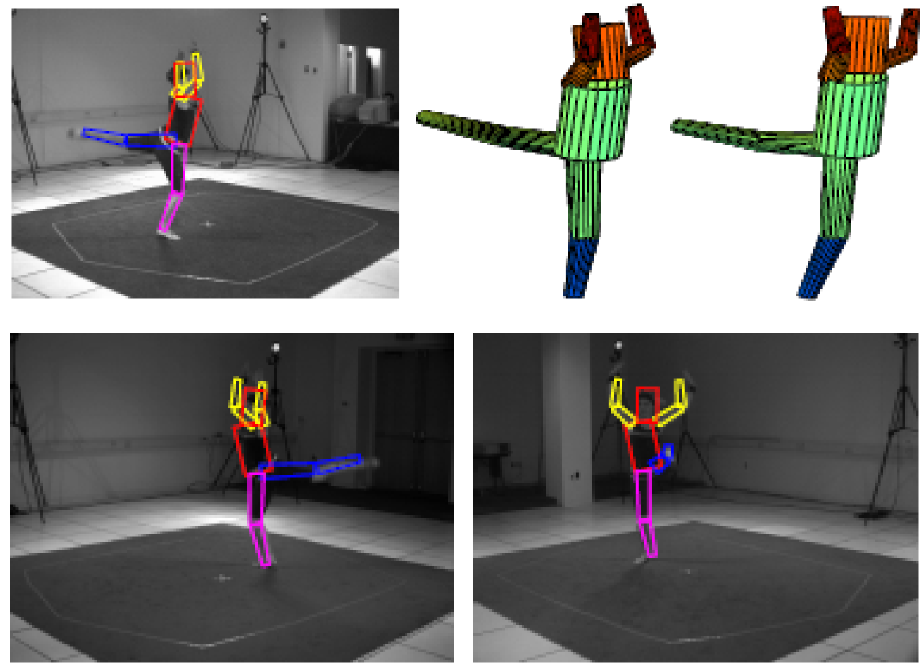
$$x_{t+1} = f(x_t, w_t) \qquad w_t \sim \mathcal{F}$$

$$y_t = g(x_t, v_t) \qquad v_t \sim \mathcal{G}$$

- State dynamics and measurements given by potentially complex nonlinear functions
- Noise sampled from non-Gaussian distributions

Examples of Nonlinear Models

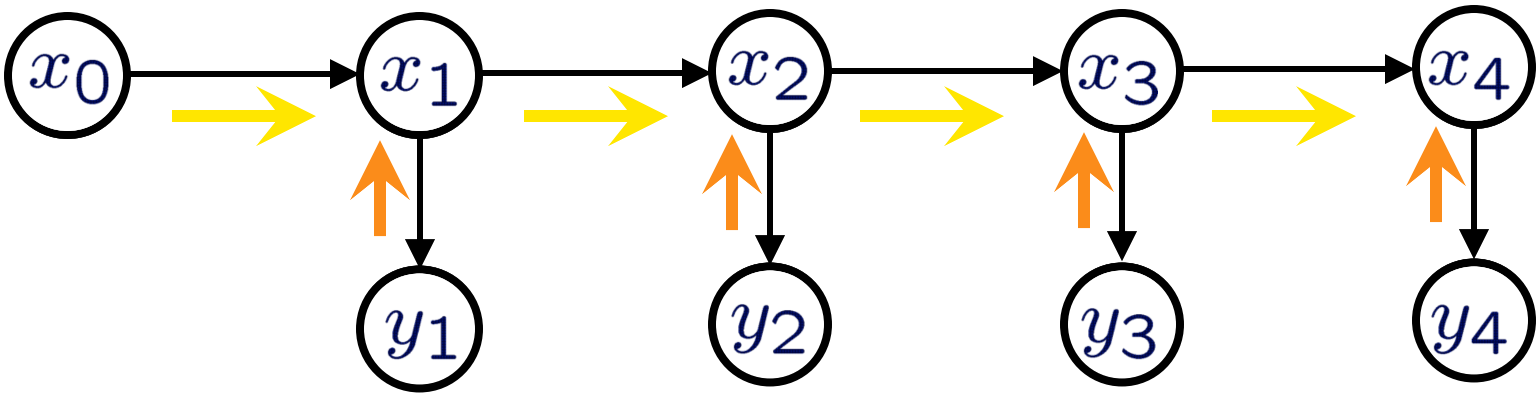




Observed image is a complex function of the 3D pose, other nearby objects & clutter, lighting conditions, camera calibration, etc.

Dynamics implicitly determined by geophysical simulations

Nonlinear Filtering



$$p(x_t \mid y_1, \ldots, y_{t-1}) = \tilde{q}_t(x_t)$$

$$p(x_t \mid y_1, \ldots, y_t) = q_t(x_t)$$

Prediction:

$$\tilde{q}_t(x_t) = \int p(x_t \mid x_{t-1}) q_{t-1}(x_{t-1}) \, dx_{t-1}$$

Update:

$$q_t(x_t) = \frac{1}{Z_t} \tilde{q}_t(x_t) p(y_t \mid x_t)$$

Approximate Nonlinear Filters

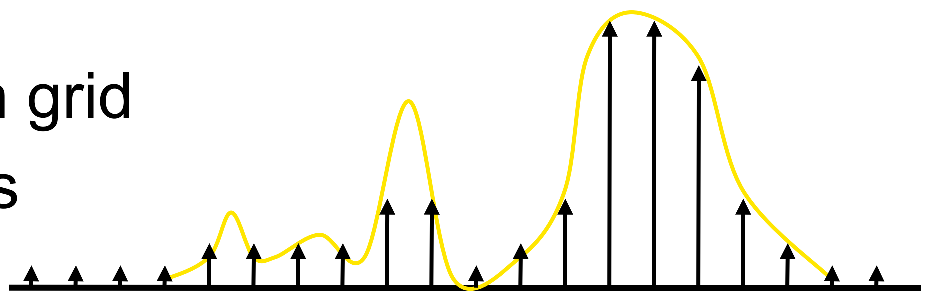
$$q_t(x_t) \propto p(y_t \mid x_t) \cdot \int p(x_t \mid x_{t-1}) q_{t-1}(x_{t-1}) \, dx_{t-1}$$

- No direct representation of continuous functions, or closed form for the prediction integral
- Big literature on approximate filtering:
  - Histogram filters
  - Extended & unscented Kalman filters
  - Particle filters
  - …

Nonlinear Filtering Taxonomy

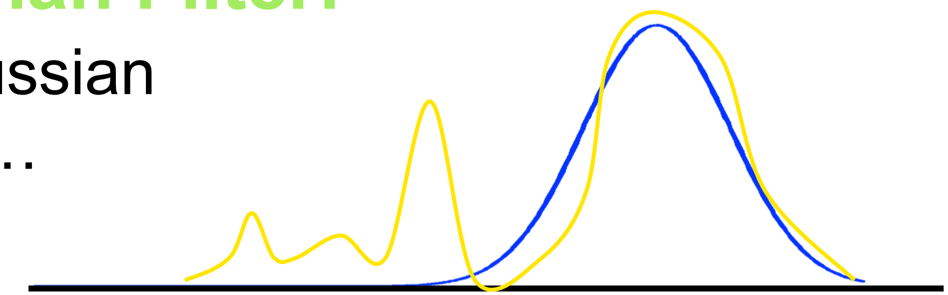
Histogram Filter:

- Evaluate on fixed discretization grid
- Only feasible in low dimensions
- Expensive or inaccurate



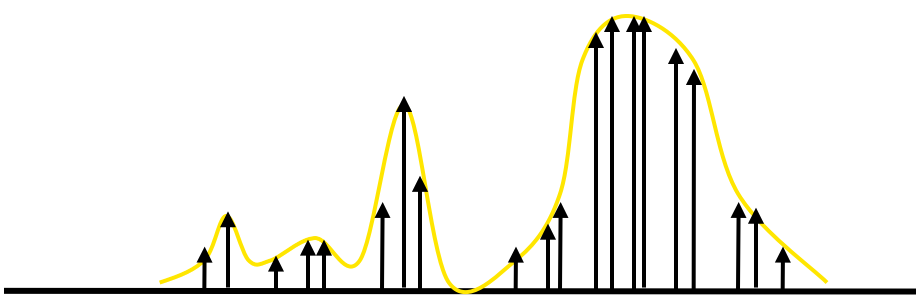
Extended/Unscented Kalman Filter:

- Approximate posterior as Gaussian via linearization, quadrature, …
- Inaccurate for multimodal posterior distributions



Particle Filter:

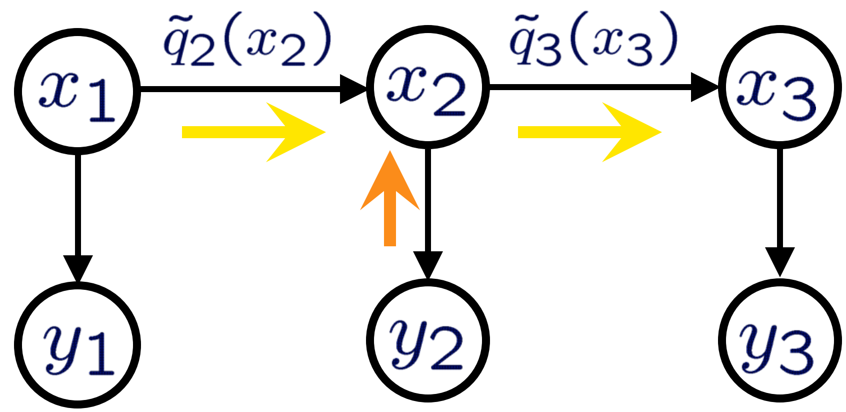
- Dynamically evaluate states with highest probability
- Monte Carlo approximation



Particle Filters

Condensation, Sequential Monte Carlo, Survival of the Fittest,…

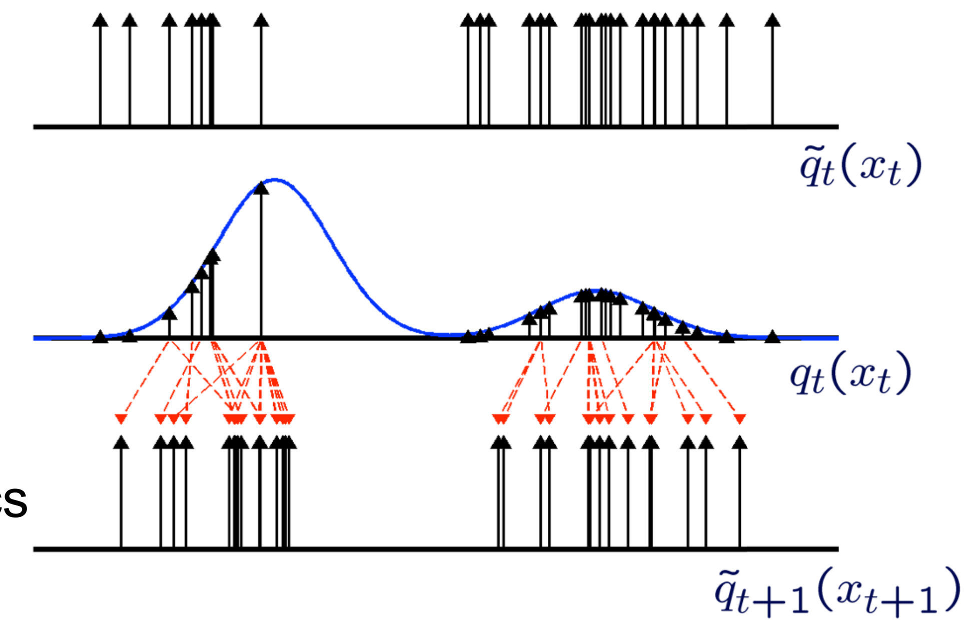
- Represent state estimates using a set of samples
- Propagate over time using importance sampling



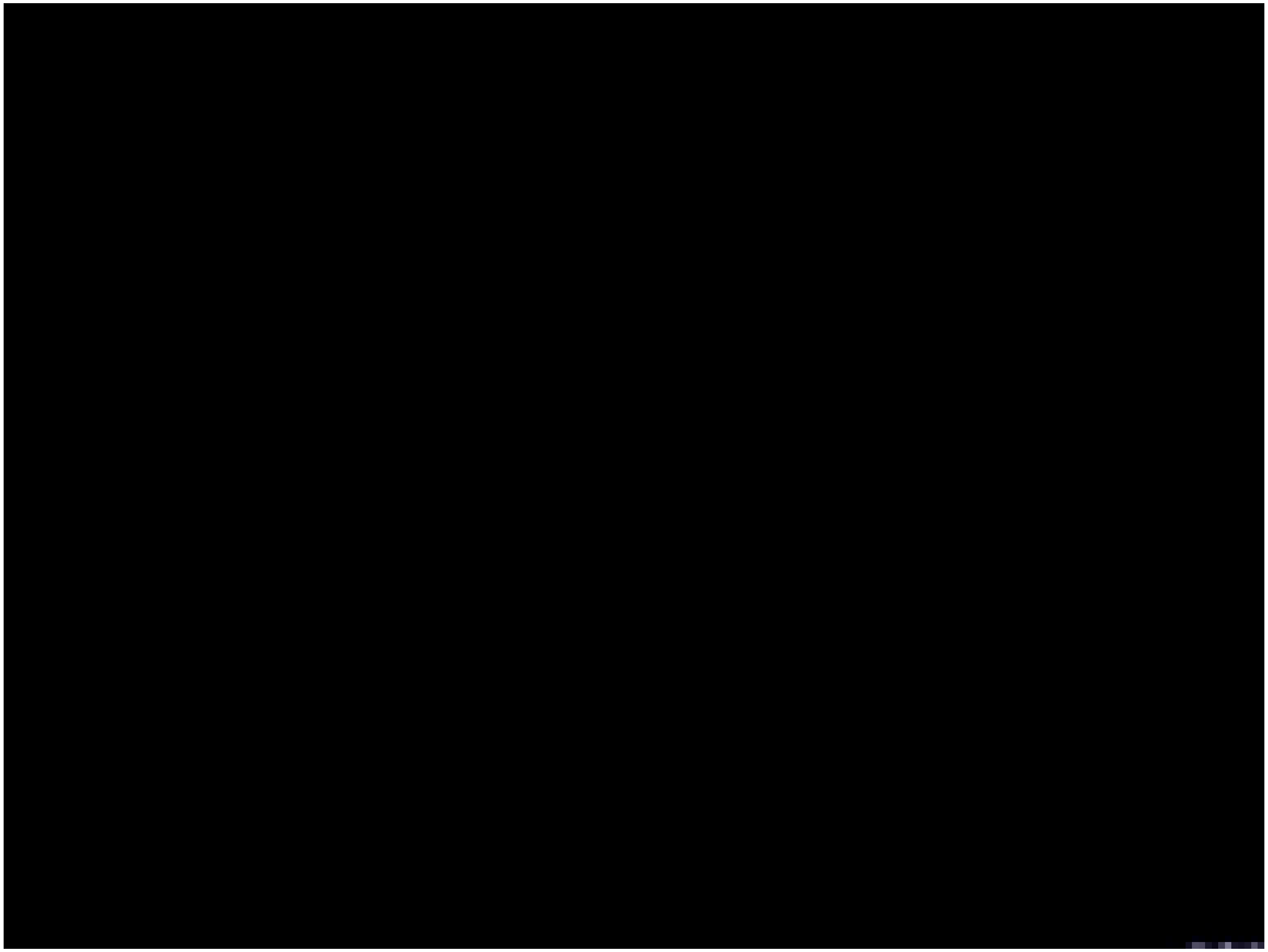
Sample-based density estimate

Weight by observation likelihood

Resample & propagate by dynamics



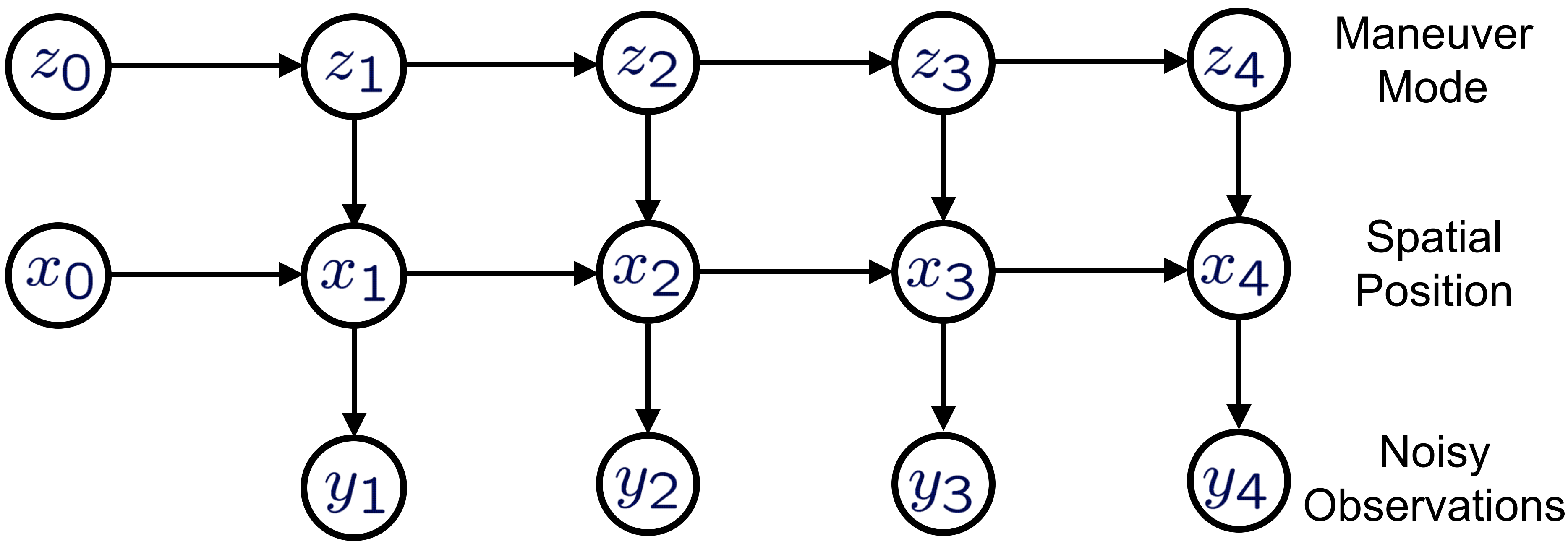
Particle Filtering Movie



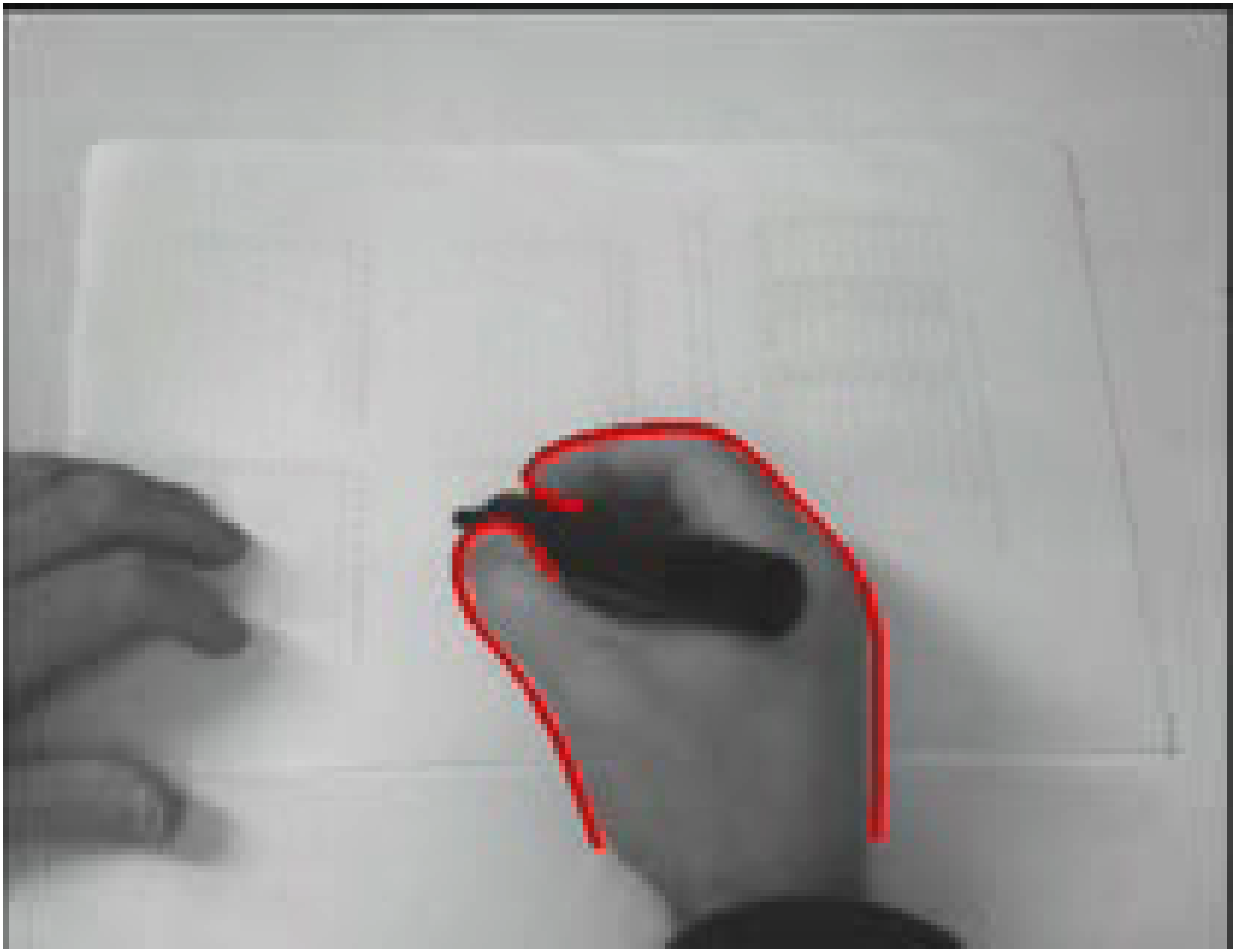
(M. Isard, 1996)

Dynamic Bayesian Networks

Specify and exploit internal structure in the hidden states underlying a time series



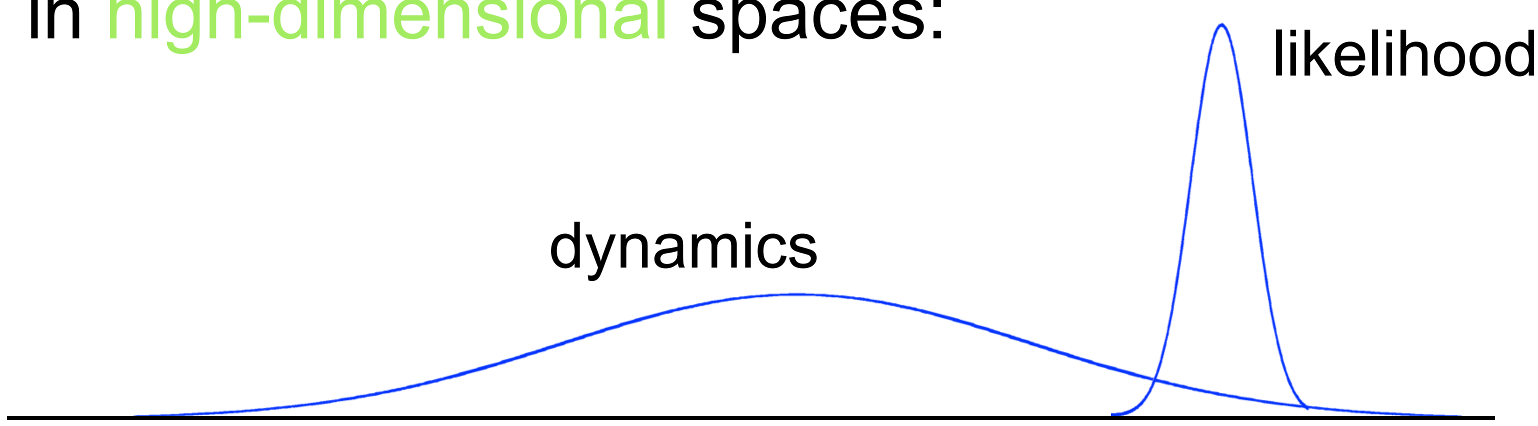
DBN Hand Tracking Video



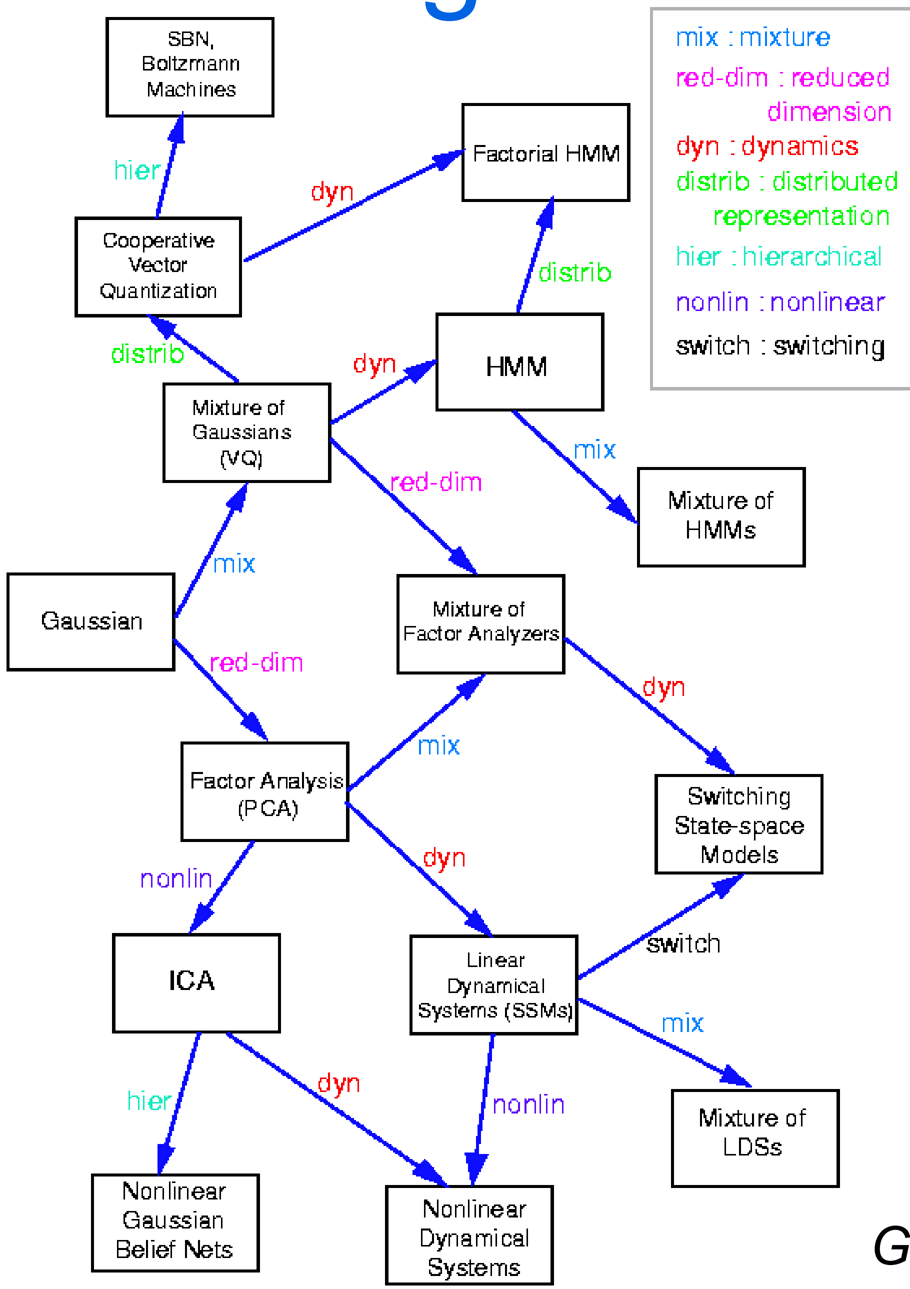
Isard et. al., 1998

Particle Filtering Caveats

- Particle filters are easy to implement, and effective in many applications, BUT
  - It can be difficult to know how many samples to use, or to tell when the approximation is poor
  - Sometimes suffer catastrophic failures, where NO particles have significant posterior probability
  - This is particularly true with "peaky" observations in high-dimensional spaces:



The Big Picture



Ghahramani & Roweis