CSC535: Probabilistic Graphical Models

Dynamical Systems

Prof. Jason Pacheco
(Some material from Prof. Kobus Barnard)

Administrivia

- HW3 Extension (tonight @ 11:59pm)
- -10pts per day after that, through Friday
- Monday, Midterm review

Outline

- Sequence Models
- Linear Dynamical Systems
- LDS Extensions

Outline

- Sequence Models
- Linear Dynamical Systems
- LDS Extensions

Sequential data

Much of this section follows Bishop chapter 13

See also Murphy chapters 17 and 18

Sequential data

Sequential data has order, and the order matters.

What has happened, informs what will happen.

Sequential data is everywhere.

Examples:

- spoken language (word production)
- written language (sentence level statistics)
- weather
- human movement
- stock market data
- genomes

Sequential data

Graphical models for such data?

The complexity of the representation seems to increase with time.

Observations over time tend to depend on the past.

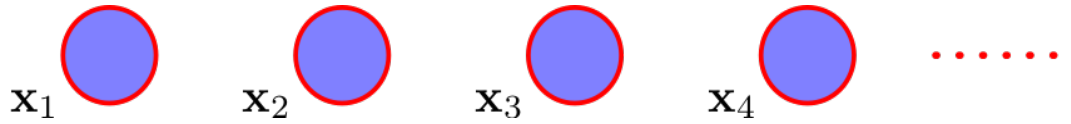
We can simply life by assuming that the distant past does not matter.

If we assume that history does not matter other than the immediate previous entity, we have a first order Markov model.

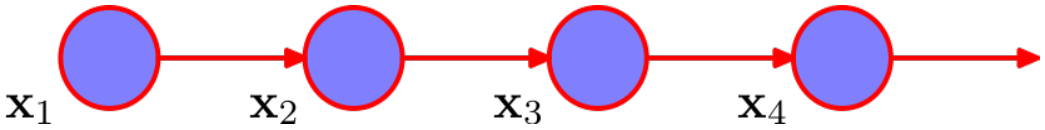
If what happens now depends on two previous entities, we have a second order Markov model.

Markov chains

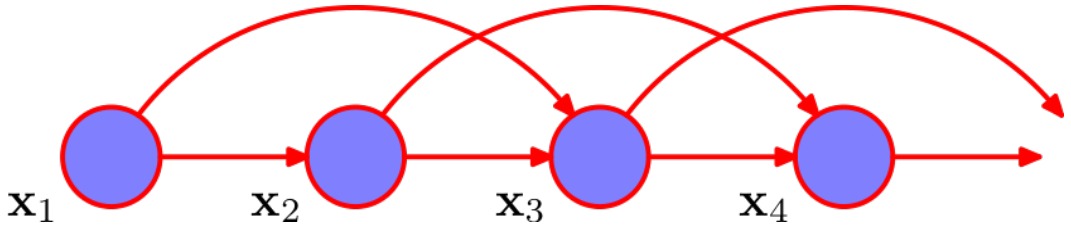
Zeroth order



First order

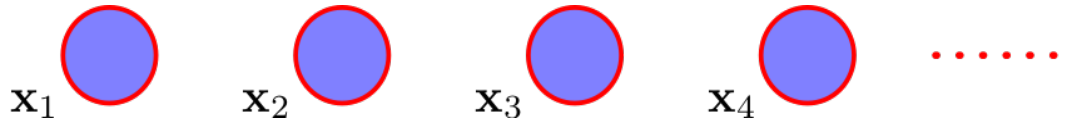


Second order

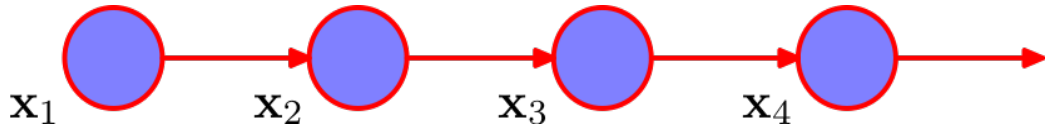


Markov chains

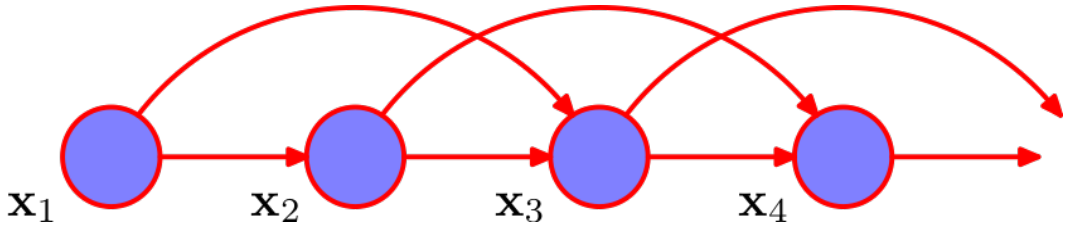
Zeroth order



First order



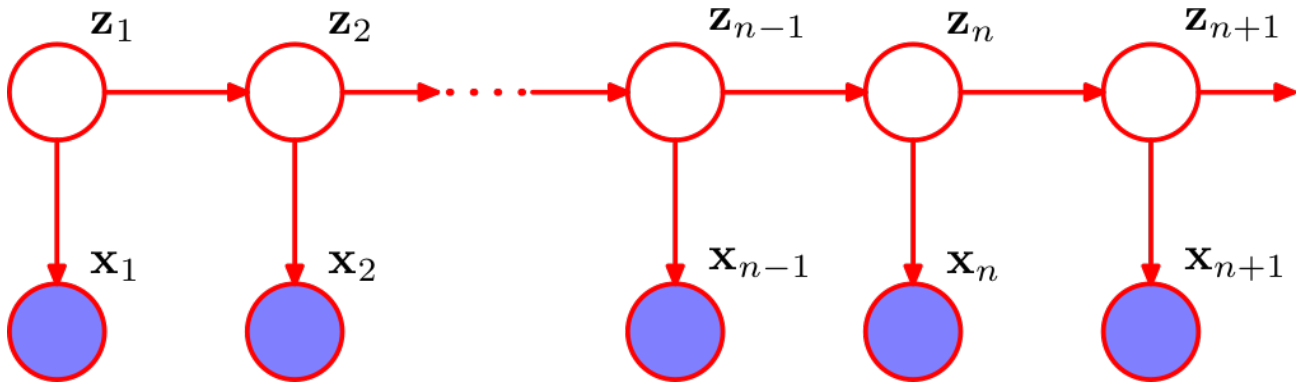
Second order



Notice that this plan has arrows **from** data-this complicates model specification

Hidden Markov Model (HMM)

Introducing latent state simplifies modeling data likelihood…



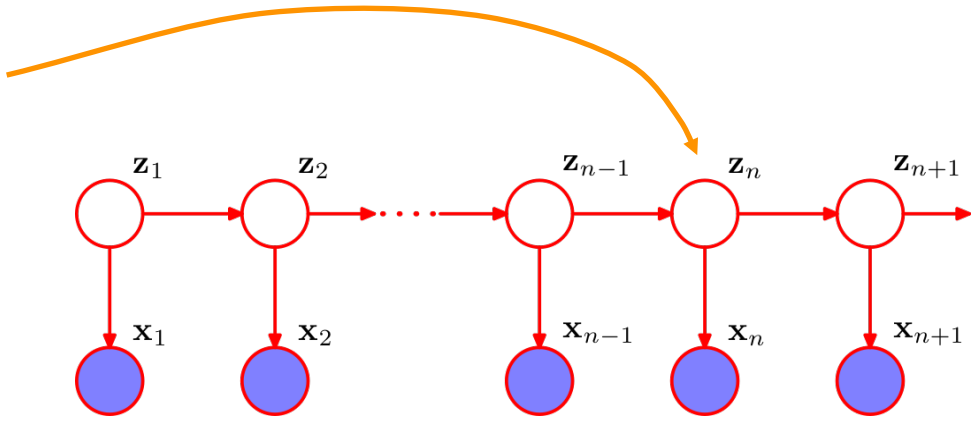
Intuition Temporal extension of mixture model

- Data clusters into hidden "states" Z at each time
- Hidden state encodes important part of history
- Markov chain models transitions among clusters

Markovian assumptions

The basic HMM is like a mixture model, with the mixture component being used for the current observations depends on the last previous component.

$$z_{n+1} \perp z_{n-1} \mid z_n$$

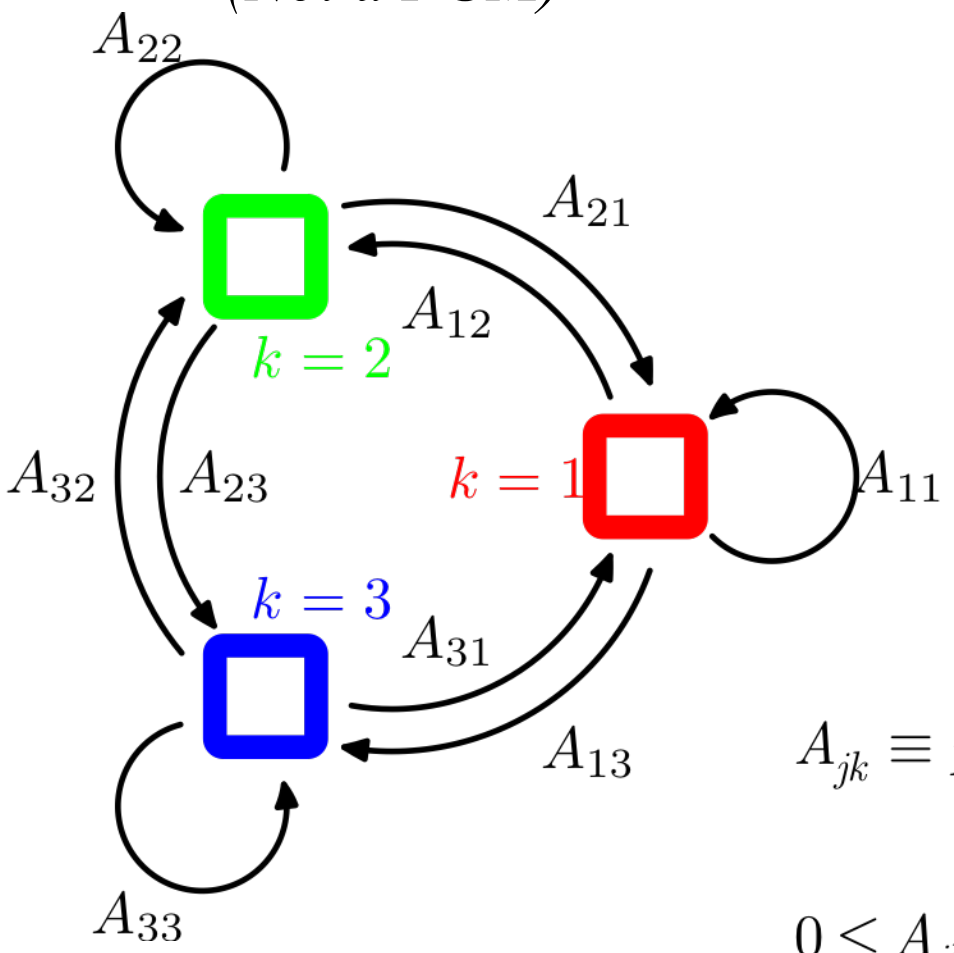


Observation likelihood $p(x_n \mid z_n)$ models how data from a component are generated (things are *easy* if you know the cluster)

Transition Dynamics

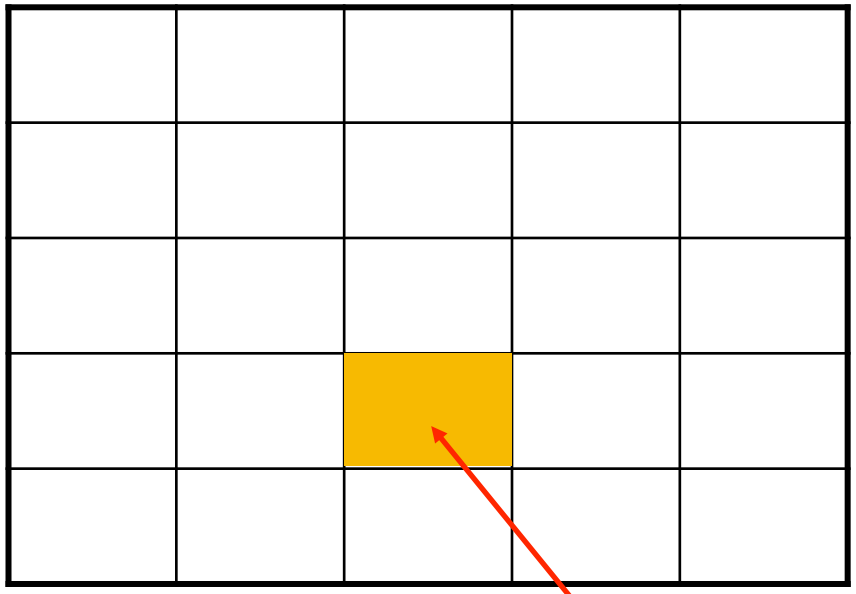
State Transition Diagram

(Not a PGM)



<—next state —>

current state



$$A_{jk} \equiv p\left(z_{nk}=1 \middle| z_{n-1,j}=1\right)$$

A_{jk} : Probability of going from state j=4 to state k=3

$$0 \le A_{jk} \le 1 \quad \text{and} \quad \sum_k A_{jk} = 1$$

Starting state

Our HMM will be a pure* generative model, so we need to know how to start.

$$\pi_k \equiv p\left(z_{1k} = 1\right)$$

$$\text{with } 0 \leq \pi_k \leq 1 \quad \text{and} \quad \sum_k \pi_k = 1$$

*By pure, I mean that we can do ancestral sampling, i.e., a Bayes net.

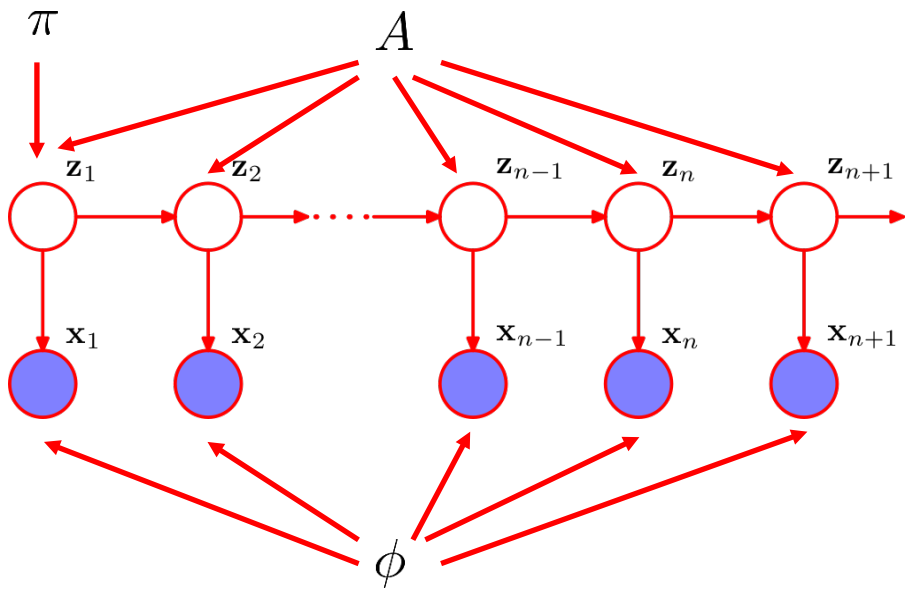
HMM parameter summary

$\Theta = \{\pi, A, \phi\}$

π is probability over initial states

A is transition matrix

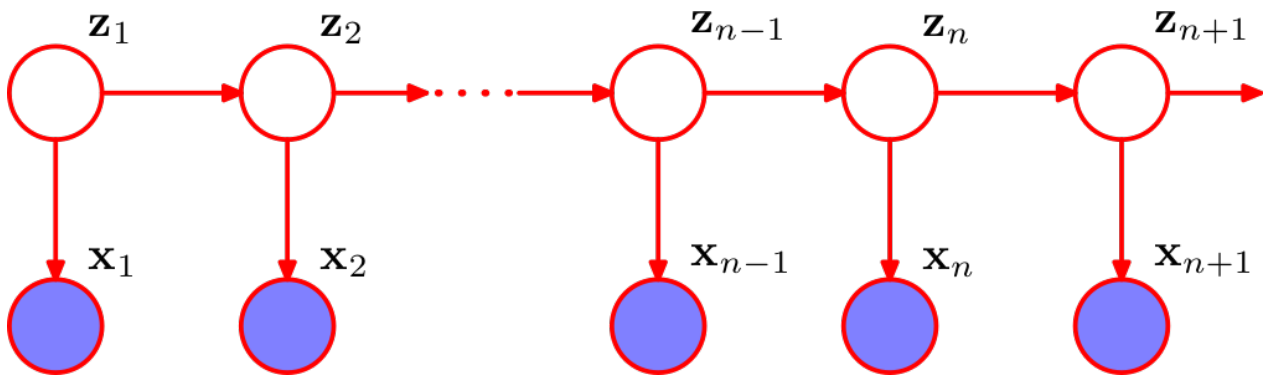
ϕ are the parameters for the data emmission probabilities (i.e., for $p(\mathbf{x}_n | z_n)$, e.g., means of Gaussians)



Data distribution from an HMM

Let $X = \{\mathbf{x}_n\}$ be the observed sequence

$$p(X|\theta) \quad = ? \qquad .$$



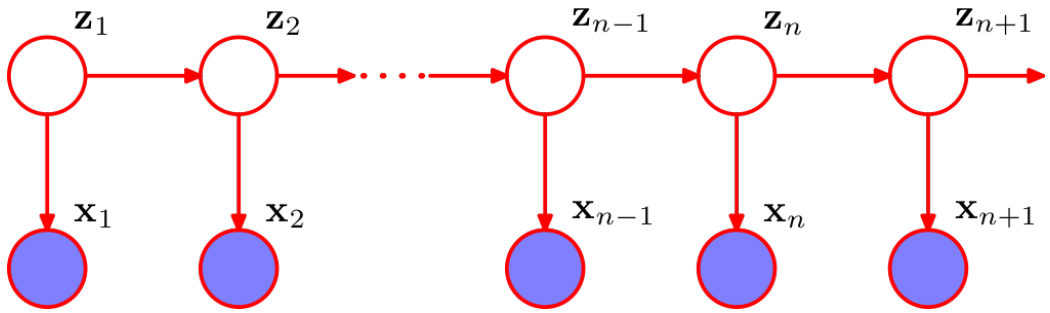
Data distribution from an HMM

Let $X = \{\mathbf{x}_n\}$ be the observed sequence

$p(X|\theta)$ is a marginalization over Z.

$p(X,Z|\theta) = ?$

Data distribution from an HMM



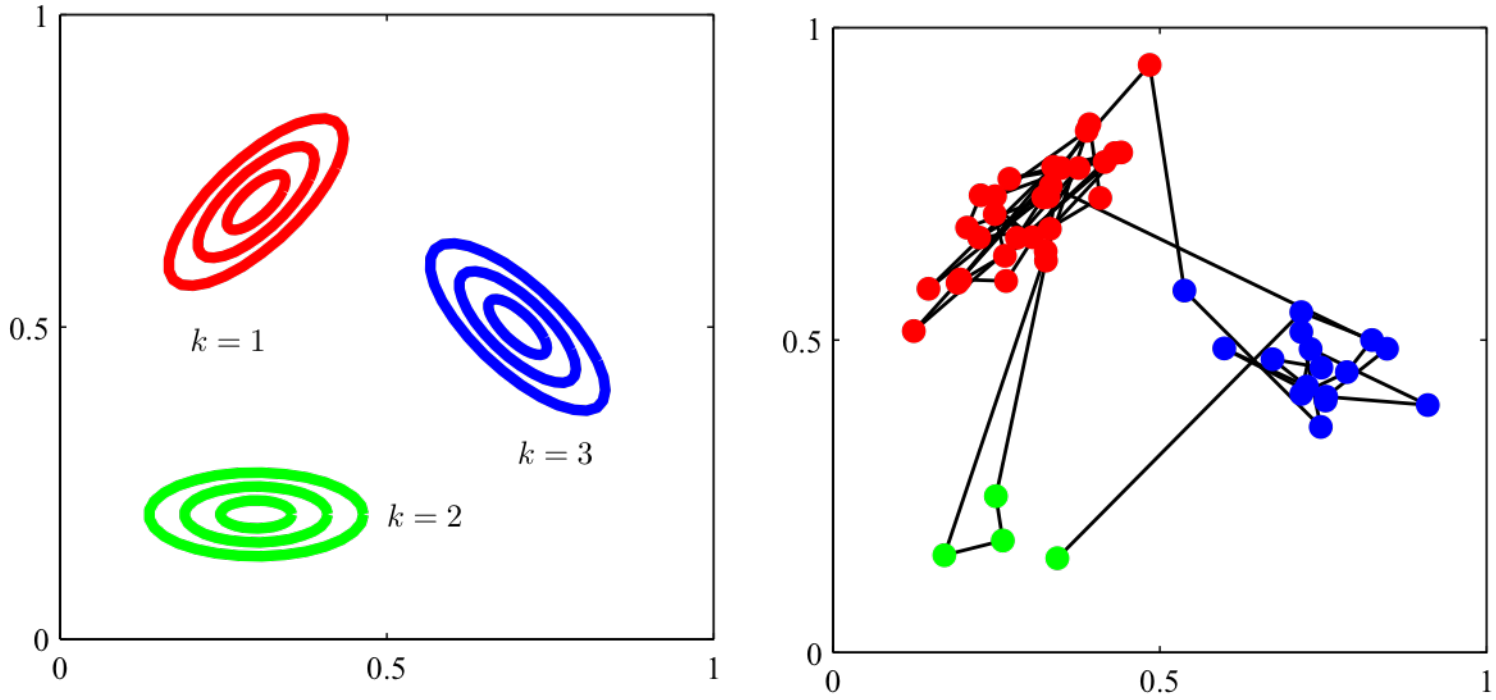
An HMM is specified by: $\theta = \{\pi, A, \phi\}$

$$p(X, Z|\theta) = p(z_1|\pi)\left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A)\right]\left[\prod_{n=1}^{N} p(\mathbf{x}_n|z_n, \phi)\right]$$

(complete data, i.e., we can generate from this).

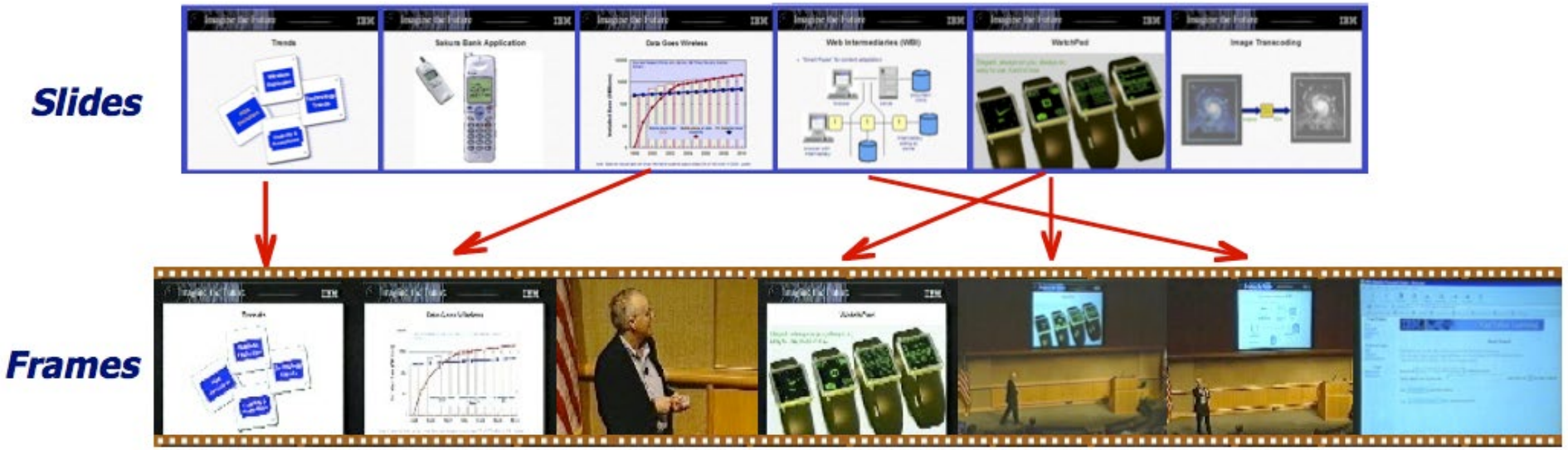
Data distribution from an HMM

Gaussian likelihood model $p(x_n|z_n)$

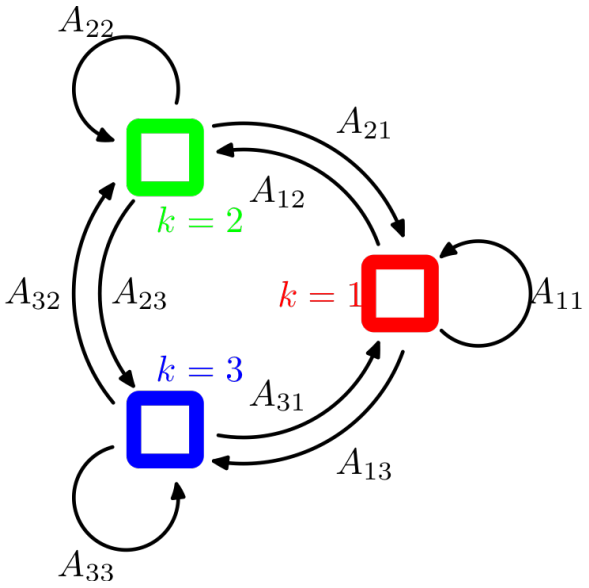


Transition probability to another state is 5% (from Bishop—the short visits in green seem a bit anomalous).

Example: Matching slides to video frames



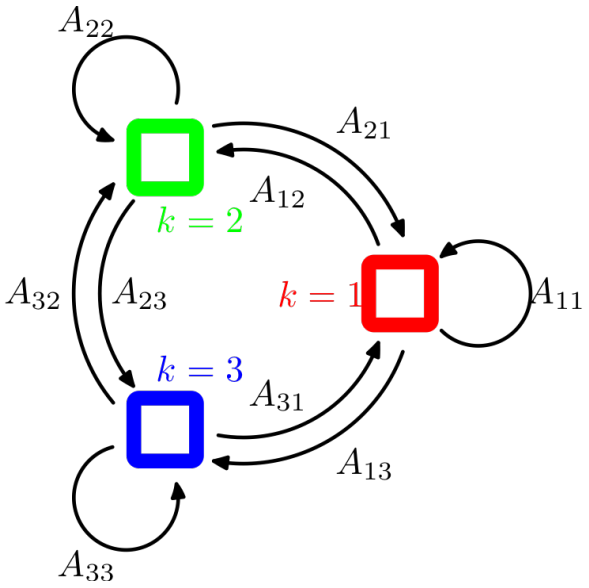
Matching slides to video frames



Our state sequence corresponds to what slide is being shown.

We assume that only the jump matters. IE, going from slide 6 to 8 has the same chance of going from 10 to 12.

Matching slides to video frames



Our state sequence corresponds to what slide is being shown.

$$p(X, Z|\theta) = p(z_1|\pi)\left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A)\right]\left[\prod_{m=1}^{N} p(x_m|z_m, \phi)\right]$$

Image matching likelihood

Classic HMM computational problems

1. Given unlabeled* data, what is the HMM (**parameter learning**).
2. Given an HMM and observed data, what is the **probability distribution of states** for each time point (z_n in our notation).
3. Given an HMM and observed data, what is the most likely **state sequence**?

*We could have data with labels (annotated) which means this step becomes trivial, much like training naive Bayes versus fitting a GMM using EM. This is what we did in the matching slides to video project.

Classic HMM computational problems

1. Given unlabeled* data, what is the HMM (**parameter learning**).
2. Given an HMM and observed data, what is the **probability distribution of states** for each time point (z_n in our notation).
3. Given an HMM and observed data, what is the most probable **state sequence**?

#2 and #3 seem similar, but to understand the difference consider a three state system about doing HW problems A, B, and C in order, with B being very easy. So you will spend most of your time in state A and C. State B may be the least likely state for every time point. But the most likely state sequence must include it.

Classic HMM computational problems

1. Given unlabeled data, what is the HMM (**learning**).

   This is a missing value problem, which we can tackle using EM, but we will need to solve #2 as sub-problem.

2. Given an HMM and data, what is the **probability distribution of states** for each time point (z_n in our notation).

   These are marginals in a Bayes net, and so we use the sum-product algorithm (in HMM often called alpha-beta or forwards backwards).

3. Given an HMM and data, what is the most likely **state sequence**?

   This is a maximal configuration of a Bayes net, and so we use max-sum (in HMM this is Viterbi).

Classic HMM computational problems

1. Given unlabeled data, what is the HMM (**learning**).

   This is a missing value problem, which we can tackle using EM, but we will need to solve #2 as sub-problem.

2. Given an HMM and data, what is the **probability distribution of states** for each time point (z_n in our notation).

   These are marginals in a Bayes net, and so we use the sum-product algorithm (in HMM often called alpha-beta or forwards backwards).

3. Given an HMM and data, what is the most likely **state sequence**?

   This is a maximal configuration of a Bayes net, and so we use max-sum (in HMM this is Viterbi).

Learning the HMM using EM (Baum-Welch Algorithm)

A natural first “hammer” for HMM is EM because the key issue is our ignorance about the correspondence between data and latent state.

But there are many other ways to do the learning that are less susceptible to local optima (at the expense of CPU time)

Learning the HMM using EM (AKA Baum-Welch)

Bishop chapter 13 only considers earning from one sequence.

Often you have many shorter sequences, and extending to multiple training sequences is easy (so we will do that).

Depending on what you want your model to do, the prior over initial state is the overall prior (you can jump in anywhere) or it really matters where you start.

I will extend Bishop's notation to have E training sequence examples (or experiments) indexed by e, with each sequence having N_e points*, and we will assume where start matters.

*It is perhaps more natural to index sequences by n and time steps within sequences with t (this is what Murphy does, page 618), but this leads to too much notational conflict with our material.

Learning the HMM with EM (sketch)

If we know the state distributions, and the successive state pair distributions (needed for the transition matrix, A), for each training sequence, we can compute the parameters.

If we know the parameters, we can compute the state distributions, for each training sequence (this is HMM computation problem #2, which we need to solve as a subproblem if we use EM).

Blue text highlights differences from the mixture model for one sequence.

Green text reminds us that we need a bit of book-keeping when we train on multiple sequences.

Recall the General EM algorithm

1. Choose initial values for $\theta^{(s=1)}$

   (can also do assignments, but then jump to M step).

2. E step: Evalute $p\left(Z \middle| X, \theta^{(s)}\right)$

3. M step: Evalute $\theta^{(s+1)} = \arg\max_{\theta} \left\{ \mathrm{Q}\left(\theta^{(s+1)}, \theta^{(s)}\right) \right\}$

   where $\mathrm{Q}\left(\theta^{(s+1)}, \theta^{(s)}\right) = \sum_{Z} p\left(Z \middle| X, \theta^{(s)}\right) \log\left(p\left(X, Z \middle| \theta^{(s+1)}\right)\right)$

4. Check for convergence; If not done, goto 2.

★ At each step, our objective function increases unless it is at a local maximum. It is important to check this is happening for debugging!

HMM complete data likelihood (one sequence)

$$p(X,Z|\theta) = p(z_1|\boldsymbol{\pi})\left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A)\right]\prod_{n=1}^{N} p(x_n|z_n, \phi)$$

HMM complete data likelihood (one sequence)

$$p(X,Z|\theta) = p(z_1|\boldsymbol{\pi})\left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A)\right]\prod_{n=1}^{N} p(x_n|z_n, \phi)$$

$$= \prod_{k=1}^{K} \pi_k^{z_{1,k}} \left[\prod_{n=2}^{N}\prod_{j=1}^{K}\prod_{k=1}^{K} A_{j,k}^{z_{n-1,j} \bullet z_{n,k}}\right] \prod_{n=1}^{N}\prod_{k=1}^{K} \left(p(x_n|\phi_k)\right)^{z_{n,k}}$$

Remember our "indicator variable" notation. Z is a particular assignment of the missing values (i.e., which cluster the HMM was in at each time. For each time point, *n*, one of the values of z_n is one, and the others are zero. So, it "selects" the factor for the particular state at that time.

HMM complete data likelihood (one sequence)

$$p(X,Z|\theta) = p(z_1|\boldsymbol{\pi})\left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A)\right]\prod_{n=1}^{N} p(x_n|z_n, \phi)$$

$$= \prod_{k=1}^{K} \pi_k^{z_{1,k}} \left[\prod_{n=2}^{N}\prod_{j=1}^{K}\prod_{k=1}^{K} A_{j,k}^{z_{n-1,j} \bullet z_{n,k}}\right] \prod_{n=1}^{N}\prod_{k=1}^{K} \left(p(x_n|\phi_k)\right)^{z_{n,k}}$$

$$\log\left(p(X,Z|\theta)\right) = \sum_{k=1}^{K} z_{1k} \log(\pi_k) + \sum_{n=2}^{N}\sum_{j=1}^{K}\sum_{k=1}^{K} z_{n-1,j} z_{n,k} \log(A_{j,k}) + \sum_{n=1}^{N}\sum_{k}^{K} z_{n,k} \log\left(p(x_n|\phi_k)\right)$$

(complete data log-likelihood)

HMM complete data likelihood (E training sequences)

$$p(X,Z|\theta) = \prod_{e=1}^{E} p(z_{e,1}|\boldsymbol{\pi}) \left[\prod_{e=1}^{E} \prod_{n=2}^{N_e} p(z_{e,n}|z_{e,n-1}, A) \right] \prod_{e=1}^{E} \prod_{n=1}^{N_e} p(x_{e,n}|z_{e,n}, \phi)$$

$$= \prod_{e=1}^{E} \prod_{k=1}^{K} \pi_k^{z_{s,1,k}} \left[\prod_{e=1}^{E} \prod_{n=2}^{N_e} \prod_{j=1}^{K} \prod_{k=1}^{K} A_{j,k}^{z_{n-1,j} \bullet z_{n,k}} \right] \prod_{e=1}^{E} \prod_{n=1}^{N_e} \prod_{k=1}^{K} \left(p(x_{e,n}|\phi_k) \right)^{z_{e,n,k}}$$

$$\log\left(p(X,Z|\theta)\right) = \sum_{e=1}^{E} \sum_{k=1}^{K} z_{s,1k} \log(\pi_k) + \sum_{e=1}^{E} \sum_{n=2}^{N_e} \sum_{j=1}^{K} \sum_{k=1}^{K} z_{e,n-1,j} z_{e,n,k} \log(A_{j,k}) + \sum_{e=1}^{E} \sum_{n=1}^{N_e} \sum_{k}^{K} z_{e,n,k} \log\left(p(x_{e,n}|\phi_k)\right)$$

(complete data log likelihood)

Learning the HMM with EM (sketch)

In the simple clustering case (e.g., GMM), the E step was simple. For HMM it is a bit more involved.

The M step works a lot like the GMM, except we need to deal with successive states. Consider the M step first.

E-Step for HMM

Provides two distributions (responsibilities)…

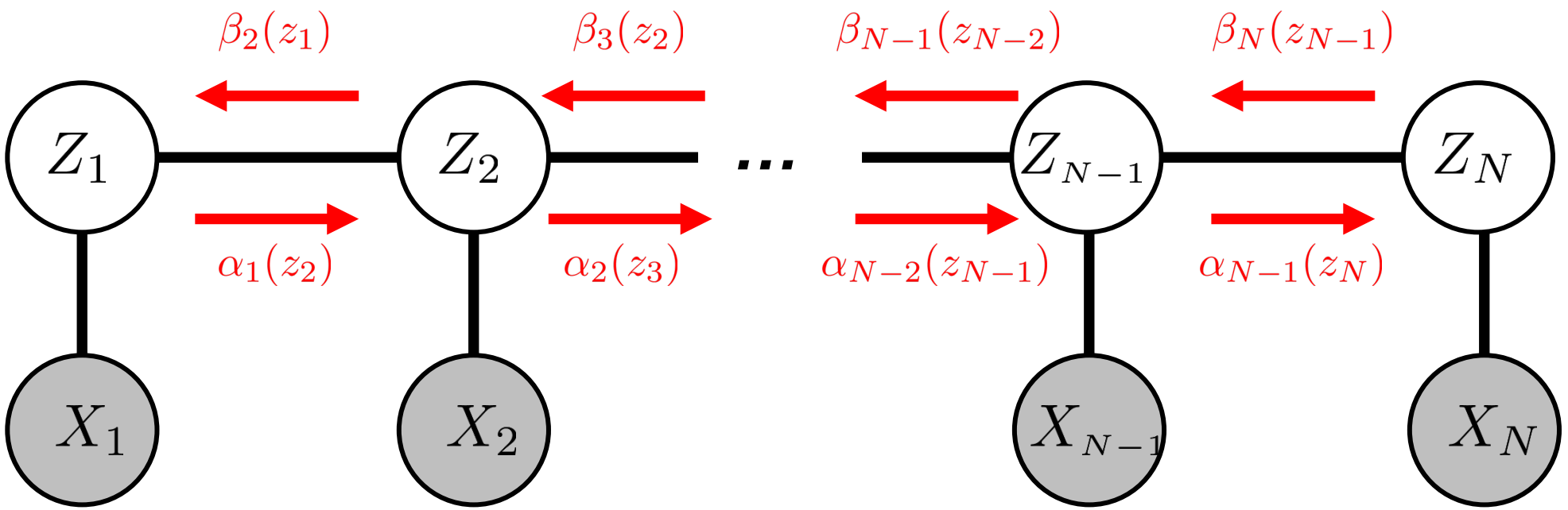
The degree each state explains each data point (analogous to GMM responsibilities). $\gamma\left(z_{e,n,k}\right) = p\left(z_{e,n,k} \middle| X_e, \theta^{(s)}\right)$

The degree that the combination of a state, and a previous one explain two data points.

$$\xi\left(e, \boldsymbol{z}_{n-1,j}, \boldsymbol{z}_{n,k}\right) = p\left(\boldsymbol{z}_{e,n-1,j}, \boldsymbol{z}_{e,n,k} \middle| X_e, \theta^{(s)}\right)$$

Q: How can we compute one / two stage-marginals in HMM?

Forward-Backward Algorithm



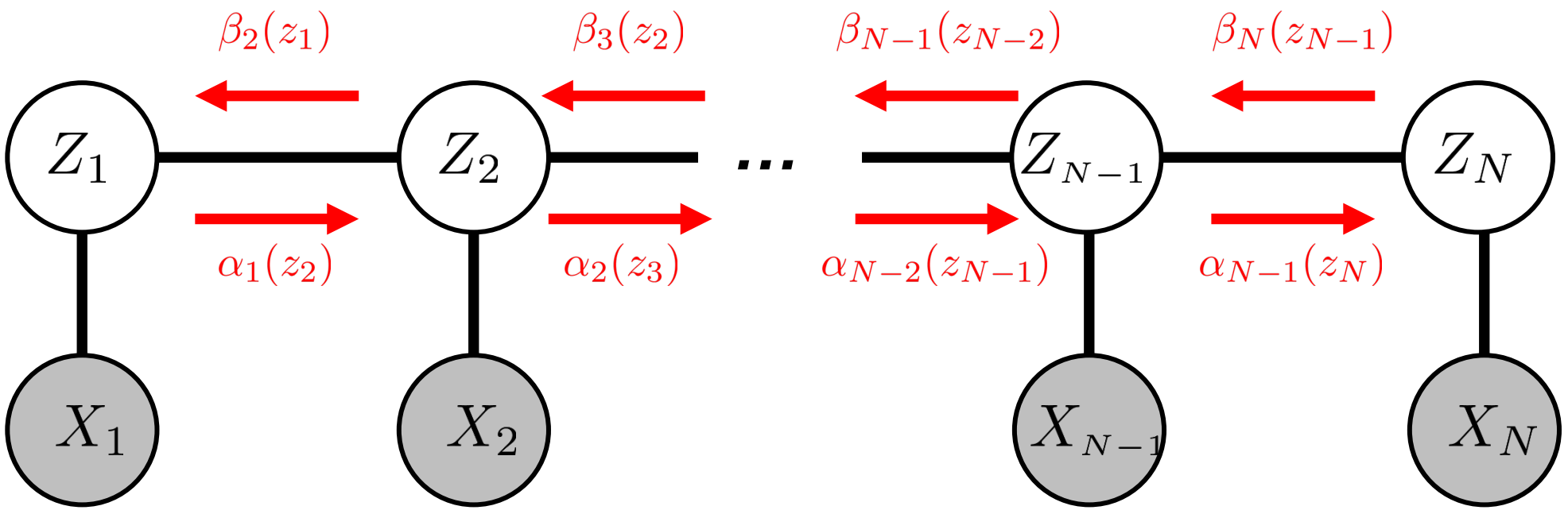
Forward message:

$$\alpha_{n-1}(z_n) = \psi(z_n, x_n) \sum_{z_{n-1}} \alpha_{n-2}(z_{n-1}) \psi(z_{n-1}, z_n)$$

Forward message:

$$\beta_{n+1}(z_n) = \sum_{z_{n+1}} \beta_{n+2}(z_{n+1}) \psi(z_n, z_{n+1}) \psi(z_{n+1}, x_{n+1})$$

Forward-Backward Algorithm



Node Responsibility (a.k.a. marginal):

$$\gamma(z_n) = p(z_n \mid \mathcal{X}) \propto \alpha_n(z_n)\beta_n(z_n)$$

Two-Node Responsibility (a.k.a. pairwise marginal):

$$\xi(z_n, z_{n+1}) = p(z_n, z_{n+1} \mid \mathcal{X}) \propto \alpha_n(z_n)\psi(z_n, z_{n+1})\beta_{n+1}(z_{n+1})$$

M-Step for HMM

Recall the complete data log-likelihood:

$$\log\left(p\left(X,Z\middle|\theta\right)\right)=\sum_{e=1}^{E}\sum_{k=1}^{K}z_{e,1k}\log\left(\pi_k\right)+\sum_{e=1}^{E}\sum_{n=2}^{N_e}\sum_{j=1}^{K}\sum_{k=1}^{K}z_{e,n-1,j}z_{e,n,k}\log\left(A_{j,k}\right)+\sum_{e=1}^{E}\sum_{n=1}^{N_e}\sum_{k}^{K}z_{e,n,k}\log\left(p\left(x_{e,n}\middle|\phi_k\right)\right)$$

M-Step for HMM

Recall the complete data log-likelihood:

$$\log\left(p\left(X,Z\middle|\theta\right)\right)=\sum_{e=1}^{E}\sum_{k=1}^{K}z_{e,1k}\log\left(\pi_k\right)+\sum_{e=1}^{E}\sum_{n=2}^{N_e}\sum_{j=1}^{K}\sum_{k=1}^{K}z_{e,n-1,j}z_{e,n,k}\log\left(A_{j,k}\right)+\sum_{e=1}^{E}\sum_{n=1}^{N_e}\sum_{k}^{K}z_{e,n,k}\log\left(p\left(x_{e,n}\middle|\phi_k\right)\right)$$

Expected complete data log-likelihood:

$$\begin{aligned}Q\left(\theta^{(s+1)},\theta^{(s)}\right)&=\sum_{z}p\left(Z\middle|\theta^{(s)}\right)\log\left(p\left(X,Z\middle|\theta^{(s+1)}\right)\right)\\&=\sum_{e=1}^{E}\sum_{k=1}^{K}\gamma\left(z_{e,1,k}\right)\log\left(\pi_k\right)+\sum_{e=1}^{E}\sum_{n=2}^{N_e}\sum_{j=1}^{K}\sum_{k=1}^{K}\xi\left(e,z_{n-1,j}z_{n,k}\right)\log\left(A_{j,k}\right)+\sum_{e=1}^{E}\sum_{n=1}^{N_e}\sum_{k}^{K}\gamma\left(z_{e,n,k}\right)\log\left(p\left(x_{e.n}\middle|z_{e,n},\phi\right)\right)\end{aligned}$$

M-Step for HMM

Doing the maximization using Lagrange multipliers (or intuition) gives us

$$\pi_k = \frac{\sum_{e=1}^{E} \gamma\left(z_{e,1,k}\right)}{\sum_{e=1}^{E} \sum_{k'} \gamma\left(z_{e,1,k'}\right)}$$

$$A_{jk} = \frac{\sum_{e=1}^{E} \sum_{n=2} \zeta\left(e, z_{n-1,j}, z_{n,k}\right)}{\sum_{e=1}^{E} \sum_{k'} \sum_{n=2} \zeta\left(z_{n-1,j}, z_{n,k'}\right)}$$

Much like the GMM. Taking the partial derivative for π_k kills second and third terms.

M-Step for HMM

The maximization of $p\left(\mathbf{x}_{e,n} \middle| \phi\right)$ is exactly the same as the mixture model.

For example, if we have Gaussian emmisions, then

$$\boldsymbol{\mu}_k = \frac{\sum_{e=1}^{E} \sum_{n} \mathbf{x}_{e,n} \, \gamma\left(z_{e,n,k}\right)}{\sum_{e=1}^{E} \sum_{n} \gamma\left(z_{e,n,k}\right)}$$

Classic HMM computational problems

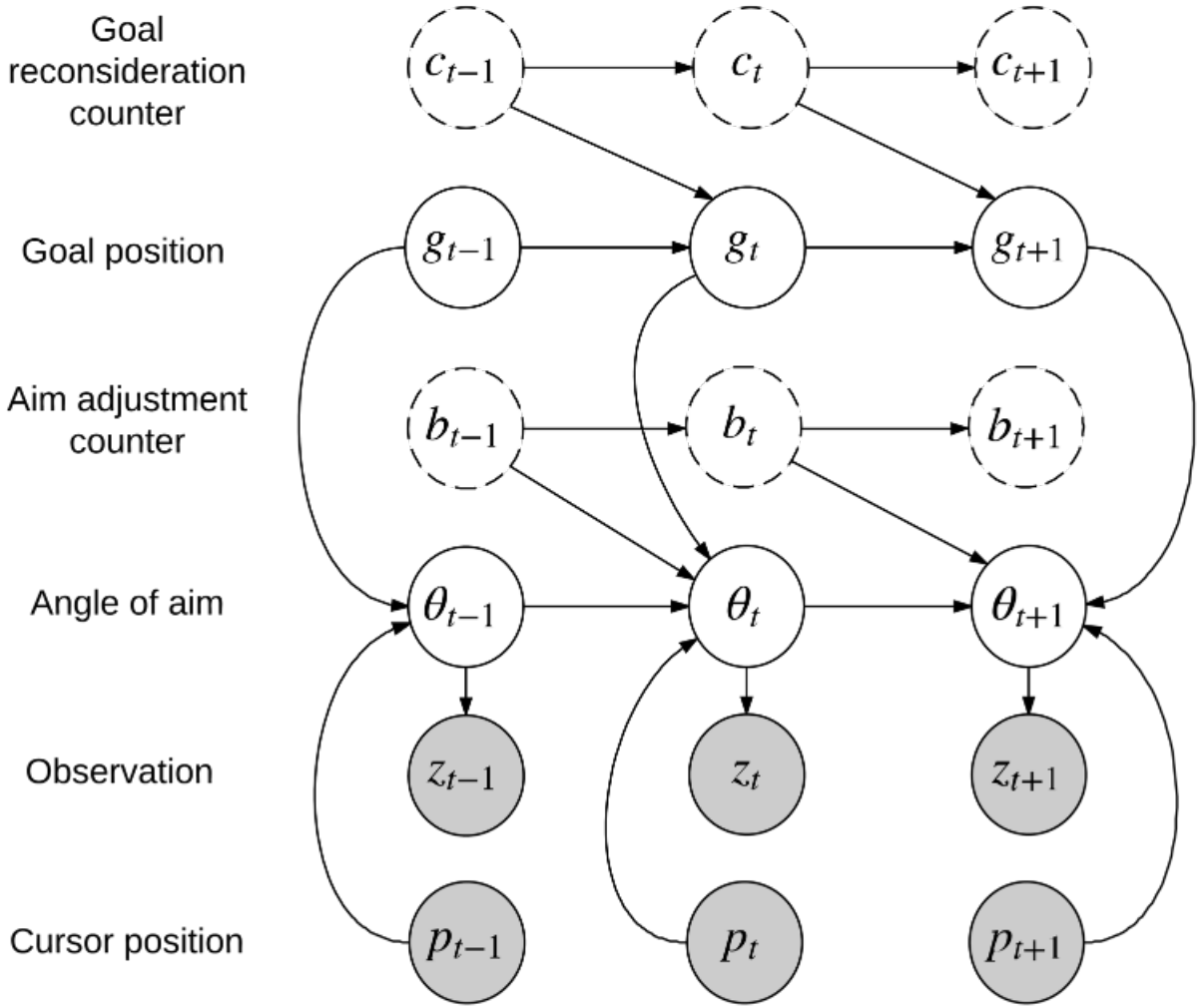
Given data, what is the HMM (**learning**). ✓

Given an HMM, what is the **distribution over the state** variables. Also, **how likely** are the observations, given the model. ✓

Given an HMM, what is the most probable **state sequence** for some data?

Semi-Markov Process

Intracortical Brain-Computer Interface



Block 12: "Multiscale Semi-Markov Model"

- Counter decrements deterministically:
  $c_t = c_{t-1} - 1 \quad \text{if} \quad c_{t-1} > 0$
- Resample when exhausted:
  $c_t \sim \text{Some-PMF}(\cdot) \quad \text{if} \quad c_{t-1} = 0$
- Controls dynamics: $g_t \mid c_{t-1} \sim p_{c_{t-1}}(\cdot)$

Viterbi algorithm (special case of max-sum)

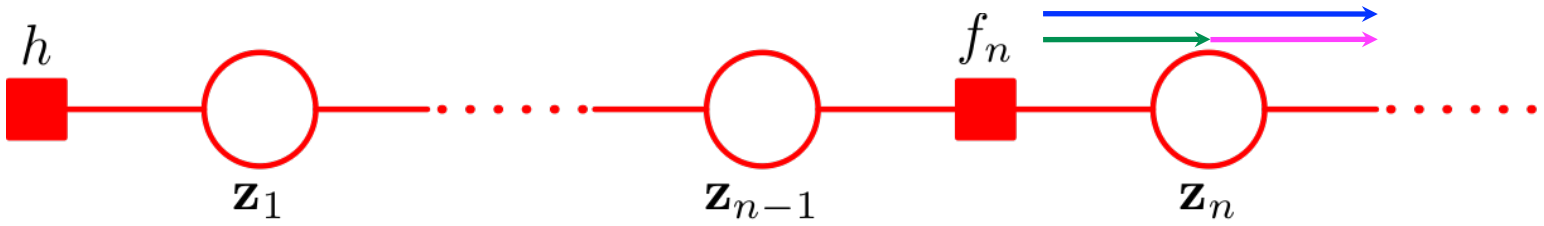
Recall max-sum

Forward direction is like sum-product, except

- Factor nodes take the max over logs instead of sum
- Variable nodes use sum of logs instead of product
- We remember incoming variable values* that give max

Backwards direction is simply backtracking on (*).

Recall sum-product for HMM

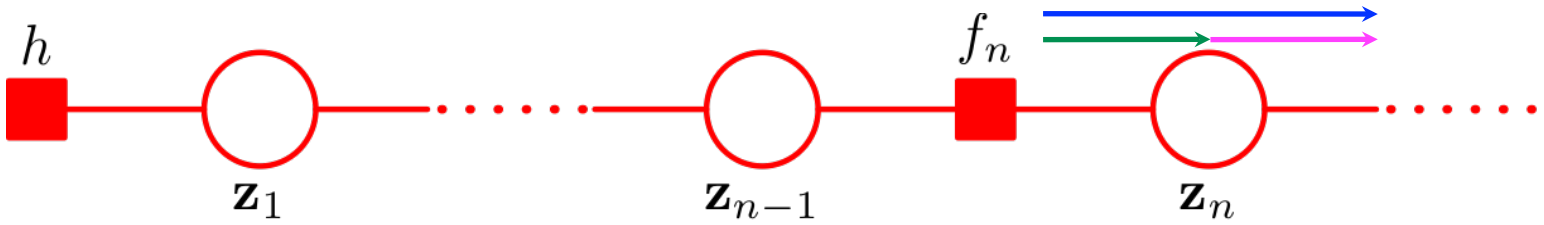


If we identify $\mu_{f_n \to f_{n+1}}(z_n) = \alpha(z_n)$

$$\alpha(z_1) = p(z_1)p(x_1|z_1)$$

$$\begin{aligned}\alpha(z_n) &= \sum_{z_{n-1}} f_n(z_{n-1}, z_n)\alpha(z_{n-1}) \\ &= \sum_{z_{n-1}} p(z_n|z_{n-1})p(x_n|z_n)\alpha(z_{n-1}) \\ &= p(x_n|z_n)\sum_{z_{n-1}} p(z_n|z_{n-1})\alpha(z_{n-1})\end{aligned}$$

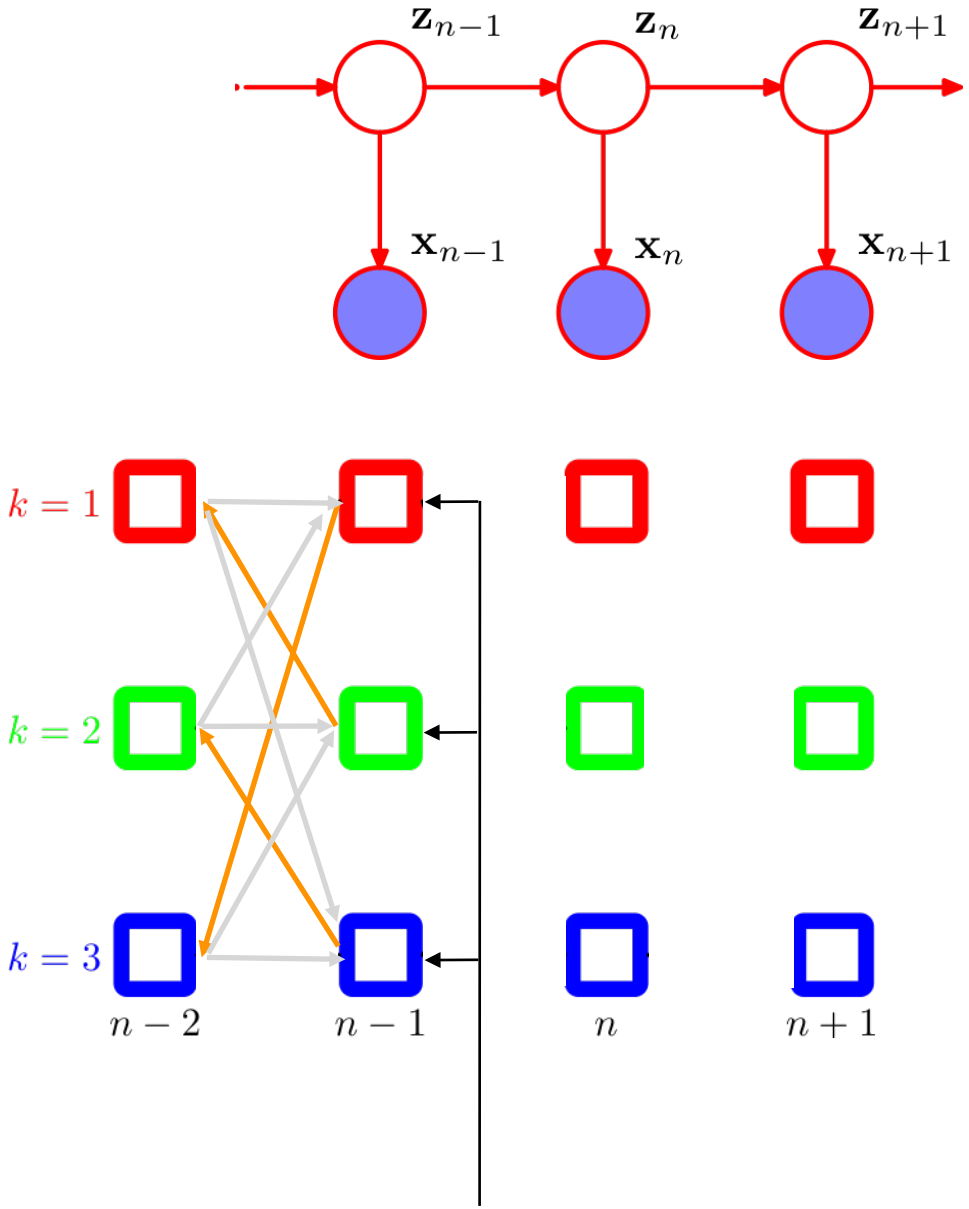
By analogy we get max sum



Left to right messages for max-sum

$$\omega(z_n) = \log(p(x_n|z_n)) + \max_{z_{n-1}} \{\log(p(z_n|z_{n-1})) + \omega(z_{n-1})\}$$

$$\omega(z_1) = \log(p(z_1)) + \log(p(x_1|z_1))$$



Squares represent being in each of the three states at a given time.

We store the log of the probability of the maximal likely way to get there.

And the particular previous state that gave the max (orange)

Story for the preceding picture

Consider all possible paths **to each** of the K states for time n.

The message encodes the probabilities for the maximum probability path for each of the K states.

I.E., for a given time, for each state k, it records is the probability of being there by via the maximal probably sequence.

That value is (recursively defined by)

$$\omega(z_n) = \log\left(p(x_n|z_n)\right) + \max_{z_{n-1}} \left\{\log\left(p(z_n|z_{n-1})\right) + \omega(z_{n-1})\right\}$$

Story (continued)

The incoming message is the vector of probabilities for the maximum probability path for each of the K states at the previous time.

$$\omega(z_n) = \log\left(p(x_n|z_n)\right) + \max_{z_{n-1}} \left\{\log\left(p(z_n|z_{n-1})\right) + \omega(z_{n-1})\right\}$$

For each state k

Consider getting there from each previous state k'

This gives the maximal probability way to be in each of K states, k, at time n.

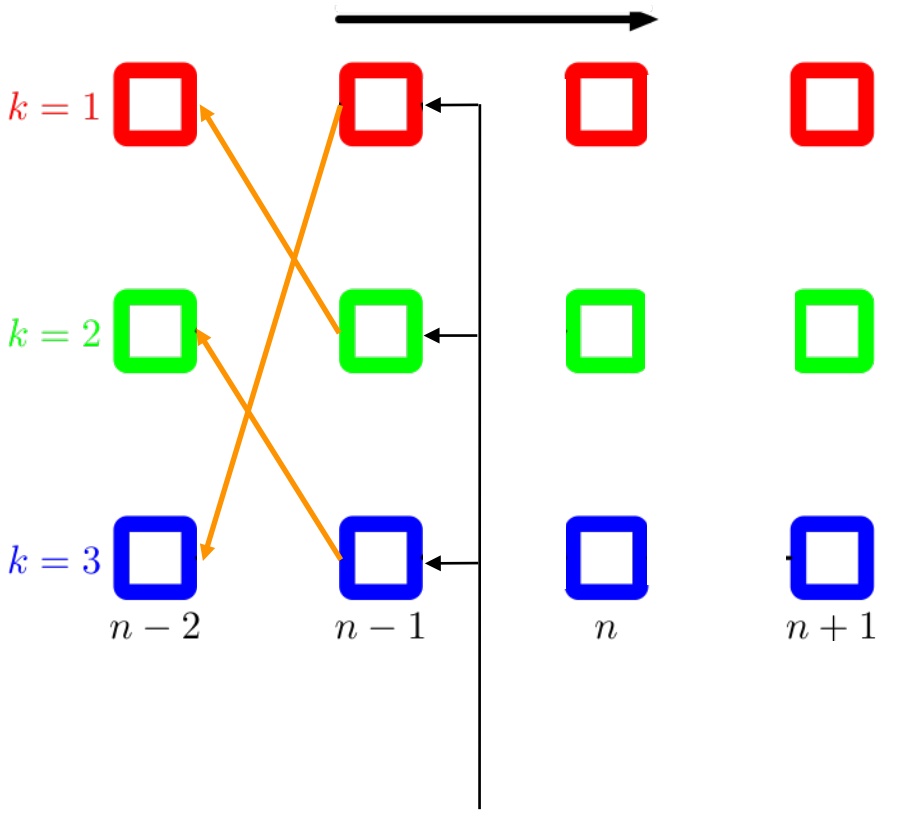
Story (continued)

For Viterbi, we remember the previous state, *k'*, leading to the max for each *k*. (This is simpler than the general case because no branches).

Once we know the end state of the maximal probability path, we can find the maximal probability path by back-tracking.

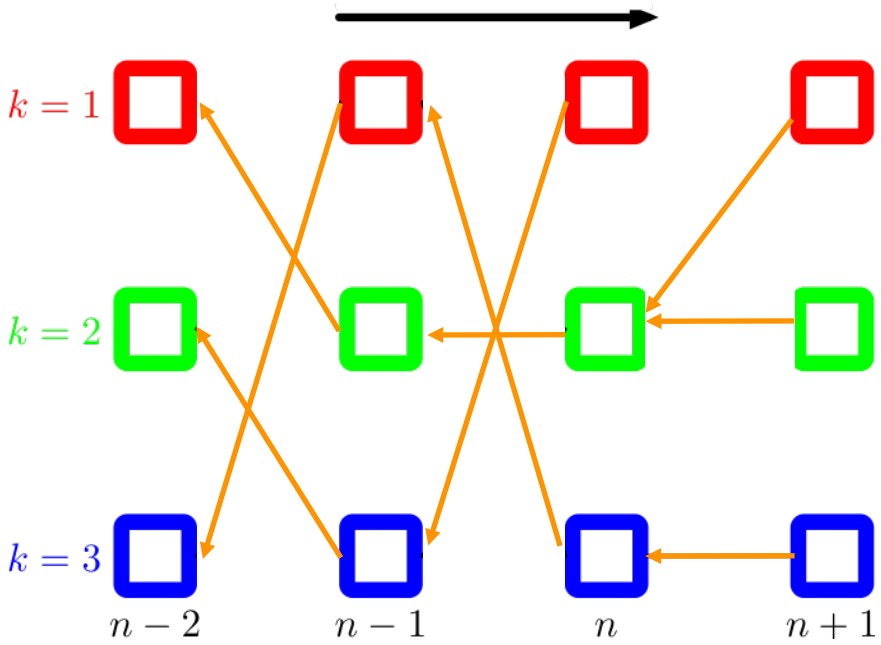
You might also recognize this as dynamic programming (think minimum cost path).

Intuitive understanding



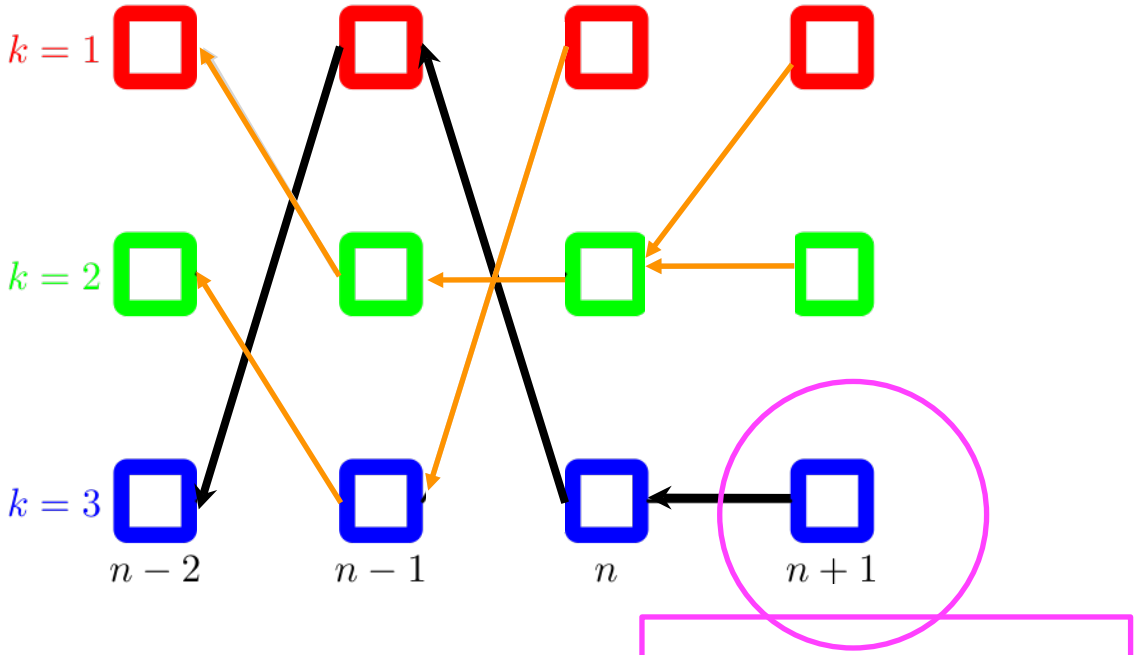
Store $\omega_{n-1,k}$ and $k' = \arg\max(\bullet)$ for k

Intuitive understanding



If this is the end, we now know the max, and what the ending state is.

Intuitive understanding



Suppose the max is attained at *k*=*3*.

The max path is dark (but we only know it when we get to the end).

To find the path, we need to chase the back pointers.

Classic HMM computational problems

Given data, what is the HMM (**learning**). ✓

Given an HMM, what is the **distribution over the state** variables. Also, **how likely** are the observations, given the model. ✓

Given an HMM, what is the most probable **state sequence** for some data? ✓

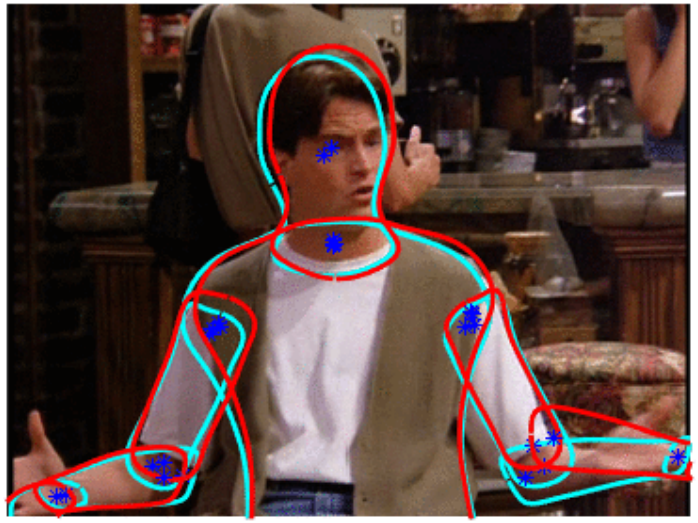
Outline

- Sequence Models
- Linear Dynamical Systems
- LDS Extensions

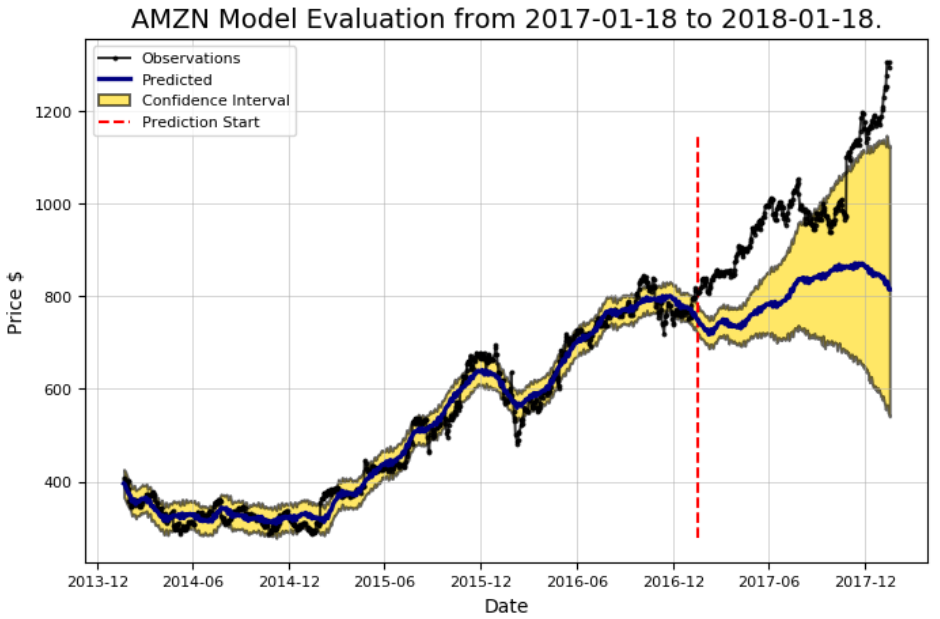
Dynamical System

Models of latent states evolving over time/space

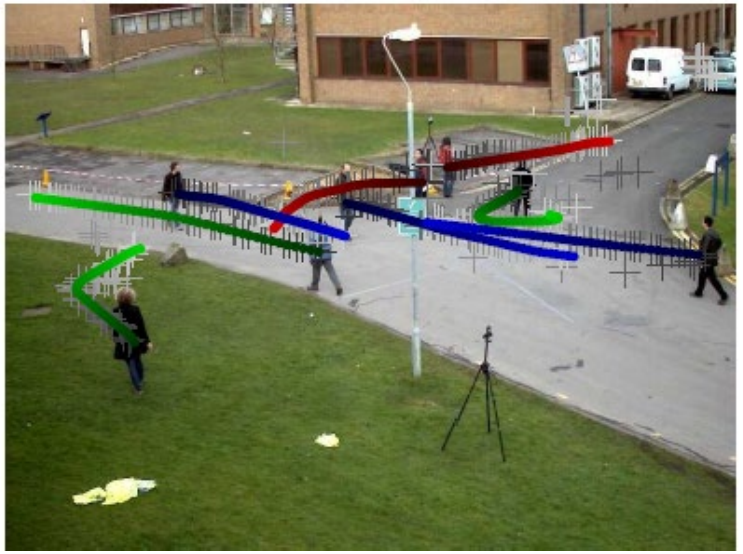
Human Pose Tracking



Stock Market Prediction

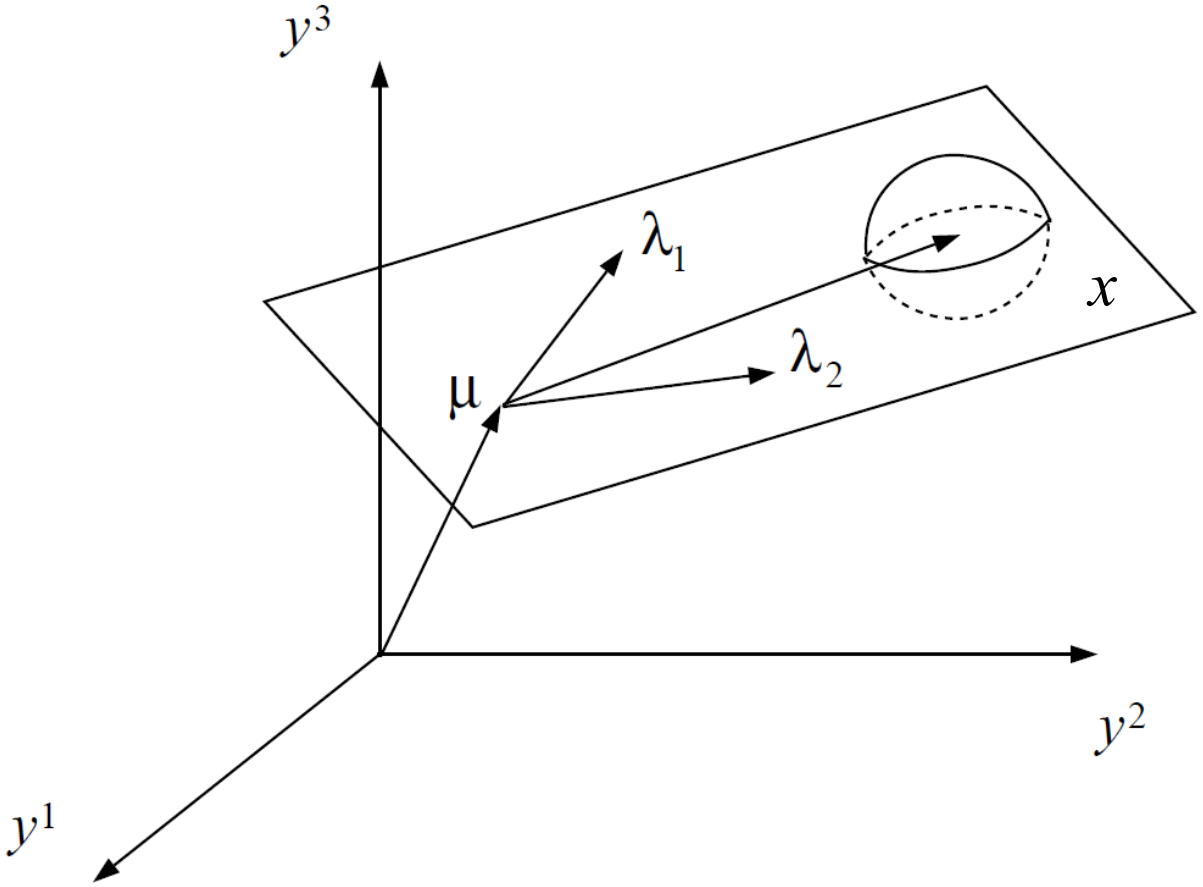


Visual Object Tracking



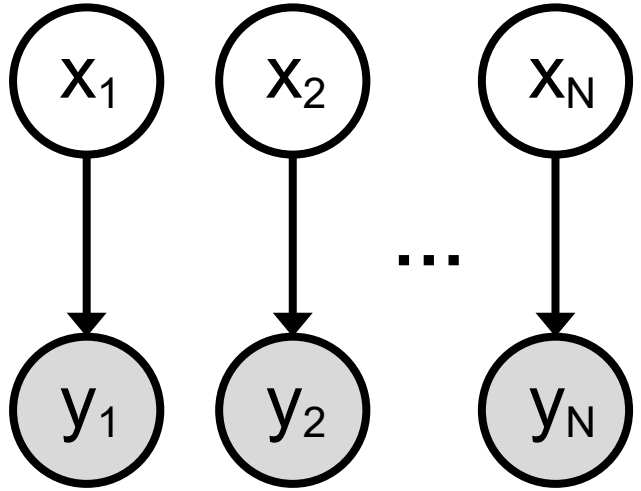
Move away from discrete HMM states to continuous ones…

Probabilistic Principal Component Analysis (PPCA)



Latent: $x \in \mathbb{R}^p$ Data: $y \in \mathbb{R}^q$

Typically p<q for dimension reduction



$$x \sim \mathcal{N}(0, I)$$

$$y \mid x \sim \mathcal{N}(\Lambda x + \mu, \sigma^2 I)$$

Data are exchangeable linear Gaussian projections of latent quantities

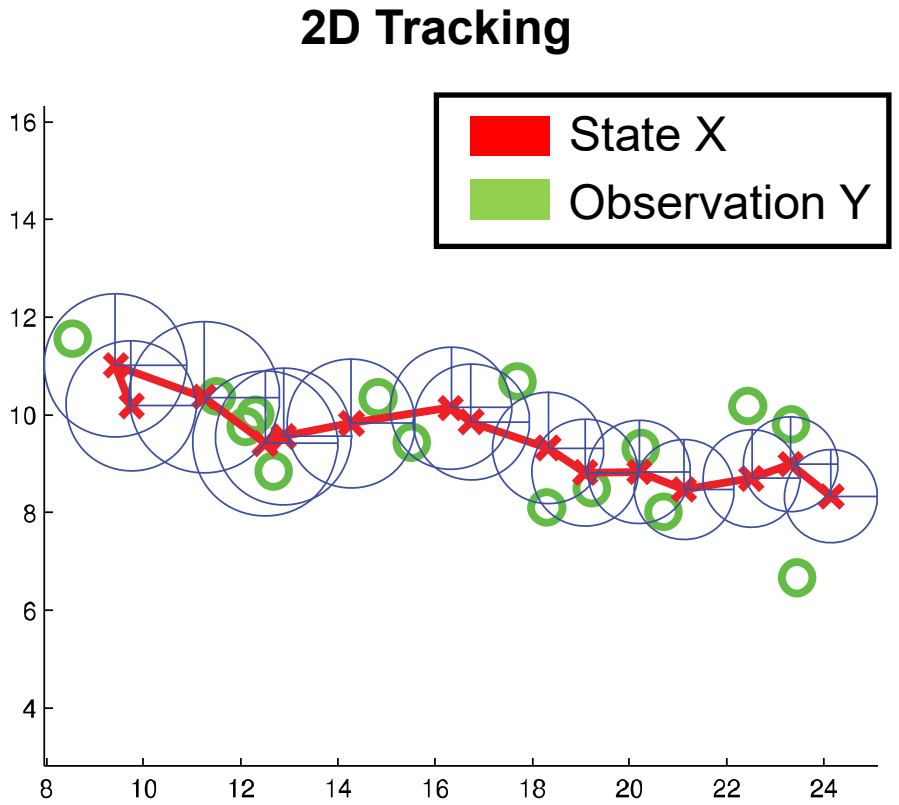
[Source: M. I. Jordan]

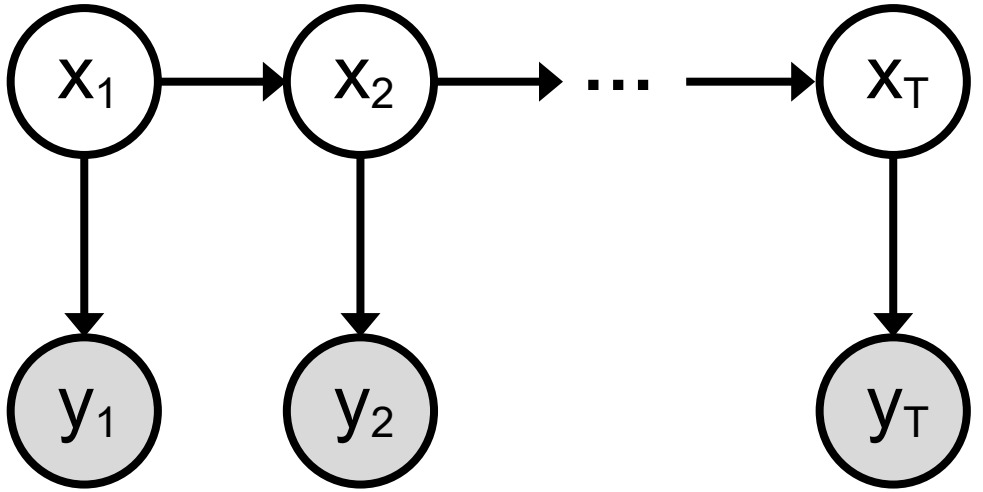
Administrivia

- **VOTE TOMORROW** If you haven't already…
- HW3 Grades
- HW4 Will be out Wednesday

Gaussian Linear Dynamical System (LDS)

Temporal extension of probabilistic PCA…





$$x_0 \sim \mathcal{N}(0, I)$$

$$x_t \mid x_{t-1} \sim \mathcal{N}(Fx_{t-1}, \Sigma)$$

$$y_t \mid x_t \sim \mathcal{N}(Hx_t, R)$$

Data are time-dependent and non-exchangeable

Linear State-Space Model

Consider the state vector:

$$x_t = \begin{pmatrix} x(t) \\ \dot{x}(t) \end{pmatrix} \quad \text{where} \quad x(t) : \text{Position} \quad \dot{x}(t) \triangleq \frac{d}{dt}x(t) : \text{Velocity}$$

Differential equations for constant velocity dynamics:

$$x(t) = x(t-1) + \dot{x}(t-1) \qquad \dot{x}(t) = \dot{x}(t-1)$$

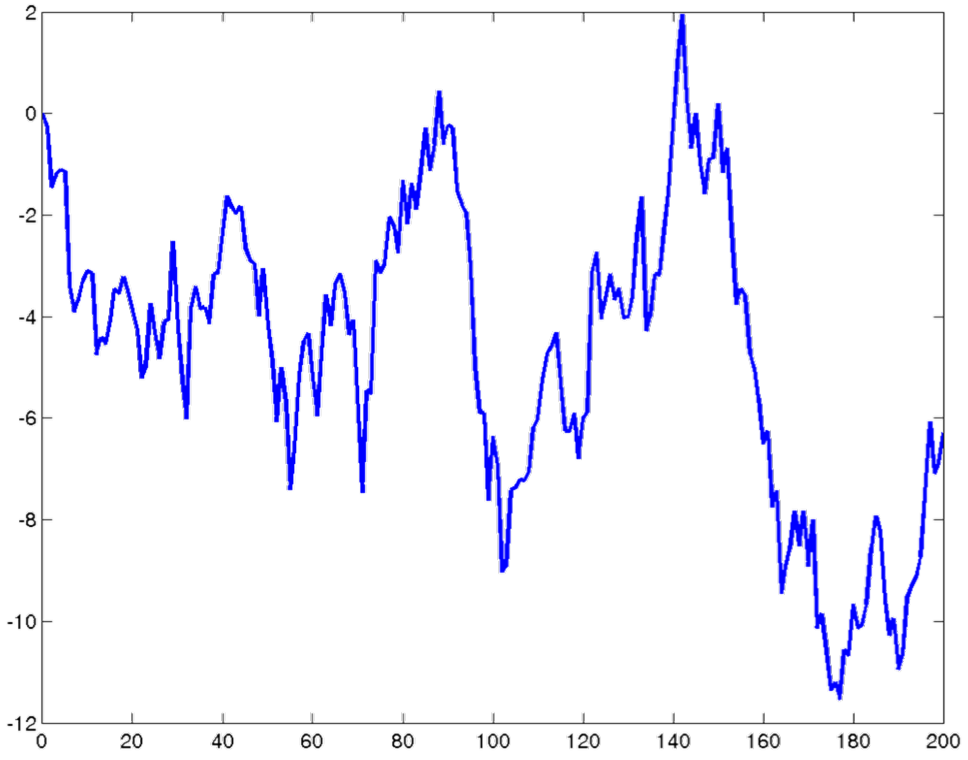
Linear Gaussian state-space model

$$\underbrace{x_t = Fx_{t-1} + \epsilon}_{\text{State-space notation}} \quad \text{where} \quad F = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \text{ and } \epsilon \sim \mathcal{N}(0, \Sigma)$$

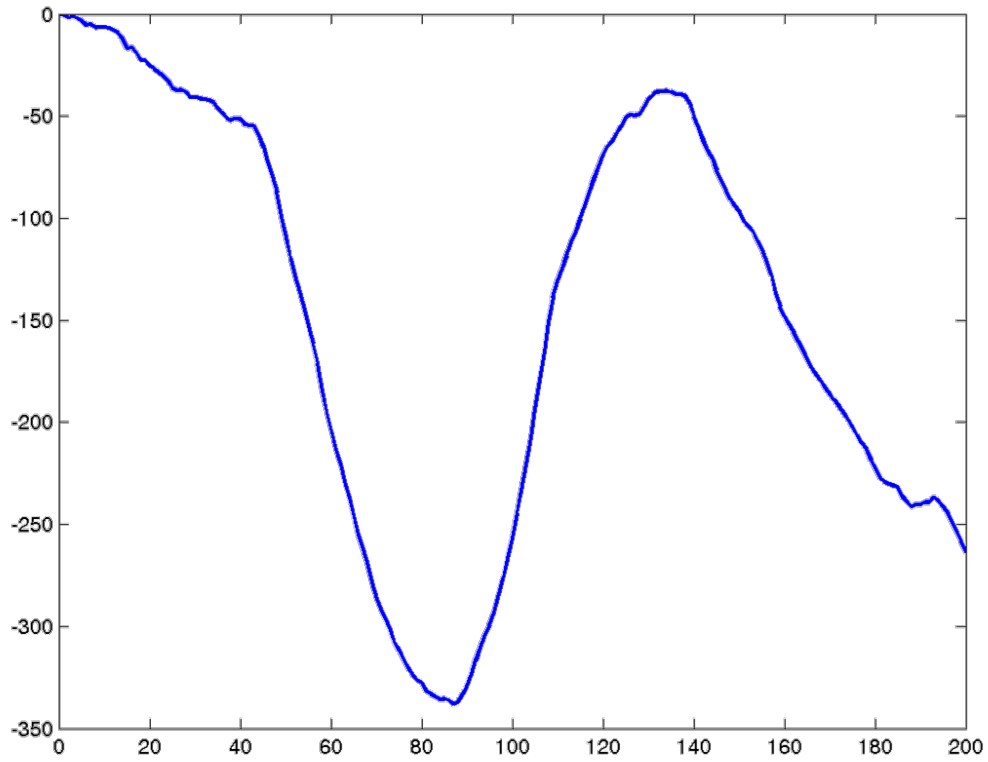
State-space notation
Linear regression model

Simple Linear Gaussian Dynamics

Random Walk
(Brownian Motion)



Constant Velocity
(a.k.a. zero acceleration)

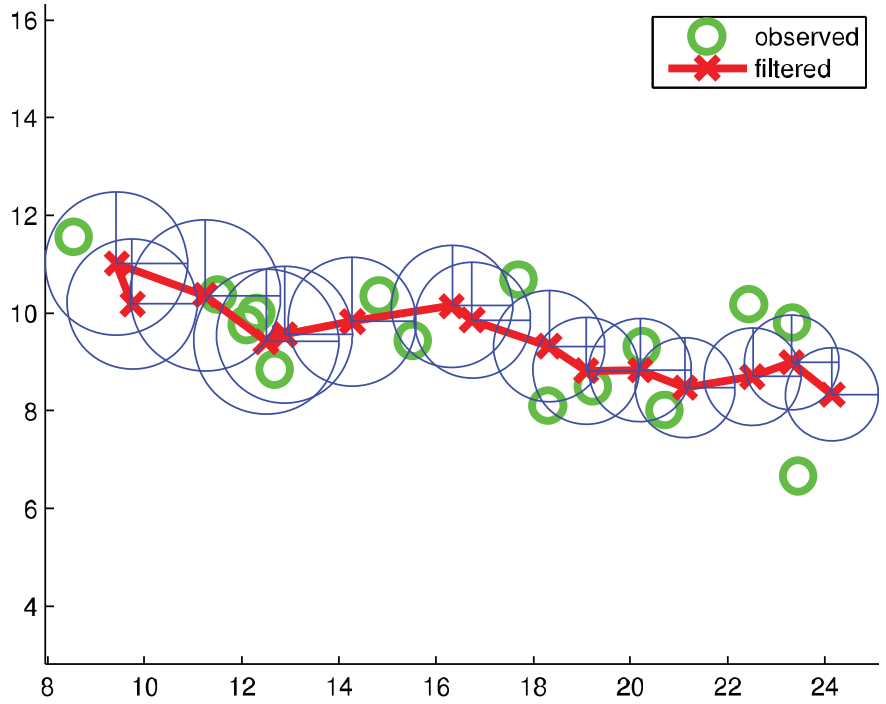


Acceleration can be included in higher-order models as well

Dynamical System Inference

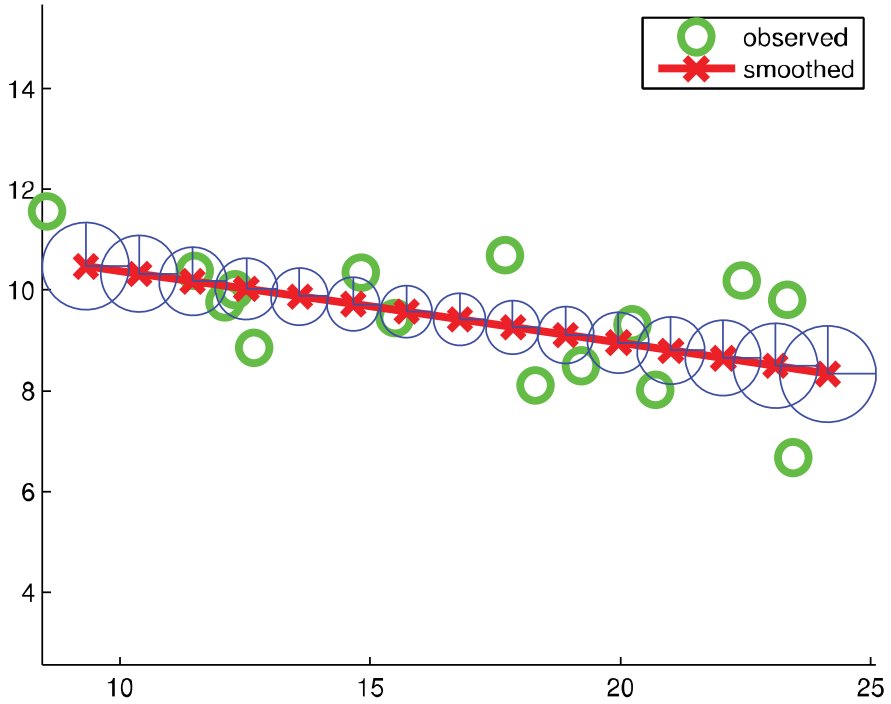
Define shorthand notation: $y_1^{t-1} \triangleq \{y_1, \ldots, y_{t-1}\}$

Filtering



Compute $p(x_t \mid y_1^t)$ at each time t

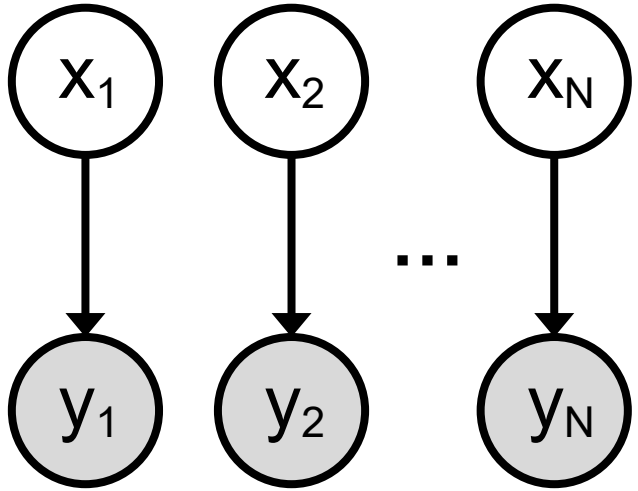
Smoothing



Compute full posterior marginal $p(x_t \mid y_1^T)$ at each time t

Linear Gaussian Inference

Generative Linear Regression



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

$$y \mid x \sim \mathcal{N}(Ax + b, R)$$

Marginal likelihood is Gaussian:

$$p(y) = \mathcal{N}(A\mu + b, R + A\Sigma A^T)$$

Posterior also Gaussian (surprise):

$$p(x \mid y) = \mathcal{N}(\mu_{x|y}, \Sigma_{x|y})$$

where,

$$\Sigma_{x|y}^{-1} = \Sigma^{-1} + A^T R^{-1} A$$

$$\mu_{x|y} = \Sigma_{x|y}\left[A^T R^{-1}(y - b) + \Sigma^{-1}\mu\right]$$

Building block for inference on linear Gaussian dynamical system

Gaussian LDS Filtering

- Suppose we have a Gaussian posterior at time t-1:

$$p(x_{t-1} \mid y_1^{t-1}) = \mathcal{N}(\mu_{t-1}, \Sigma_{t-1}) \qquad \text{where} \qquad y_1^{t-1} \triangleq \{y_1, \ldots, y_{t-1}\}$$

- Forward prediction at time t:

$$\begin{aligned} p(x_t \mid y_1^{t-1}) &= \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_1^{t-1})\, dx_{t-1} \\ &= \int \mathcal{N}(x_t \mid F x_{t-1}, \Sigma) \mathcal{N}(x_{t-1} \mid \mu_{t-1}, \Sigma_{t-1})\, dx_{t-1} \\ &= \mathcal{N}(x_t \mid F\mu_{t-1}, \Sigma + F\Sigma_{t-1}F^T) \boxed{\int \mathcal{N}(x_{t-1} \mid \cdot, \cdot)\, dx_{t-1}} \end{aligned}$$

Integrates to 1

Same form as marginal likelihood on previous slide

Gaussian LDS Filtering

- Forward prediction at time t: $p(x_t \mid y_1^{t-1}) = \mathcal{N}(x_t \mid \mu_{t|t-1}, \Sigma_{t|t-1})$

  where $\mu_{t|t-1} \triangleq F\mu_{t-1}$ and $\Sigma_{t|t-1} = \Sigma + F\Sigma_{t-1}F^T$

  State Prediction **Predicted Covariance**

- Posterior at time t is also Gaussian:

$$p(x_t \mid y_1^t) \propto p(x_t \mid y_1^{t-1})p(y_t \mid x_t)$$

$$= \mathcal{N}(x_t \mid \mu_{t|t-1}, \Sigma_{t|t-1})\mathcal{N}(y_t \mid Hx_t, R) \propto \mathcal{N}(x_t \mid \mu_{t|t}, \Sigma_{t|t})$$

Gain Matrix: $K_t = \Sigma_{t|t-1}H^T(H\Sigma_{t|t-1}H^T + R)^{-1}$

Filter Covariance: $\Sigma_{t|t} = \Sigma_{t|t-1} - K_tH\Sigma_{t|t-1}$

Filter Mean: $\mu_{t|t} = \mu_{t|t-1} + K_t(y_t - H\mu_{t|t-1})$

Can be derived from Gaussian conditional formulas and Woodbury matrix identity

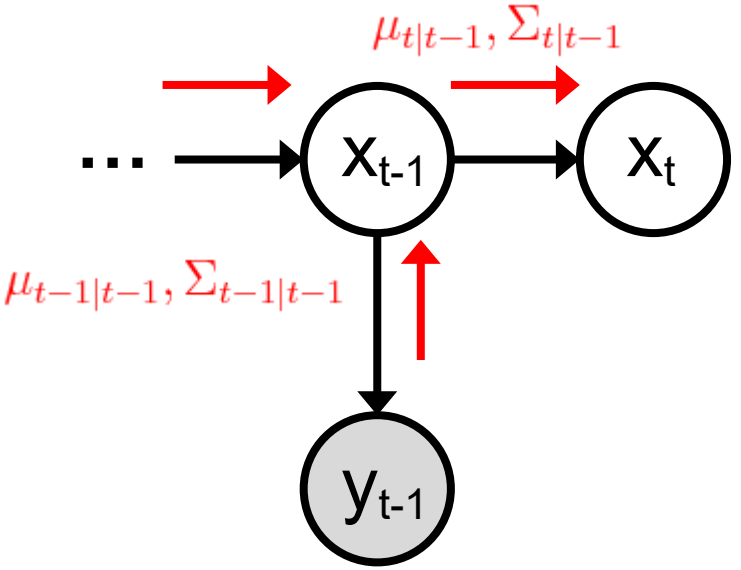
Kalman Filter

Prediction Step:

State Prediction: $\mu_{t|t-1} = F\mu_{t-1|t-1}$

Covariance Prediction:

$$\Sigma_{t|t-1} = \Sigma + F\Sigma_{t-1|t-1}F^T$$

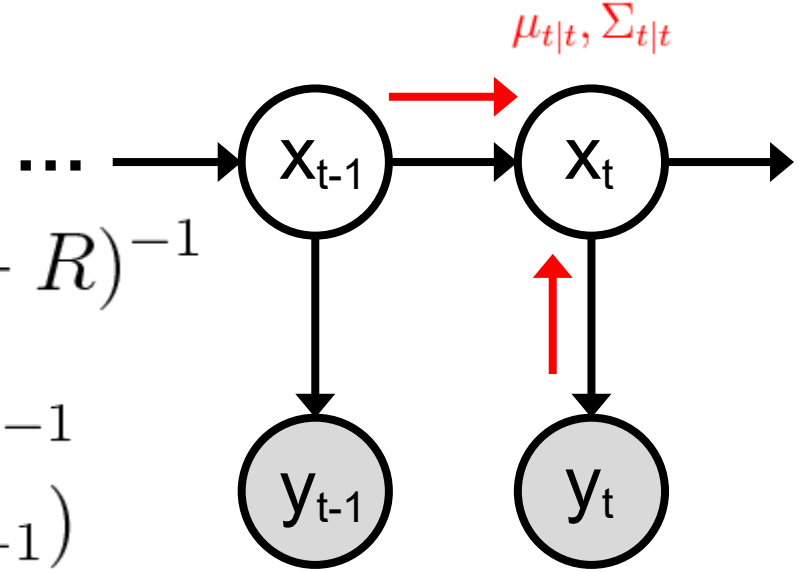


Measurement Update Step:

Gain Matrix: $K_t = \Sigma_{t|t-1}H^T(H\Sigma_{t|t-1}H^T + R)^{-1}$

Filter Covariance: $\Sigma_{t|t} = \Sigma_{t|t-1} - K_tH\Sigma_{t|t-1}$

Filter Mean: $\mu_{t|t} = \mu_{t|t-1} + K_t(y_t - H\mu_{t|t-1})$



Kalman Filter

Prediction Step:

State Prediction: $\mu_{t|t-1} = F\mu_{t-1|t-1}$

Covariance Prediction:

$\Sigma_{t|t-1}$ = $\Sigma_{t-1|t-1} F^T$

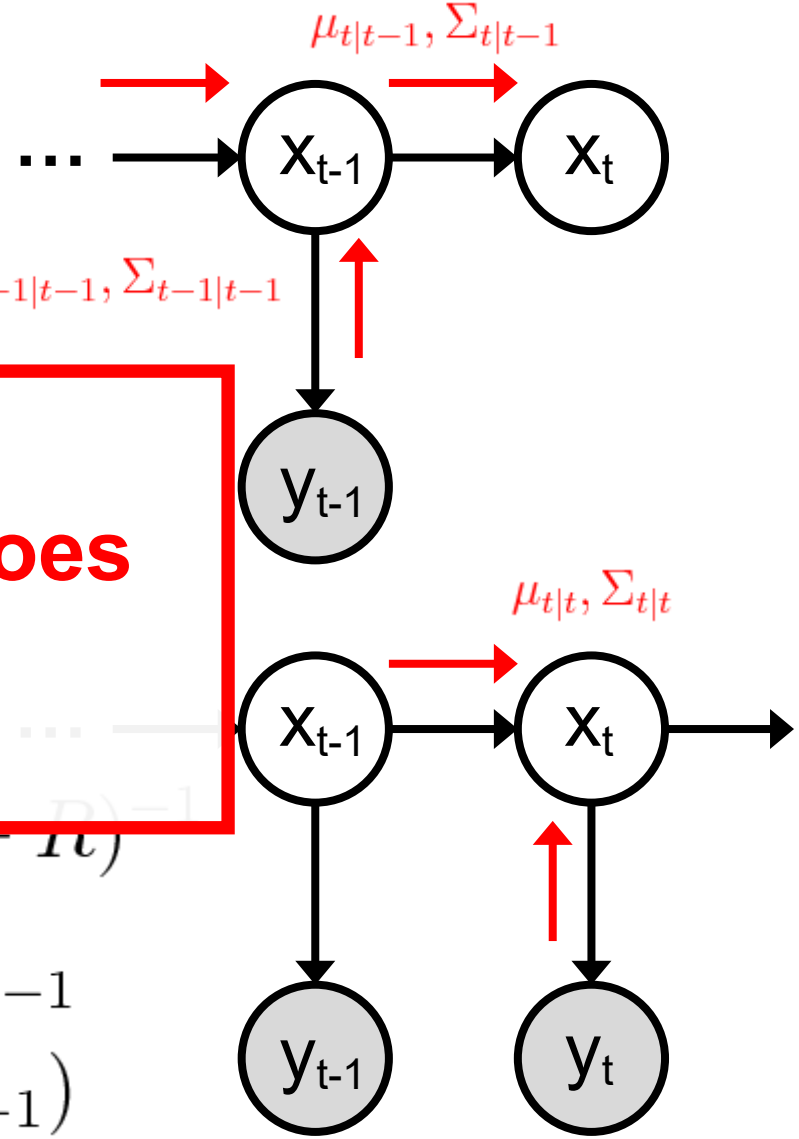
What algorithm does this look like?

Measurement Update Step:

Gain Matrix: $K_t = \Sigma_{t|t-1} H^T (H\Sigma_{t|t-1}H^T + R)^{-1}$

Filter Covariance: $\Sigma_{t|t} = \Sigma_{t|t-1} - K_t H \Sigma_{t|t-1}$

Filter Mean: $\mu_{t|t} = \mu_{t|t-1} + K_t(y_t - H\mu_{t|t-1})$



Kalman Filter

Prediction Step:

State Prediction: $\mu_{t|t-1} = F\mu_{t-1|t-1}$

Covariance Predic

$\Sigma_{t|t-1}$

What algorithm does this look like?

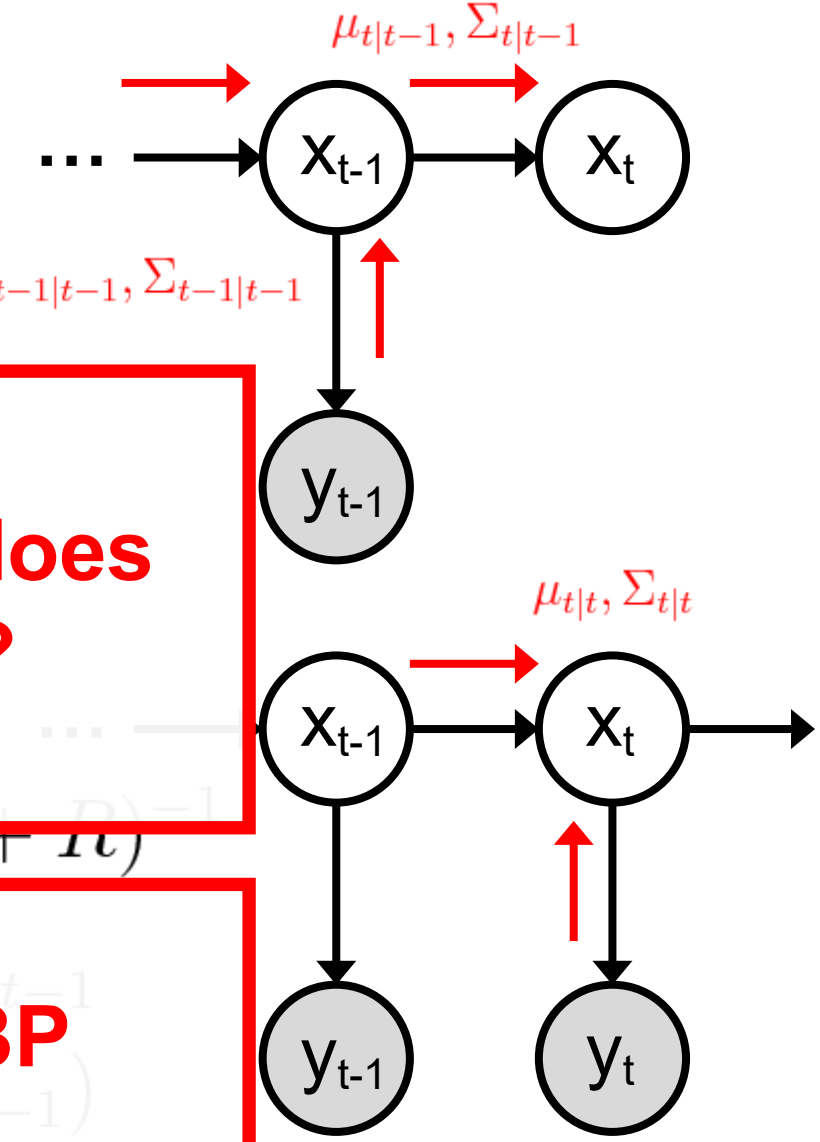
Measurement Upd

Gain Matrix: $K_t =$

Filter Covariance:

Filter Mean: $\mu_{t|t} =$

Sum-Product BP



Gaussian Parameterization

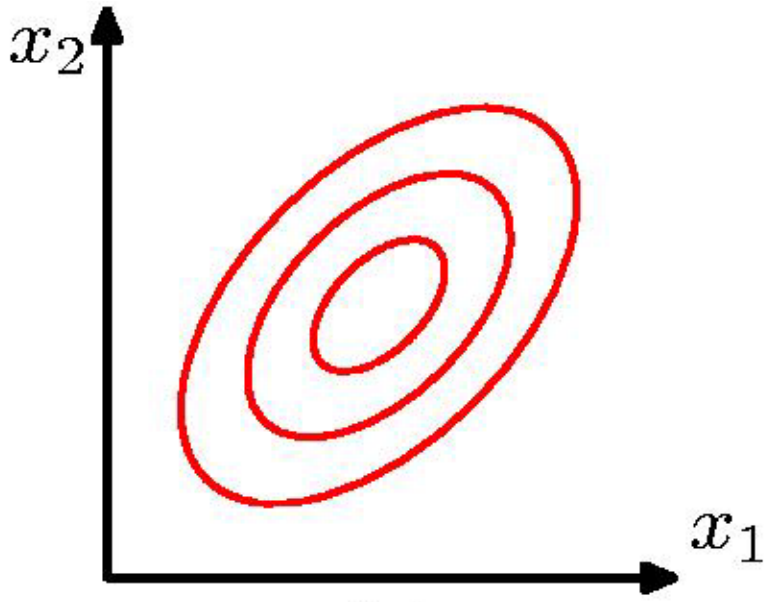
Mean Parameterization:

$$\mathcal{N}(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{N/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\}$$

Canonical Parameterization

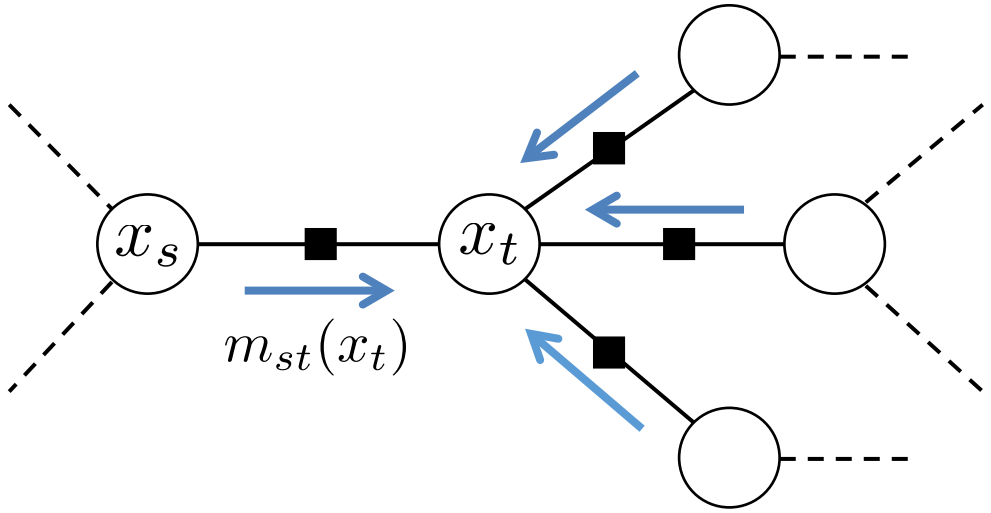
$$\mathcal{N}^{-1}(x \mid \vartheta, \Lambda) \propto \exp\left\{-\frac{1}{2}x^T \Lambda x + \vartheta^T x\right\}$$

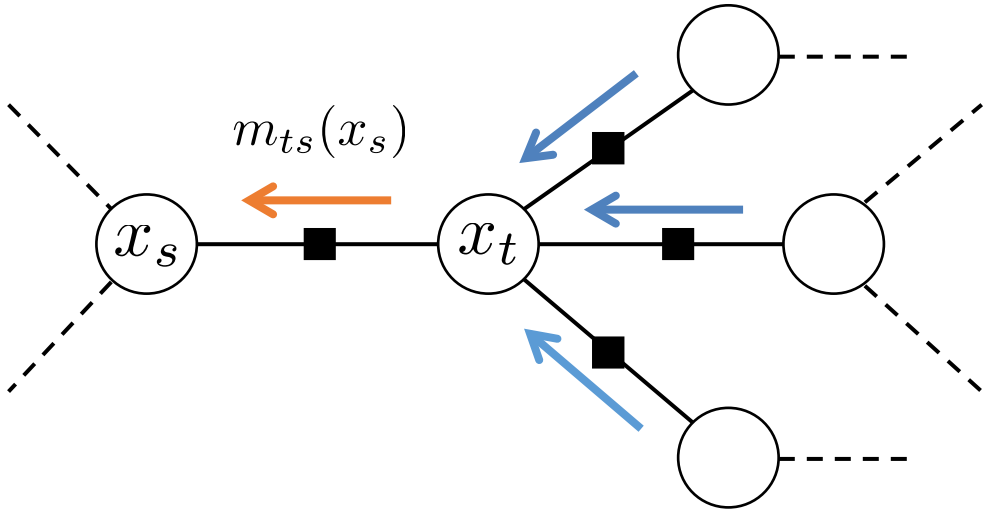
where $\Lambda = \Sigma^{-1} \quad \vartheta = \Sigma^{-1}\mu$



Also called natural parameters and information parameters in some texts…

Gaussian Belief Propagation





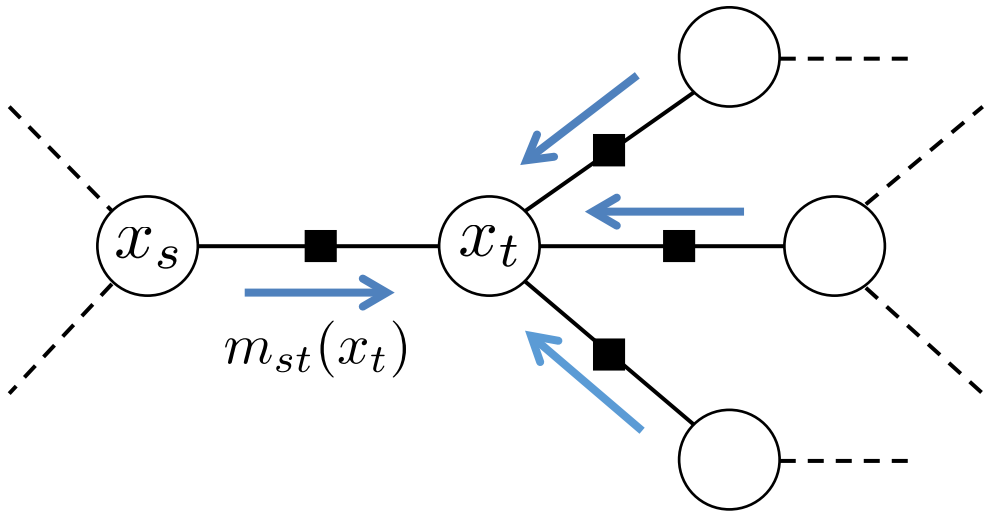
$$p_t(x_t) \propto \prod_{s \in \Gamma(t)} m_{st}(x_t) \qquad m_{ts}(x_s) = \int_{\mathcal{X}_t} \psi_{st}(x_s, x_t) \prod_{u \in \Gamma(t) \setminus s} m_{ut}(x_t)\, dx_t$$

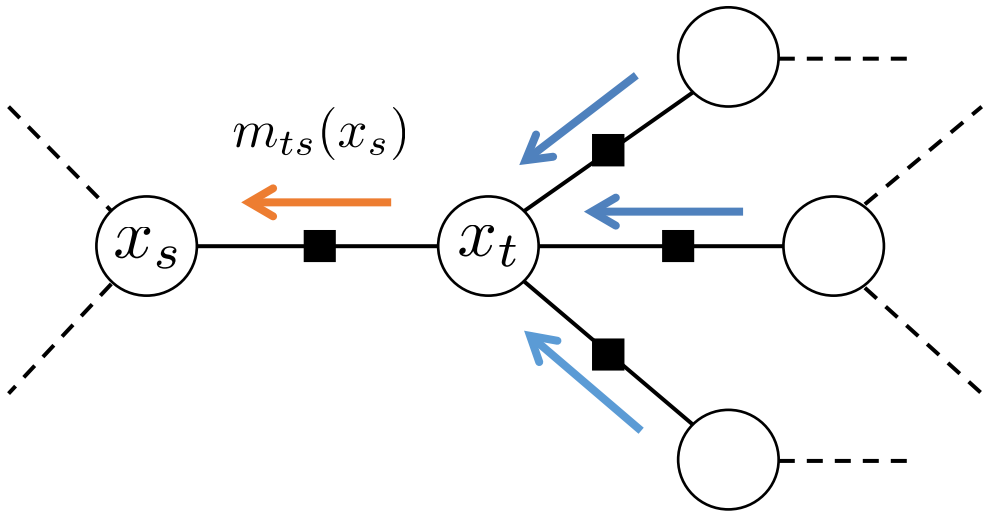
$$p_t(x_t) = \mathcal{N}^{-1}(x_t \mid \vartheta_t, \Lambda_t) \qquad m_{ts}(x_s) \propto \mathcal{N}^{-1}(x_s \mid \vartheta_{ts}, \Lambda_{ts})$$

Computing *marginal mean and covariance* from messages:

$$\Lambda_t = \sum\nolimits_{s \in \Gamma(t)} \Lambda_{st} \qquad \vartheta_t = \sum\nolimits_{s \in \Gamma(t)} \vartheta_{st}$$

Gaussian Belief Propagation





- Compute message mean & covar as algebraic function of incoming message mean & covar (generalizes Kalman)
- For tree of *N* nodes of dimension *d*, cost is $O(Nd^3)$

Computing *marginal mean and covariance* from messages:

$$\Lambda_t = \sum_{s \in \Gamma(t)} \Lambda_{st} \qquad \vartheta_t = \sum_{s \in \Gamma(t)} \vartheta_{st}$$

LDS Summary

Linear dynamical system,

$$x_t \mid x_{t-1} \sim \mathcal{N}(Fx_{t-1}, \Sigma) \qquad y_t \mid x_t \sim \mathcal{N}(Hx_t, R)$$

Linear Gaussian Dynamics **Linear Gaussian Observation**

State-space representation same as linear regression,

$$x_t = Fx_{t-1} + \epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \Sigma)$$

Exact posterior inference via Kalman filter

- Recursively pass marginal moments
- Two steps: prediction, measurement update
- Forward pass filtering, backward pass smoothing

Kalman is special case of Gaussian sum-product BP

Outline

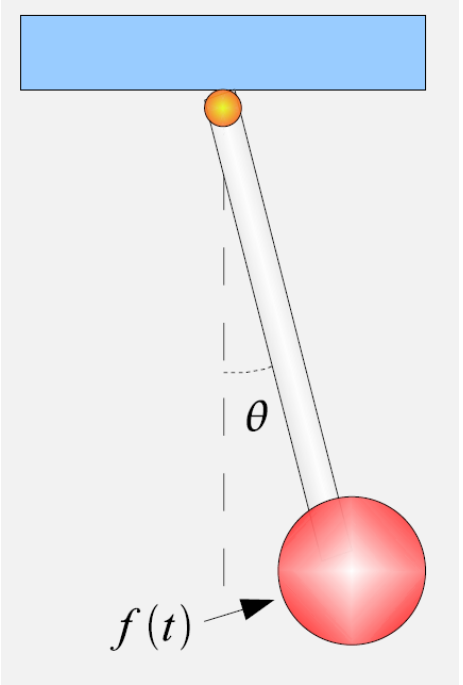
- Sequence Models
- Linear Dynamical Systems
- LDS Extensions

Nonlinear Dynamical System

Pendulum with mass m=1,pole length L=1:

$$x_t = \begin{pmatrix} \theta_t \\ \dot{\theta}_t \end{pmatrix} = \underbrace{\begin{pmatrix} \theta_{t-1} + \dot{\theta}_{t-1} \\ \dot{\theta}_{t-1} - g\sin(\theta_{t-1}) \end{pmatrix}}_{f(x_{t-1})} + \epsilon$$

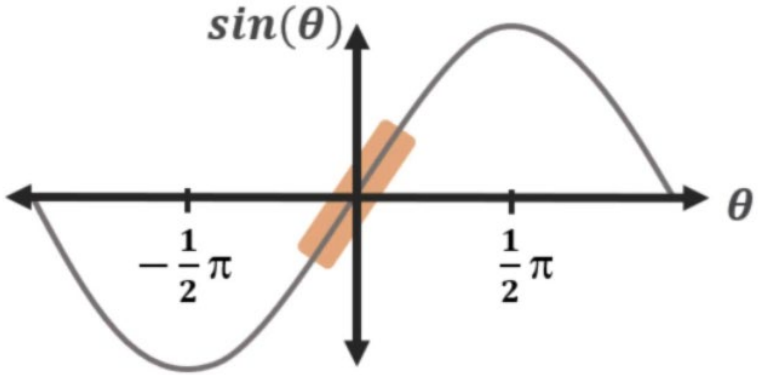
$$y_t = \underbrace{\sin(\theta_t)}_{h(x_t)} + \omega$$



Nonlinear dynamics / measurement:

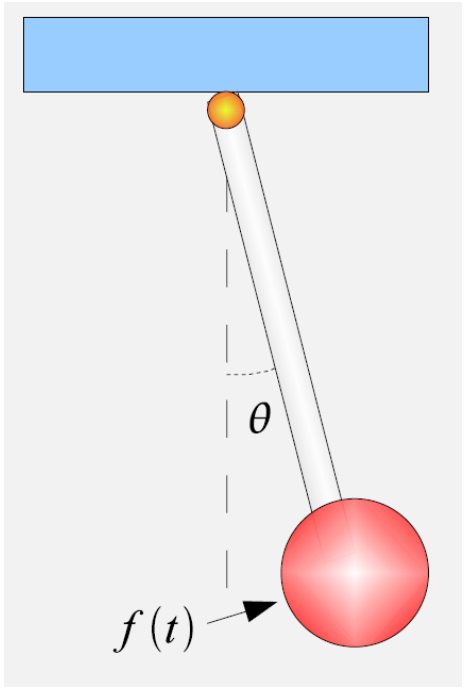
$$x_t = f(x_{t-1}) + \epsilon \sim \mathcal{N}(0, \Sigma)$$

$$y_t = h(x_t) + \omega \sim \mathcal{N}(0, R)$$



Nonlinear Dynamical System

Filter equations lack a closed-form:



Prediction:

$$p(x_t \mid y_1^{t-1}) = \int \mathcal{N}(x_t \mid f(x_{t-1}), \Sigma) p(x_{t-1} \mid y_1^{t-1})\, dx_{t-1}$$

Measurement Update:

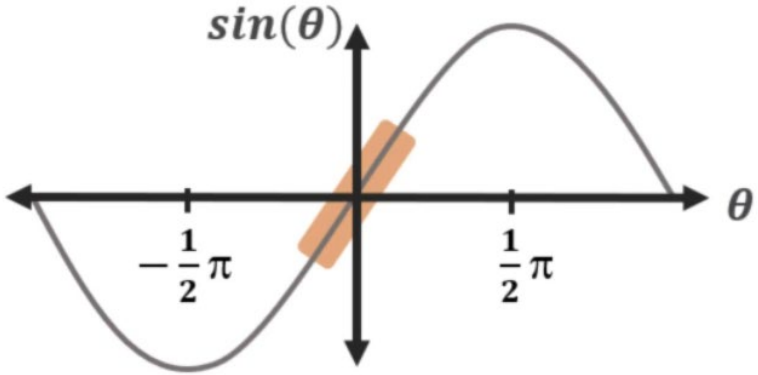
$$p(x_t \mid y_1^t) \propto \mathcal{N}(y_t \mid h(x_t), R) p(x_t \mid y_1^{t-1})$$

Idea *Linearize* f(.) and h(.) about a point m:

$$f(x) \approx f(m) + \mathbf{J}_f(m)(x - m)$$

$$h(x) \approx h(m) + \mathbf{J}_h(m)(x - m)$$

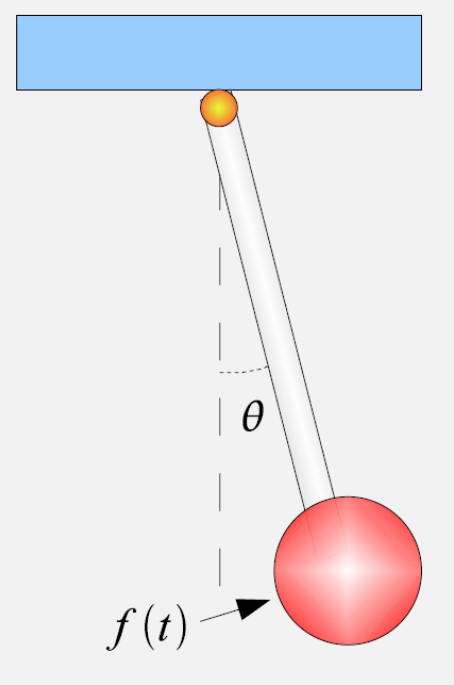
1st Order Taylor expansion



where $\mathbf{J}_f = \left(\frac{\partial f_i}{\partial x_j}\right)$ is Jacobian matrix of partials

Extended Kalman Filter

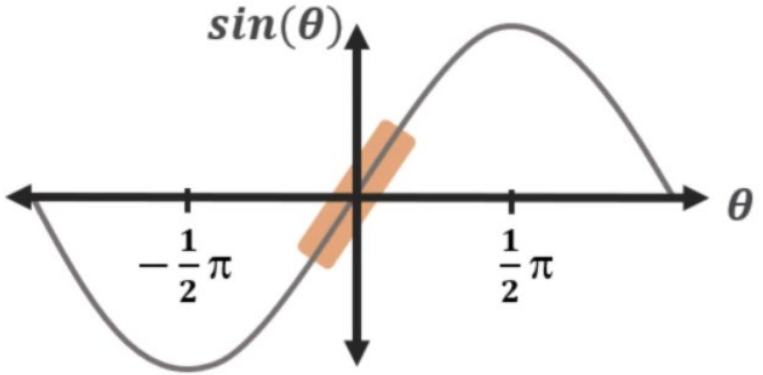
1. Linearize f(.) and h(.) about filter mean
2. Assume linear Gaussian model
3. Do standard Kalman updates



Example Linearization of pendulum model

$$x_t = \begin{pmatrix} \theta_t \\ \dot{\theta}_t \end{pmatrix} = \underbrace{\begin{pmatrix} \theta_{t-1} + \dot{\theta}_{t-1} \\ \dot{\theta}_{t-1} - g\sin(\theta_{t-1}) \end{pmatrix}}_{f(x_{t-1})} + \epsilon \qquad y_t = \underbrace{\sin(\theta_t)}_{h(x_t)} + \omega$$

$$\mathbf{J}_f(x) = \begin{pmatrix} 1 & 1 \\ -g\cos(x^1) & 1 \end{pmatrix} \qquad \mathbf{J}_h(x) = (\cos(x^1), 0)$$



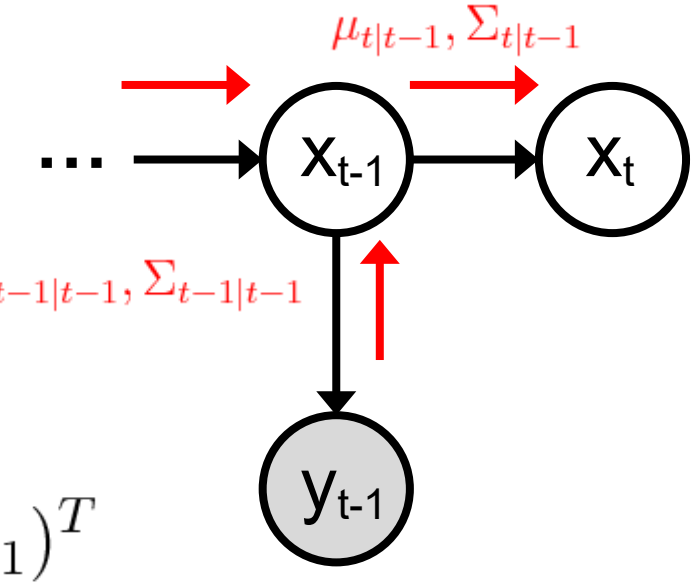
EKF Update Equations

Prediction Step:

State Prediction: $\mu_{t|t-1} = f(\mu_{t-1|t-1})$

Covariance Prediction:

$$\Sigma_{t|t-1} = \Sigma + \mathbf{J}_f(\mu_{t-1|t-1})\Sigma_{t-1|t-1}\mathbf{J}_f(\mu_{t-1|t-1})^T$$

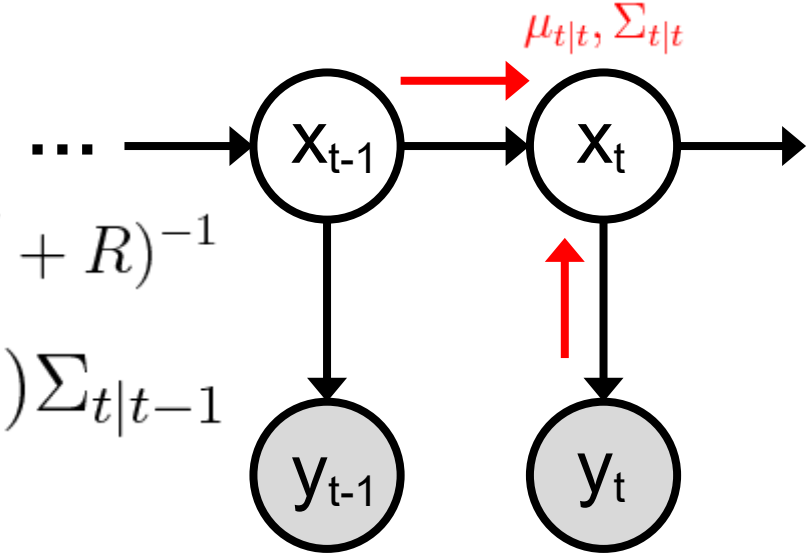


Measurement Update Step:

Gain: $K_t = \Sigma_{t|t-1}\mathbf{J}_h(\mu_{t|t-1})^T(\mathbf{J}_h(\mu_{t|t-1})\Sigma_{t|t-1}\mathbf{J}_h(\mu_{t|t-1})^T + R)^{-1}$

Filter Covariance: $\Sigma_{t|t} = \Sigma_{t|t-1} - K_t\mathbf{J}_h(\mu_{t|t-1})\Sigma_{t|t-1}$

Filter Mean: $\mu_{t|t} = \mu_{t|t-1} + K_t(y_t - h(\mu_{t|t-1}))$

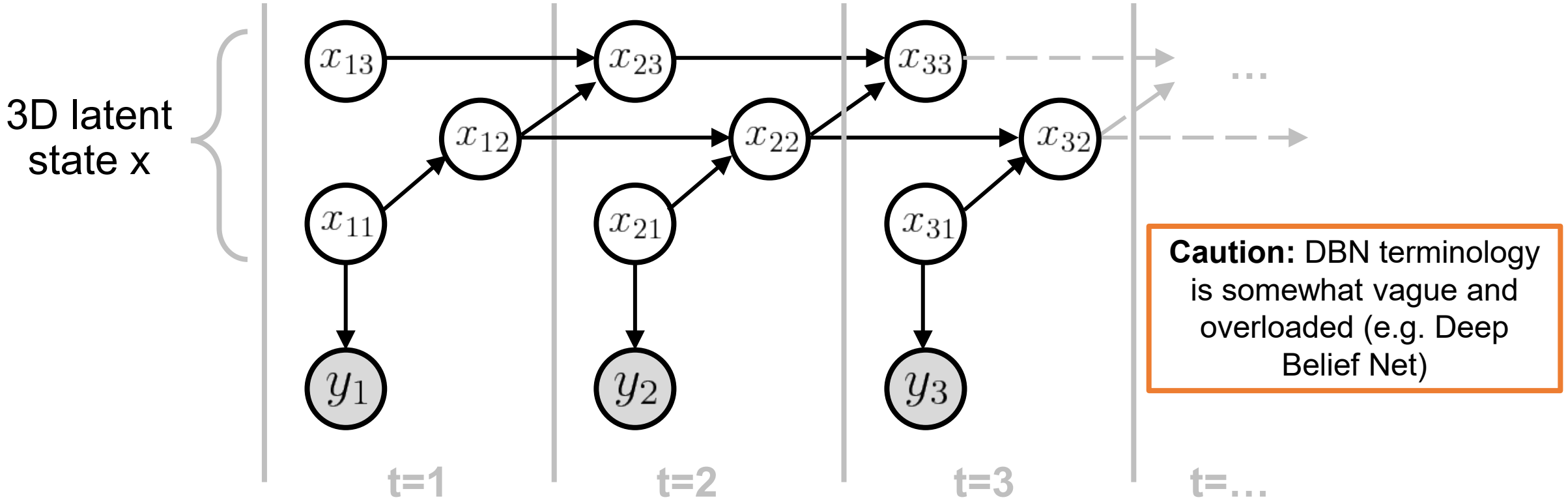


Extended Kalman Filter

- **PROS:**
  - Easy to implement – updates analogous to standard Kalman
  - Computationally efficient
  - Known theoretical stability results
- **CONS:**
  - Linearity assumption poor for highly nonlinear models
  - Requires model differentiability
  - Jacobian matrices can be hard to calculate & implement

Unscented Kalman filter (UKF) typically more accurate in practice

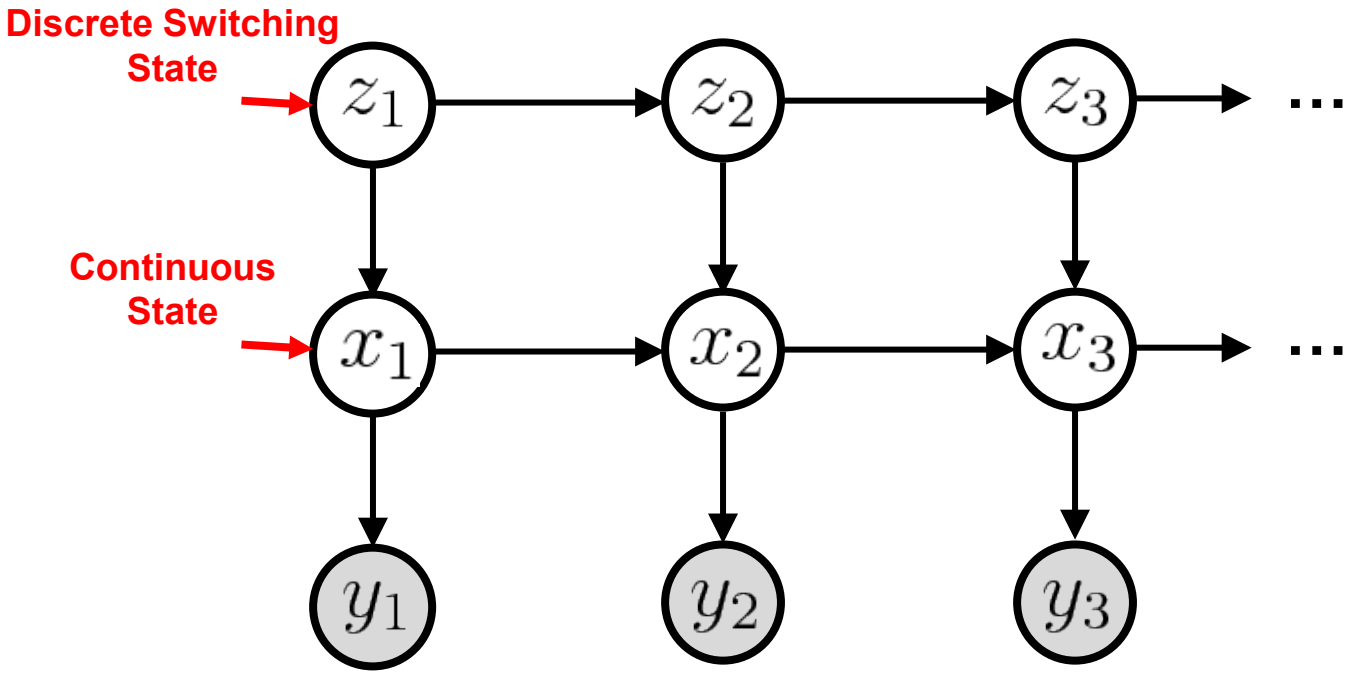
Dynamic Bayesian Networks

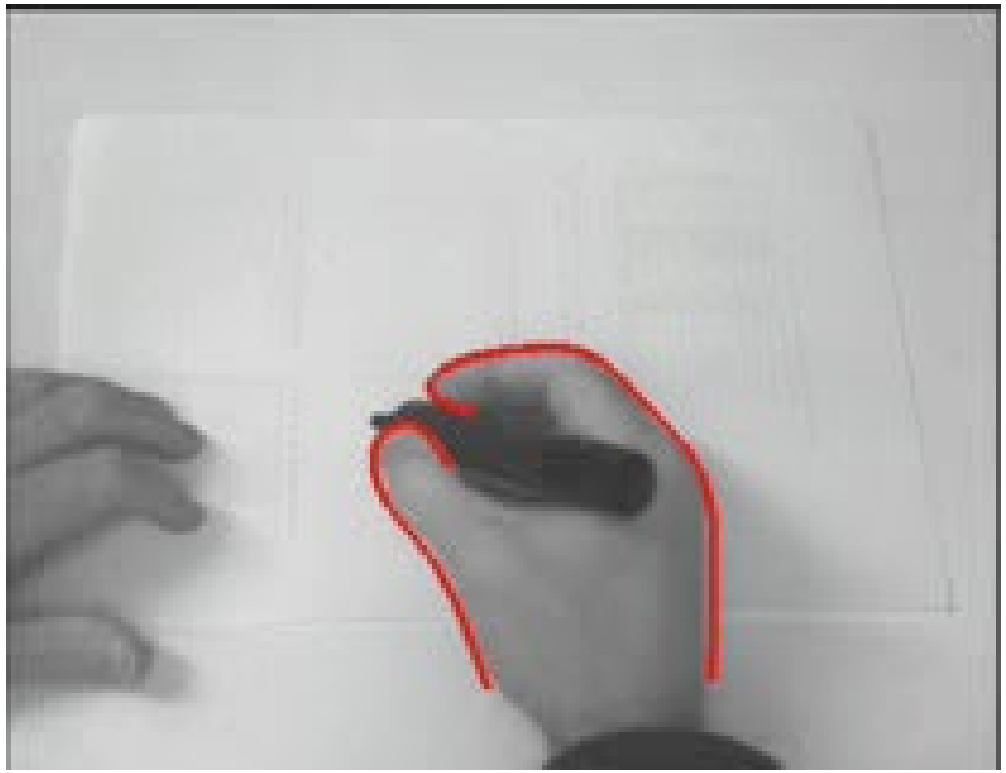


Caution: DBN terminology is somewhat vague and overloaded (e.g. Deep Belief Net)

- Multivariate latent state (e.g. $x \in \mathbb{R}^3$)
- Dynamics for each component within and across time
- Sometimes used as catch-all term for dynamical systems

Switching Linear Dynamical System





Colors indicate 3 writing modes

[Video: Isard & Blake, ICCV 1998.]

Discrete switching state:

$$z_t \mid z_{t-1} \sim \mathrm{Cat}(\pi(z_{t-1}))$$

With stochastic transition matrix π

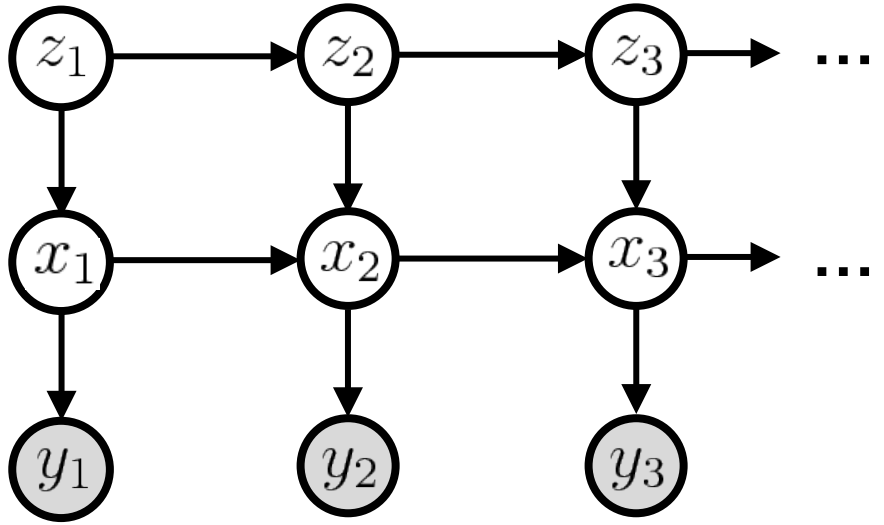
Switching state selects linear dynamics:

$$x_t \mid x_{t-1} \sim \mathcal{N}(A_{z_t} x_{t-1}, \Sigma_{z_t})$$

(e.g. Linear Gaussian)

Switching Linear Dynamical System

We can do sum-product for HMM and LDS, so maybe we can do it for SLDS…



Forward message,

$$m_{t-1,t}(z_t, x_t) \propto \int \sum_{z_{t-1}} m_{t-2,t-1}(z_{t-1}, x_{t-1}) p(y_{t-1} \mid x_{t-1})$$

$$p(z_t \mid z_{t-1}) p(x_t \mid x_{t-1}, z_t) dx_{t-1}$$

Integrates to Gaussian for each possible state z_{t-1}

$$= \int \sum_{z_{t-1}} m_{t-2,t-1}(z_{t-1}, x_{t-1}) \mathcal{N}(y_{t-1} \mid H x_{t-1}, R) \mathrm{Cat}(z_t \mid \pi(z_{t-1})) \mathcal{N}(x_t \mid A_{z_t} x_{t-1}, \Sigma_{z_t}) dx_{t-1}$$

- Message is Gaussian mixture over K states (for some K)
- But incoming message is also a Gaussian mixture
- Number of components (K) grows exponentially with time t

One way is to use sample-based methods (Particle Filter)

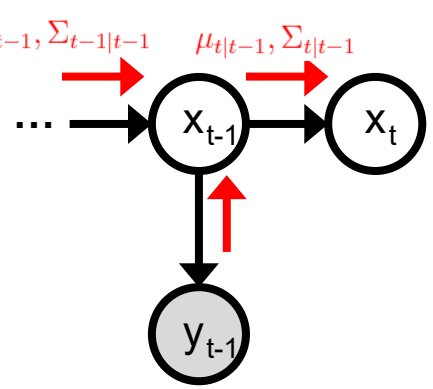
Summary

- Linear dynamical system is time-extension of PCA,

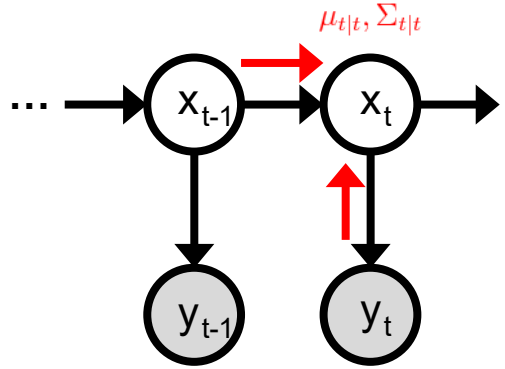
$$x_t \mid x_{t-1} \sim \mathcal{N}(Fx_{t-1}, \Sigma) \qquad y_t \mid x_t \sim \mathcal{N}(Hx_t, R)$$

- Exact inference via Kalman filtering,

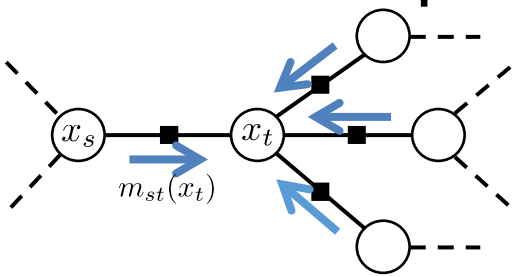
Prediction step updates moments of predictive distribution $p(x_t \mid y_1^{t-1})$

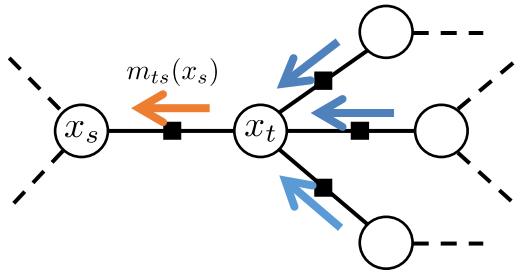


Measurement step updates filter distribution with newest measumrent $p(x_t \mid y_1^t)$



- Kalman filter is special case of Gaussian belief propagation





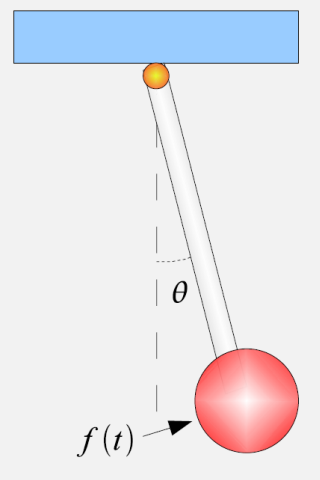
Messages represented as Gaussian **natural parameters**

Summary

- Nonlinear state-space models allow more complex dynamics,

$$x_t = \begin{pmatrix} \theta_t \\ \dot{\theta}_t \end{pmatrix} = \underbrace{\begin{pmatrix} \theta_{t-1} + \dot{\theta}_{t-1} \\ \dot{\theta}_{t-1} - g\sin(\theta_{t-1}) \end{pmatrix}}_{f(x_{t-1})} + \epsilon$$

$$y_t = \underbrace{\sin(\theta_t)}_{h(x_t)} + \omega$$



Approximate Kalman filter inference via linearization (Extended Kalman Filter) or Gaussian quadrature (Unscented Kalman Filter)

- Switching state-space model represents discrete & continuous states,

Exact inference intractable due to exponential growth in message parameters

Both nonlinear and SSMs can be addressed using sample-based methods (e.g. Particle Filtering) as we will see