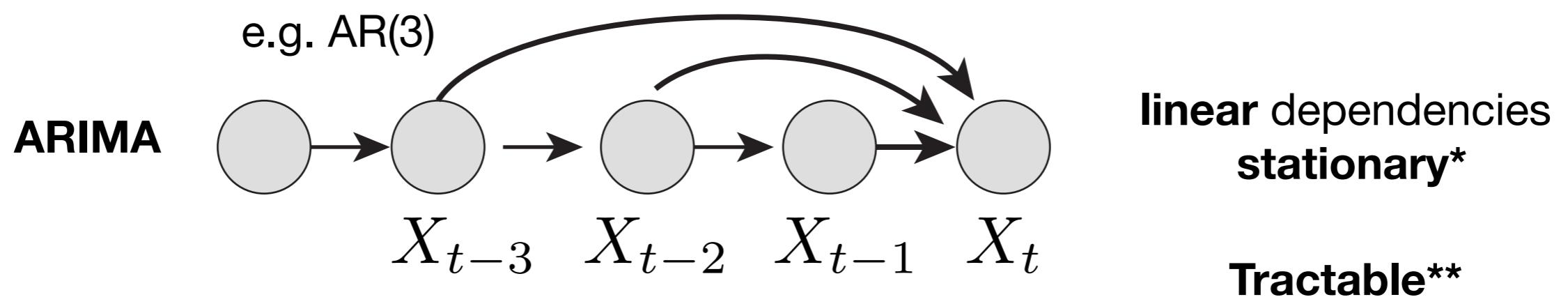


DS-GA 1018.001  
Probabilistic time series analysis  
Lecture 4  
Latent state model: Kalman filtering

Instructor: Cristina Savin  
NYU, CNS & CDS

So far : **directly** model the dependencies across time, under certain **restrictions**.  
**BUT** only for **scalar** variables really



The polynomial representation provides simple characterization  
of the properties of the corresponding stochastic process:

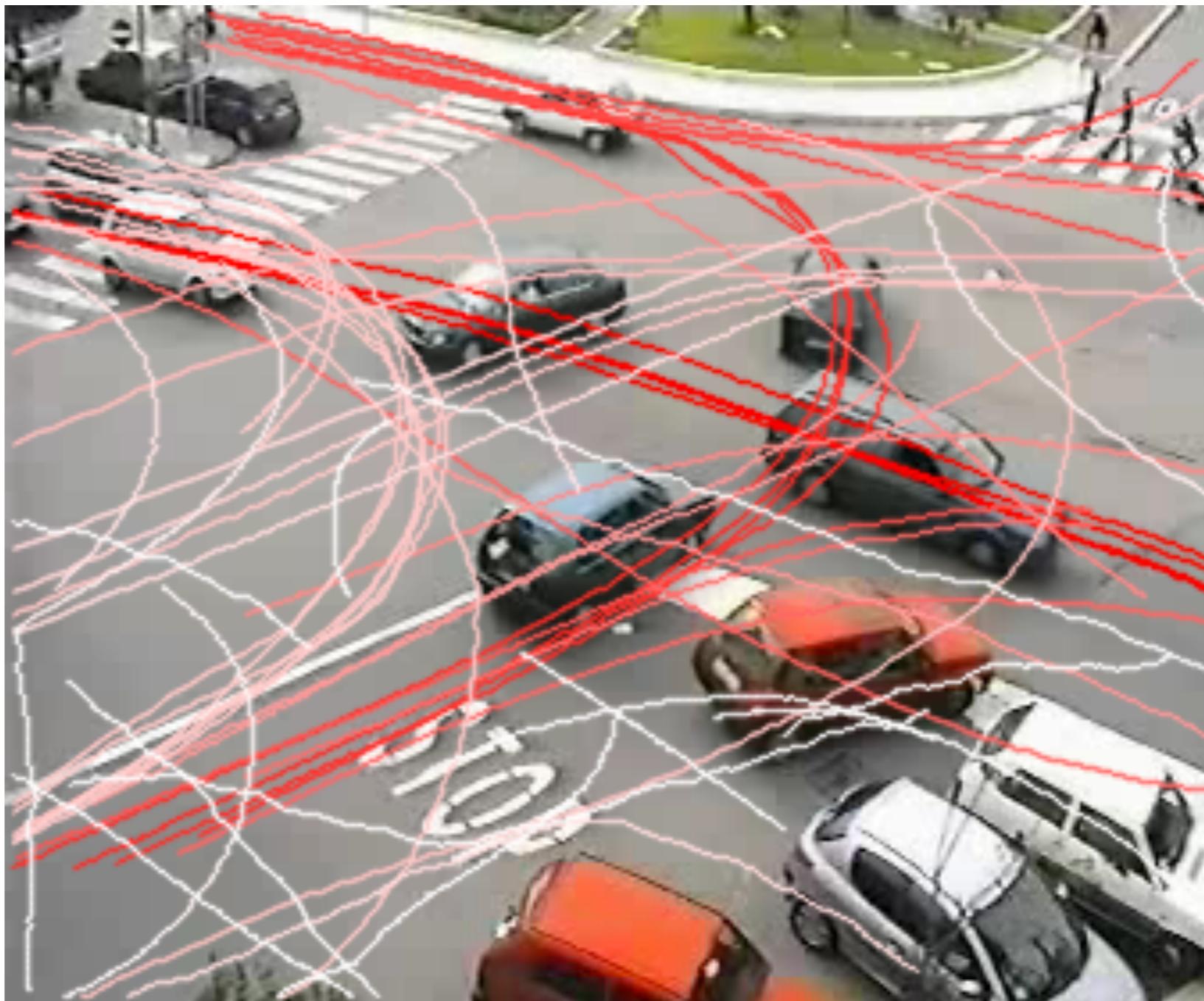
- The roots of the polynomial determine **causality** and **invertibility**, etc.
- They also provide a way to simplify complex models to **minimal form**.
- We can also use the roots to do **prediction**
- Polynomial form defines recursion expressions for the moments of the underlying distribution, which we can use to estimate model **parameters**.

\*Polynomial dependence of the mean, for ARIMA

# Building ARIMA models

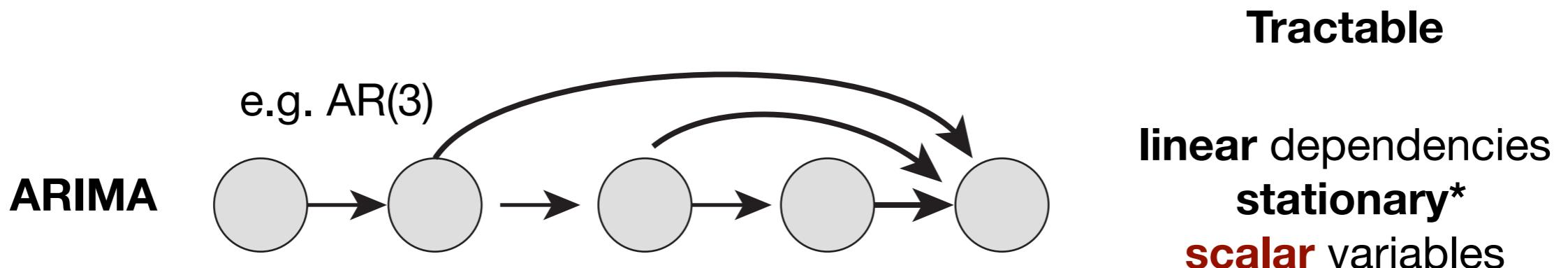
- plotting the data, check gaussianity
- possibly transforming the data,
- identifying the dependence **orders** of the model,
- parameter estimation,
- diagnostics
- model selection.

Most interesting data is complex and **multidimensional**



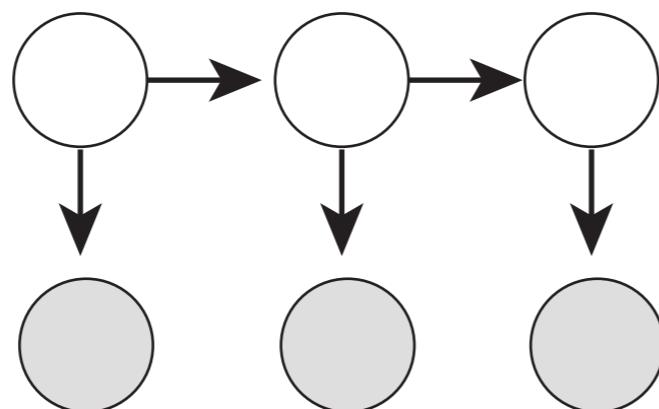
*From Saunier & Sayed 2008*

**Directly** model the dependencies across time, under certain **restrictions**.



In the latent space the temporal dependencies are simple:

**1<sup>st</sup> order Markov**

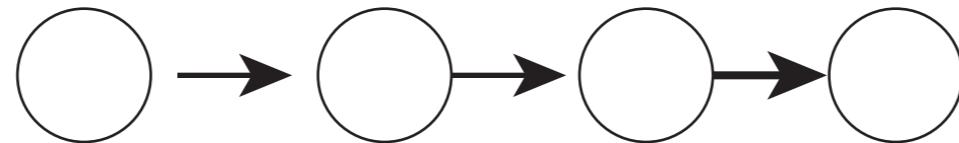


Goal: find a **latent** variable  
that summarizes  
the relevant **history**  
while keeping math simple

The latent variable is either discrete for **hidden Markov models** (HMMs) or  
continuous for **latent dynamical systems** (LDS)

## Markov property

e.g. first order Markov process



A sequence has the Markov property if, once we know the most recent value  $\mathbf{x}_{t-1}$  the current state  $\mathbf{x}_t$  is independent on the past history.

$$P(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \dots, \mathbf{x}_0) = P(\mathbf{x}_t | \mathbf{x}_{t-1})$$

This means that we have a simple factorization for the full sequence:

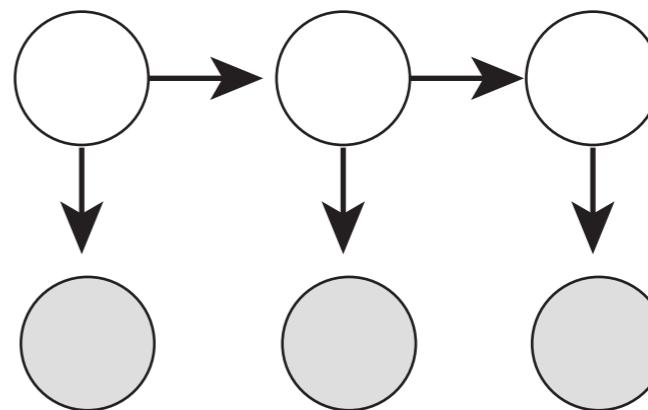
$$P(\mathbf{x}_t, \mathbf{x}_{t-1}, \dots, \mathbf{x}_0) = \prod_k P(\mathbf{x}_k | \mathbf{x}_{k-1})$$

Nice if individual components are simple (exponential family)

**Reminder: chain rule**

$$P(\mathbf{x}_t, \mathbf{x}_{t-1}, \dots, \mathbf{x}_0) = \prod_k P(\mathbf{x}_k | \mathbf{x}_{k-1}, \dots, \mathbf{x}_0)$$

## Some useful conditional independencies



$$P(\mathbf{x}_{1:t} | \mathbf{z}_i) = P(\mathbf{x}_{1:i} | \mathbf{z}_i) P(\mathbf{x}_{i+1:t} | \mathbf{z}_i)$$

$$P(\mathbf{x}_{1:i} | \mathbf{z}_{i+1}, \mathbf{x}_{i+1}) = P(\mathbf{x}_{1:i} | \mathbf{z}_{i+1})$$

$$P(\mathbf{x}_{i+2:t} | \mathbf{z}_{i+1}, \mathbf{x}_{i+1}) = P(\mathbf{x}_{i+2:t} | \mathbf{z}_{i+1})$$

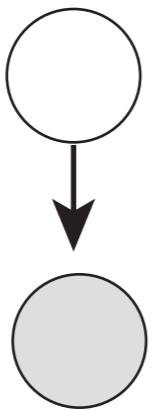
$$P(\mathbf{x}_{1:i} | \mathbf{z}_i, \mathbf{z}_{i+1}) = P(\mathbf{x}_{1:i} | \mathbf{z}_i)$$

$$P(\mathbf{x}_{i+1:t} | \mathbf{z}_i, \mathbf{z}_{i+1}) = P(\mathbf{x}_{i+1:t} | \mathbf{z}_{i+1})$$

$$P(\mathbf{x}_{1:t} | \mathbf{z}_i, \mathbf{z}_{i+1}) = P(\mathbf{x}_{1:i} | \mathbf{z}_i) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) P(\mathbf{x}_{i+2:t} | \mathbf{z}_{i+1})$$

# **Kalman filtering**

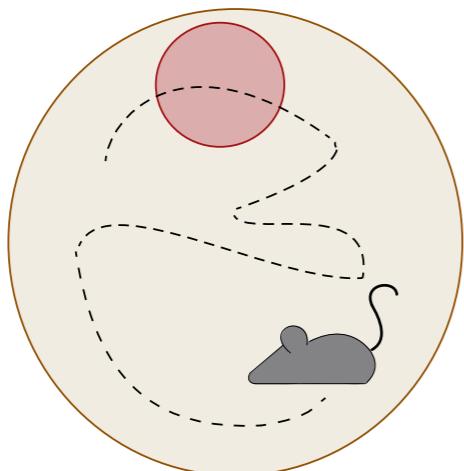
## Kalman filtering motivation



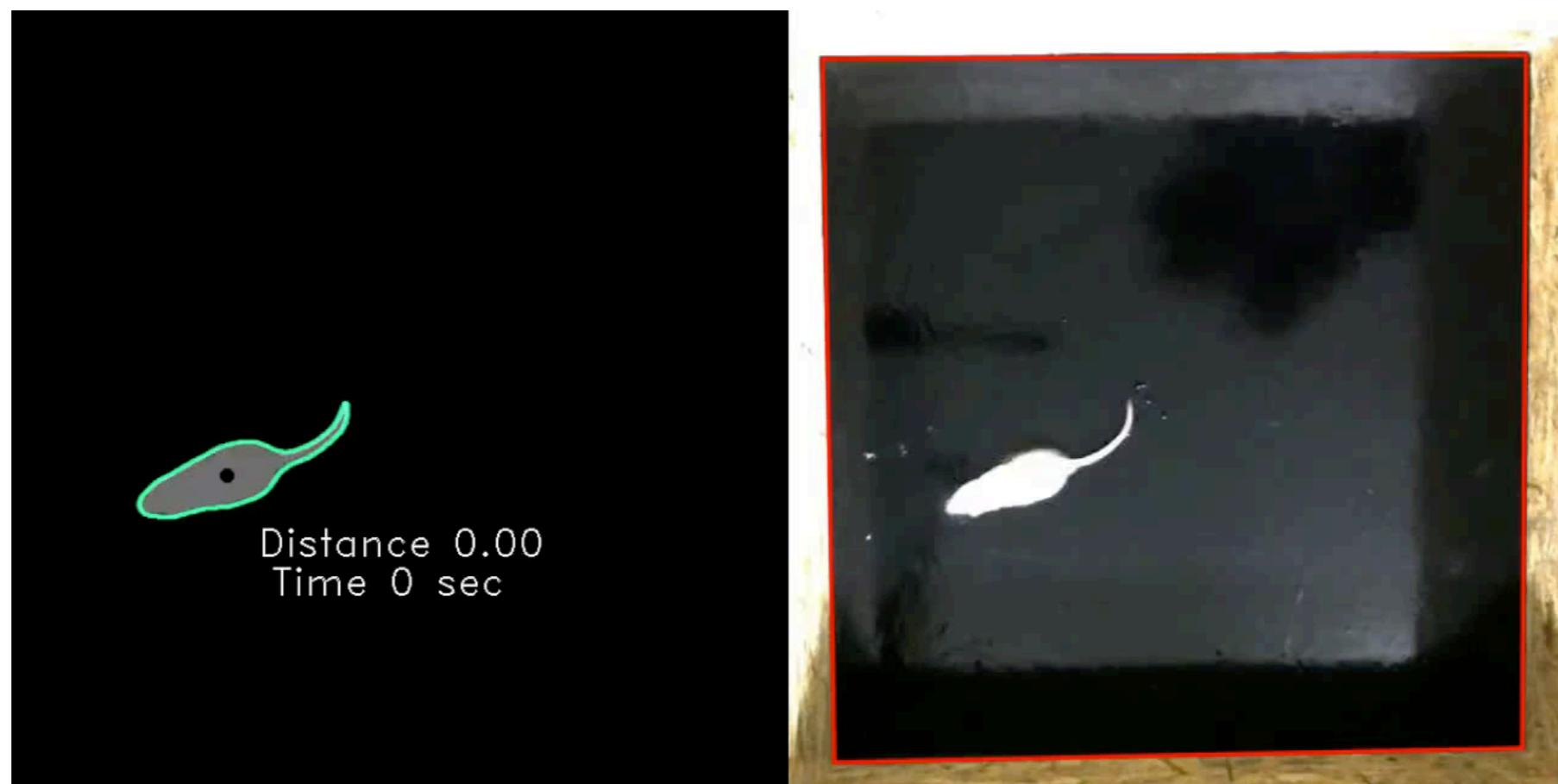
If I want to determine the value of a variable  $z$ ,  
based on a noisy measurement  $x$ ,  
I just take multiple measurements to get  
a better estimate

Can't do that if  $z$  changes over time,  
but there is still hope if  $z$  changes slowly  
(intuitively, some sort of weighted average)

Applications include: tracking objects (e.g., people, hands), navigation  
Computer vision: tracking features in video, stabilizing depth measurements,  
and many more



E.g. noisy measurements  
of the animal's  
position from top camera

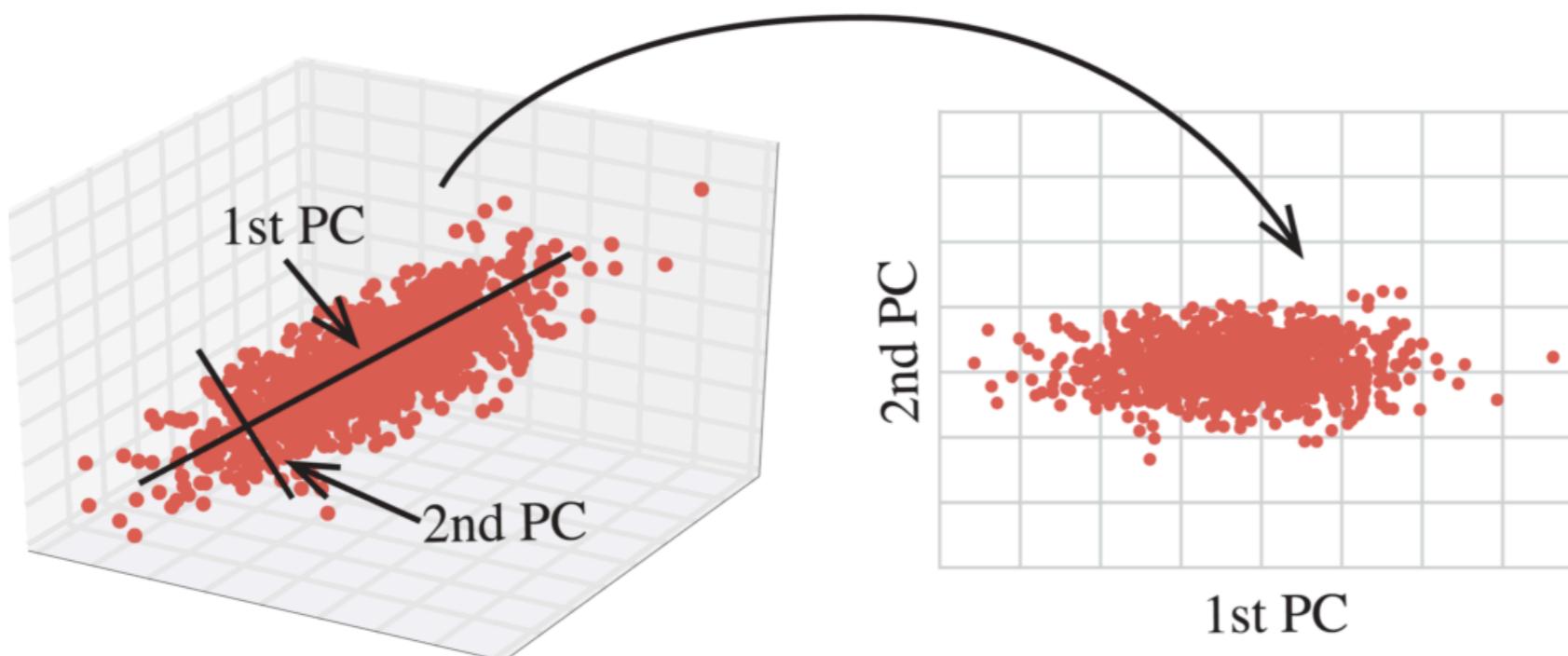


OpenCV demo

## Application : dimensionality reduction

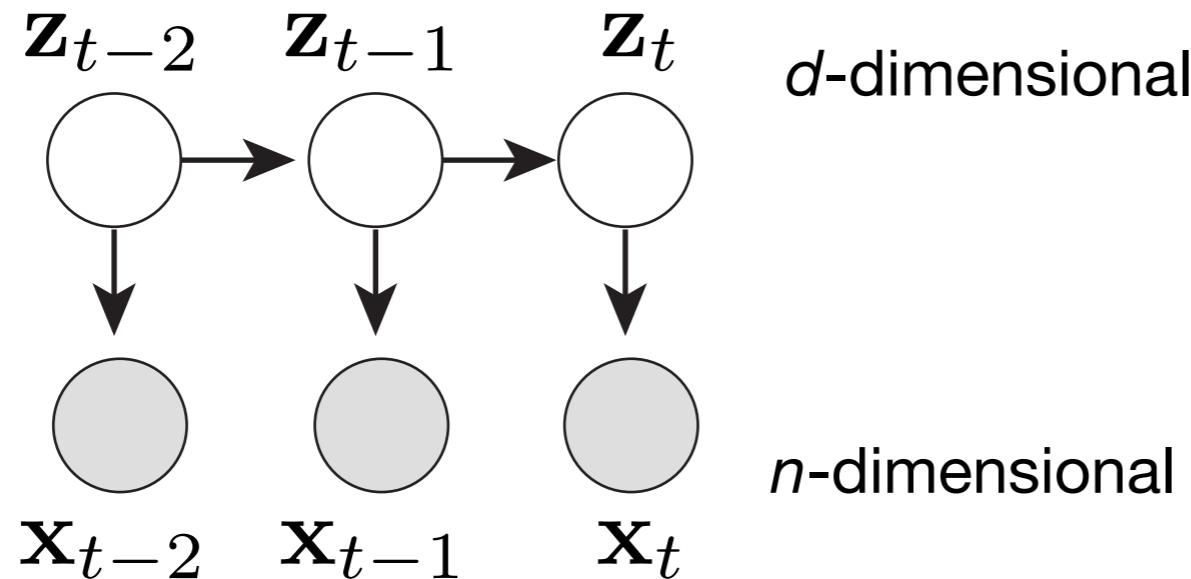
Often we have  $d < n$ , i.e. the data itself is low-dimensional.

Can think about this as a time-dependent version of PCA  
(we'll make this link formal later in the course)



Play with in the lab!

## Latent state models



$$\mathbf{z}_t = \mathbf{A}\mathbf{z}_{t-1} + \mathbf{w}_t$$

$$\mathbf{x}_t = \mathbf{C}\mathbf{z}_t + \mathbf{v}_t$$

Where the noise terms  
are iid Gaussian:

$$\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{Q})$$

$$\mathbf{v}_t \sim \mathcal{N}(0, \mathbf{R})$$

$$\mathbf{z}_0 \sim \mathcal{N}(\mu_0, \Sigma)$$

**We are still in a happy linear gaussian world!**

The noise terms are essential elements of the model : without  $\mathbf{w}$ , deterministic behavior given by leading eigenvalue of  $\mathbf{A}$ ; without  $\mathbf{v}$ , no latents

Without loss of generality, we can take  $\mathbf{Q}$  to be the identity\*,  $\mathbf{R}$  can be any positive definite matrix

\*If  $\mathbf{Q}$  is not diagonal, it is still positive definite, so it can be diagonalized  $\mathbf{Q} = \mathbf{E}\mathbf{D}\mathbf{E}^T$

With the variable change (rotating axes) we can always rewrite this so that the noise term is the identity  $\mathbf{y} = \mathbf{D}^{-1/2}\mathbf{E}^T\mathbf{x}$

## LDS inference

We are given the data up to  $s$ ,  $\{\mathbf{x}_1, \dots, \mathbf{x}_s\}$ .

We want to estimate the latent state,  $\mathbf{z}_t$ .

If  $t > s$  this is prediction, if  $s = t$  **filtering**, if  $t < s$  **smoothing**.

$$\mathbf{z}_t | \mathbf{z}_{t-1} \sim \mathcal{N}(\mathbf{A}\mathbf{z}_{t-1}; \mathbf{Q})$$

$$\mathbf{x}_t | \mathbf{z}_t \sim \mathcal{N}(\mathbf{C}\mathbf{z}_t; \mathbf{R})$$

$$P(\mathbf{z}_*, \mathbf{x}_*) = P(\mathbf{z}_*) \prod_i P(\mathbf{x}_i | \mathbf{z}_i) \quad * \text{ marks full series}$$

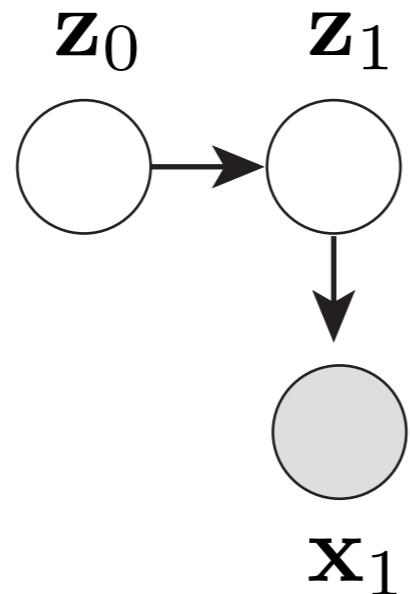
$$P(\mathbf{z}_*) = P(\mathbf{z}_0) \prod_i P(\mathbf{z}_t | \mathbf{z}_{t-1})$$

Everything is Gaussian, so we could explicitly write it down.

Kalman filtering uses recursion to compute things more efficiently.

Note: This can be derived as a specific instance of message passing in trees (see e.g. Bishop, chp.6), but we're not going to do this here.

Intuitively, we can construct the forward pass by induction



**2 sub-steps:**

$$z_1 | z_0$$

**Take into account what we know about  $z_0$**

$$z_1 | x_1, z_0$$

**Take into account evidence  $x_1$**

## Basic Gaussian identities

a  $d$ -dimensional multidimensional gaussian (normal) density for  $\mathbf{x}$  is:

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

E1. Rescaling and shifting gaussian variables

$$E[\mathbf{Ax} + \mathbf{y}] = \mathbf{A}(E[\mathbf{x}]) + \mathbf{y}$$

$$\text{Covar}[\mathbf{Ax} + \mathbf{y}] = \mathbf{A}(\text{Covar}[\mathbf{x}])\mathbf{A}^T$$

E2. Sum of 2 independent gaussian variables

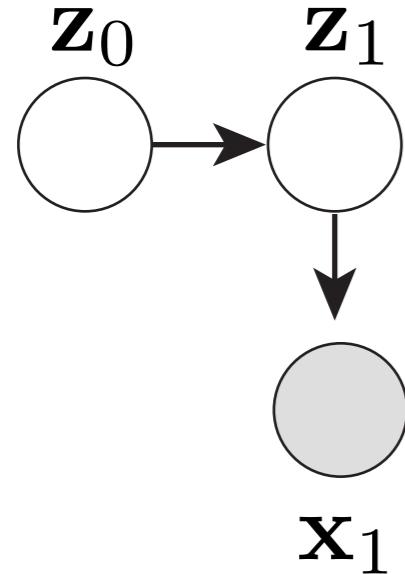
$$\begin{aligned}\boldsymbol{\Sigma}_c &= \boldsymbol{\Sigma}_a + \boldsymbol{\Sigma}_b \\ \boldsymbol{\mu}_c &= \boldsymbol{\mu}_a + \boldsymbol{\mu}_b\end{aligned}$$

E3. Conditioning

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{B} \end{bmatrix} \right)$$

$$\mathbf{x} | \mathbf{y} \sim \mathcal{N} \left( \mathbf{a} + \mathbf{CB}^{-1}(\mathbf{y} - \mathbf{b}), \mathbf{A} - \mathbf{CB}^{-1}\mathbf{C}^T \right)$$

Intuitively, we can construct the forward pass by induction



**2 sub-steps:**

**Forecasting**

$$z_1 | z_0 \quad \mu_{1|0} = A\mu_0$$

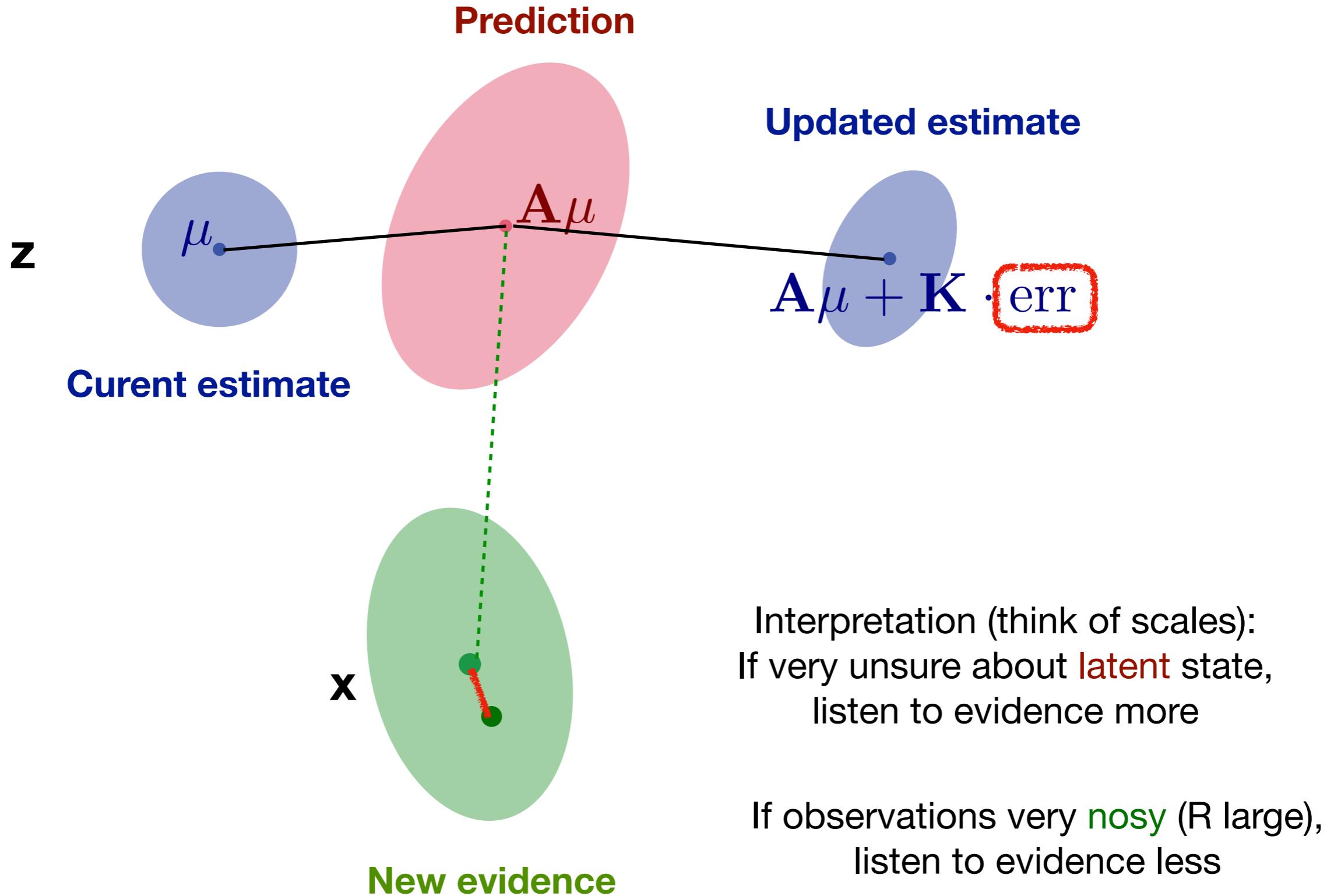
$$\Sigma_{1|0} = A\Sigma_0 A^T + Q$$

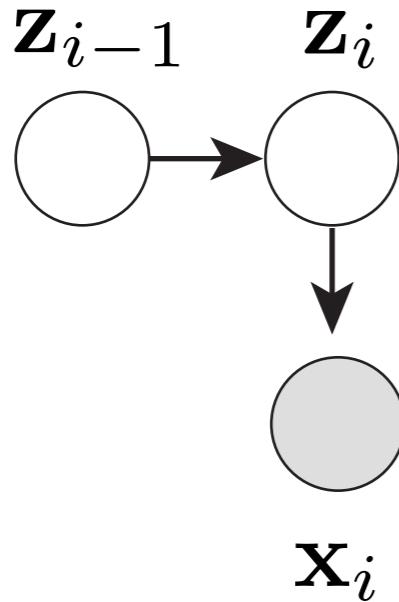
**Combine with new evidence**

$$z_1 | x_1, z_0 \quad \mu_{1|1} = \mu_{1|0} + K(x_1 - C\mu_{1|0})$$

$$\Sigma_{1|1} = \Sigma_{1|0} - K C \Sigma_{1|0}$$

$$K = \Sigma_{1|0} C^T (C \Sigma_{1|0} C^T + R)^{-1}$$





Exactly in the same way, if we have a posterior distribution for  $\mathbf{z}_{i-1}$ , we can use it instead of  $P(\mathbf{z}_0)$  to compute the posterior for  $\mathbf{z}_i$ .

$$\mu_{i|i} = \mu_{i|i-1} + \mathbf{K}_i (\mathbf{x}_i - \mathbf{C}\mu_{i|i-1})$$

$$\Sigma_{i|i} = \Sigma_{i|i-1} - \mathbf{K}_i \mathbf{C} \Sigma_{i|i-1}$$

$$\mathbf{K}_i = \Sigma_{i|i-1} \mathbf{C}^T (\mathbf{C} \Sigma_{i|i-1} \mathbf{C}^T + R)^{-1} \quad \text{Kalman gain}$$

The gain changes dynamically!

**Notation:** double index i|j:  
 i-we are estimating variable  $\mathbf{z}_i$ ,  
 j-datapoints observed up to (including)  $\mathbf{x}_j$

## Smoothing - backward sweep

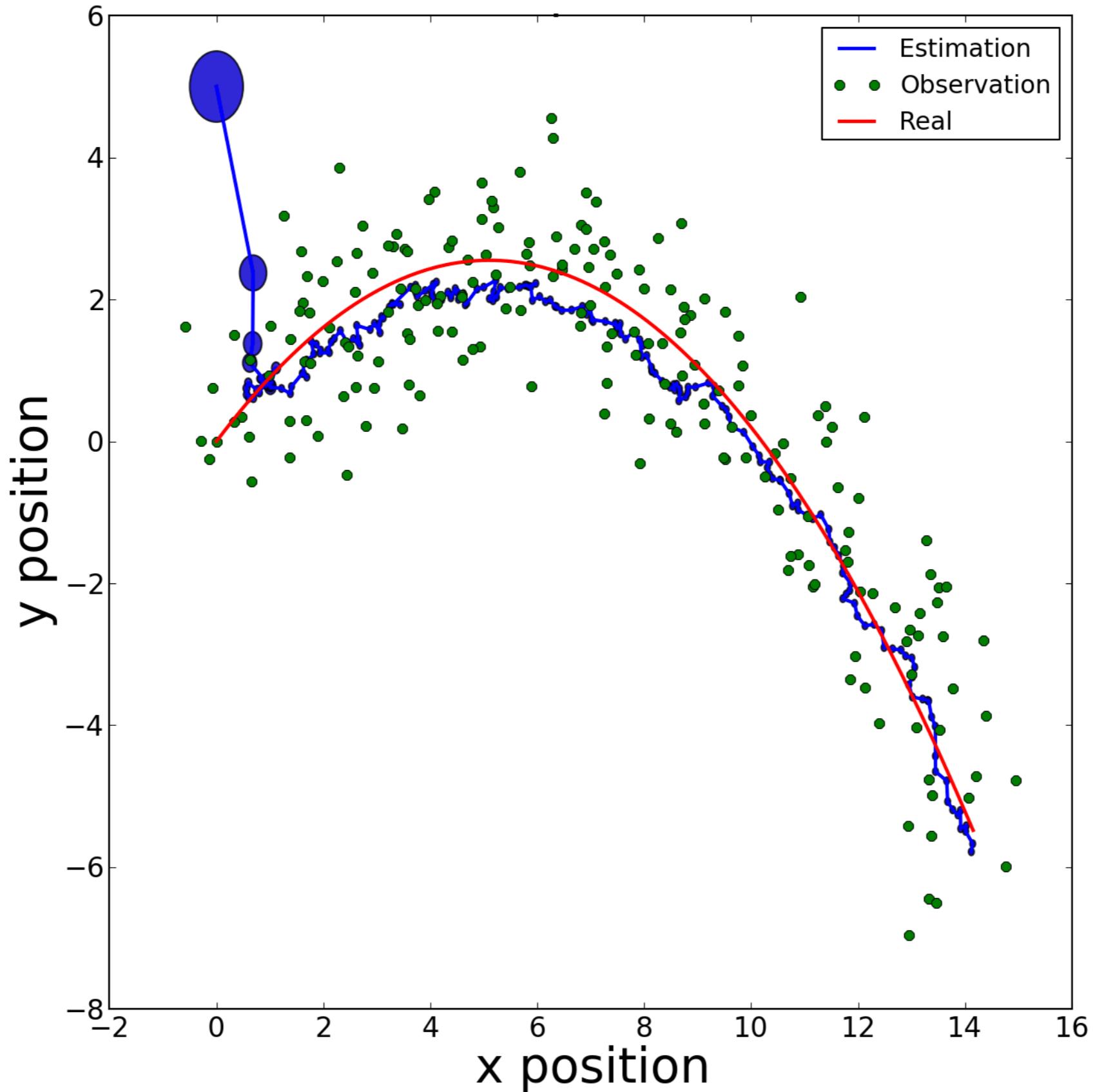
Same principle, but now we propagate information backwards  
in time, using the estimates we have already  
(we no longer need the data)

$$\mu_{i|t} = \mu_{i|i} + \mathbf{F}_i(\mu_{i+1|t} - \mu_{i+1|i})$$

$$\Sigma_{i|t} = \mathbf{F}_i(\Sigma_{i+1|t} - \Sigma_{i+1|i})\mathbf{F}_i^T + \Sigma_{i|i}$$

$$\mathbf{F}_i = \Sigma_{i|i} \mathbf{A}^T \Sigma_{i+1|i}^{-1}$$

See handout for proof



**How do we use this for prediction?**

**How do we use learn the parameters?**

## **Estimating parameters via expectation maximization**

Find parameters that are most consistent with the observed data

To simplify notation we use shorthand  $\theta = \{\mathbf{A}, \mathbf{C}, \mathbf{Q}, \mathbf{R}, \mu_0, \Sigma_0\}$

The goal is to find the parameters that maximize the (log) likelihood

$$\mathcal{L}(\theta) = \log P(\mathbf{x}_* | \theta)$$

Which we get after **marginalizing out the latents**

$$P(\mathbf{x}_* | \theta) = \int P(\mathbf{x}_*, \mathbf{z}_* | \theta) d\mathbf{z}$$

## General idea of EM

$$\log \int_z P(\mathbf{x}, \mathbf{z} | \theta) d\mathbf{z} = \log \int_z Q(\mathbf{z}) \frac{P(\mathbf{x}, \mathbf{z} | \theta)}{Q(\mathbf{z})} d\mathbf{z}$$
$$= \log \left( \mathbb{E}_Q \left[ \frac{P(\mathbf{x}, \mathbf{z} | \theta)}{Q(\mathbf{z})} \right] \right)$$

via **Jensen's Inequality**

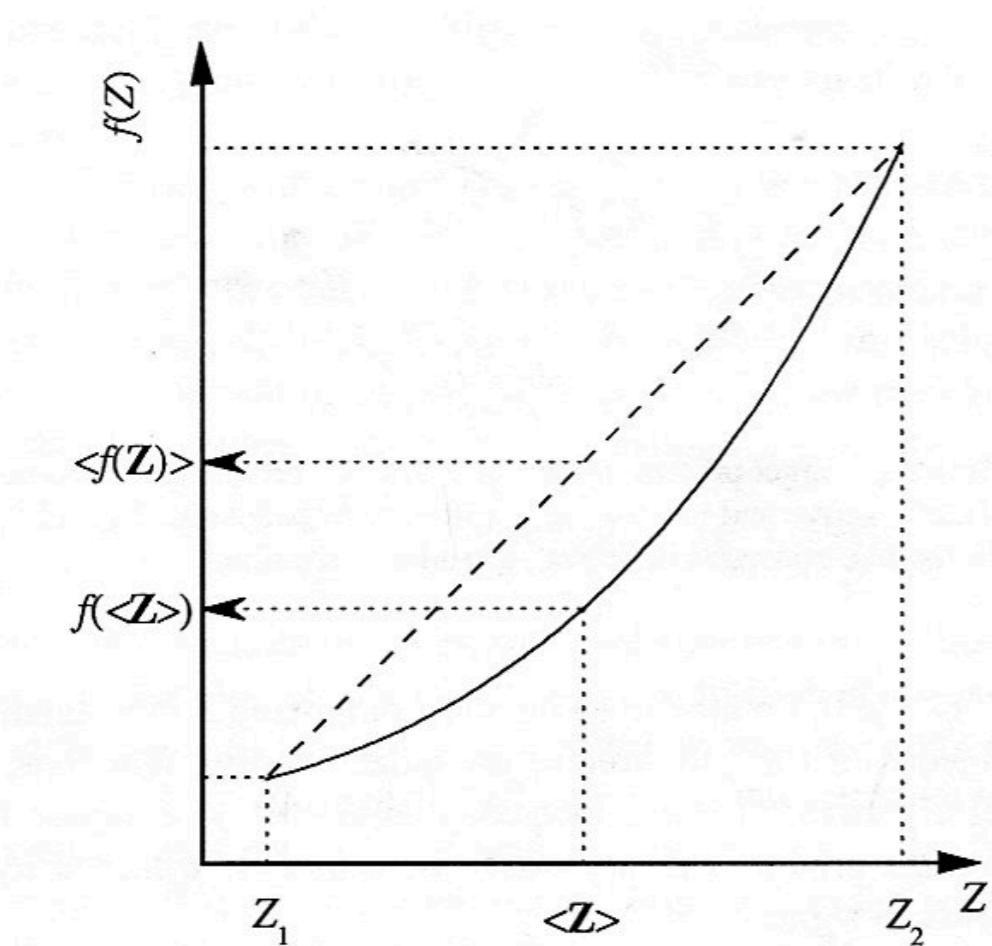
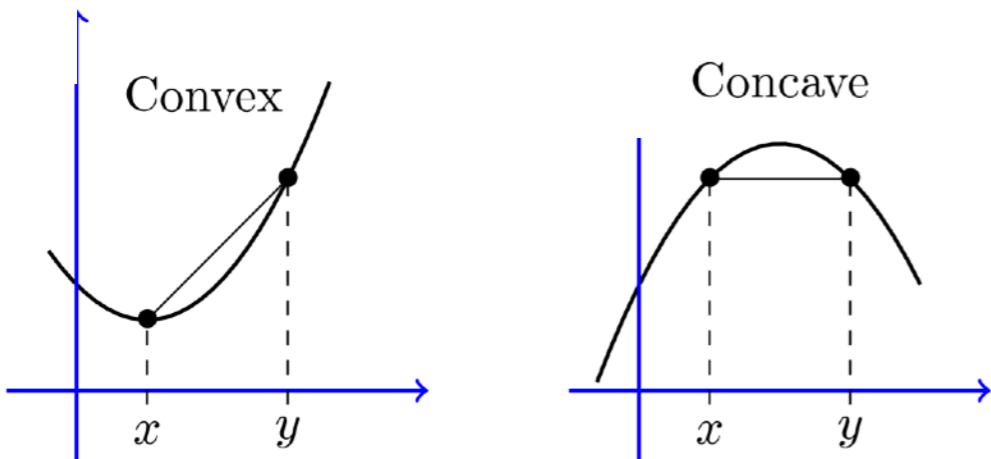
$$\geq \int_z Q(\mathbf{z}) \log \frac{P(\mathbf{x}, \mathbf{z} | \theta)}{Q(\mathbf{z})} d\mathbf{z}$$
$$= \mathbb{E}_Q \left[ \log \frac{P(\mathbf{x}, \mathbf{z} | \theta)}{Q(\mathbf{z})} \right]$$

$$= \int_Q Q(\mathbf{z}) \log P(\mathbf{x}, \mathbf{z} | \theta) d\mathbf{z} - \int_Q Q(\mathbf{z}) \log Q(\mathbf{z}) d\mathbf{z}$$

$$= \mathcal{F}(Q, \theta) \quad \text{Free energy}$$

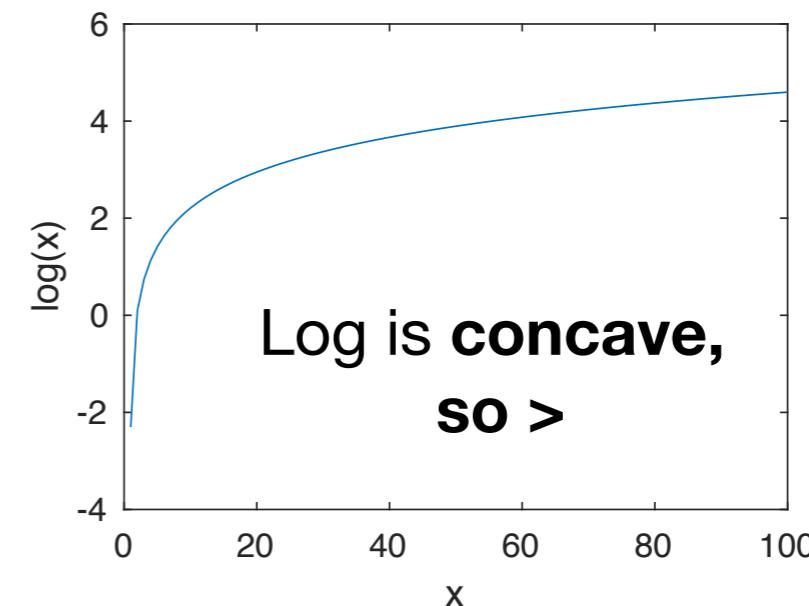
Big picture: log of integral is not nice, so we replace it with bound on integral of log

## Reminder: convex vs concave functions; Jensen's inequality



$$f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)]$$

If  $f$  is **convex**



EM alternates between finding a good approximation  $Q$ ,  
and then changing the parameters to improve the approx likelihood

This is done coordinate-wise: improve one keeping the other fixed, then swap.

**E step:** 
$$Q_{k+1} \leftarrow \arg \max_Q \mathcal{F}(Q, \theta_k)$$

**M step:** 
$$\theta_{k+1} \leftarrow \arg \max_\theta \mathcal{F}(Q_{k+1}, \theta)$$

It is guaranteed that the likelihood will **never decrease** during this procedure.

### General philosophy:

use simple parametric form to ‘guesstimate’ the state of the latent variables, given current model parameters. Then use this fictitious complete data to find new model parameters.

**E-step:**  $Q_{k+1}(\mathbf{z}) = P(\mathbf{z}|\mathbf{x}, \theta_k)$

The approximation is locally exact

