

# Welcome!

## #pod-031

### Week #2, Day 3

(Reviewed by: Deepak)



facebook  
Reality Labs



mindCORE  
Center for Outreach, Research, and Education

UC Irvine



UNIVERSITY  
OF MINNESOTA

CIFAR

IEEEbrain

SIMONS  
FOUNDATION

TEMPLETON WORLD  
CHARITY FOUNDATION

THE KAVLI  
FOUNDATION



CHEN TIANQIAO & CHRISSEY  
INSTITUTE

WELLCOME

GATSBY

Bernstein Network  
Computational Neuroscience

NB  
DT

hhmi | janelia  
Research Campus

# Agenda

- Day-#8
  - Tutorial part 1 (Sequential Probability Ratio Test)
    - 4 Exercises
  - Tutorial part 2 (Hidden markov models)
    - 4 Exercises
  - Tutorial part 3 (Kalman filters)
    - 3 Exercises + 1 bonus

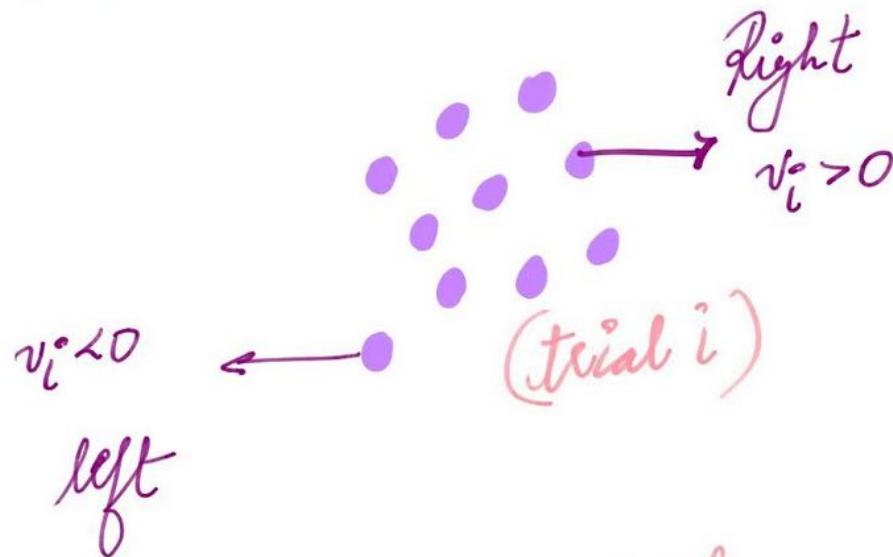


# *Tutorial #1*

# *Explanations*

# Simplified random dot motion task

W2D3\_pod 031



Set of all  $v_i$  are generated by -  
true / data generating /  
fixed probability distribution'

$P_L \{ N(-1, \sigma^2) \}$   
accepting hypothesis  
 $H_L$

$P_R \{ N(+1, \sigma^2) \}$   
accepting hypothesis  
 $H_R$

$\{ P_L / P_R : \text{true data generating distribution} \}$

To distinguish between two hypotheses  $H_L$  and  $H_R$ , we use the sequential probability ratio test by running simulations of a Drift Diffusion Model (DDM).

## Stopping Criteria

As independent and identically distributed (i.i.d) samples from the true data-generating distribution coming in, we accumulate our evidence linearly until a certain criterion is met before deciding which hypothesis to accept. After seeing fixed amount of data or if likelihood ratio passes predefined threshold

*Why thresholding?  
Controlling mechanism*

# Noisy Observations

Expected mean + Drifting term

Observation Noise + Diffusion term

# Why Drifting?

Due to the noisy nature of observations, there will be a drifting term governed by expected mean output. The **drift-diffusion model** (DDM) is a well defined **model**, that is proposed to implement an optimal decision policy. It is the continuous analog of a random walk **model**.

# Diffusion

*The diffusion term is governed by observation noise.  
With just a few parameters and a conceptually simple process, this model can describe behavioral performance (choice and reaction time) for a whole range of tasks (in humans and animals); the best way to develop better intuitions is to play with simulations.*

# Integrating concepts of drift and diffusions



# Why log likelihood?

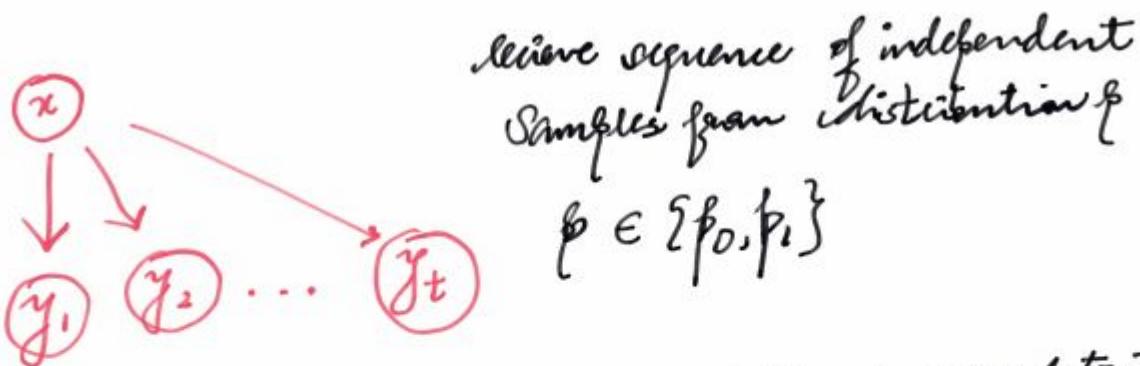
In statistics, the likelihood-ratio test assesses the goodness of fit of two competing statistical models based on the ratio of their likelihoods,

Combine sensory evidence and our prior experience  
with Bayes' theorem, producing a posterior  
probability distribution that would let us  
choose between most probable of these 2 options -

accepting hypothesis  $H_L$

or accepting hypothesis  $H_R$ .

choose more evidence before making a decision.



determined by binary latent variable  $x$  & need to test b/w  
 $H_L: p = p_L \text{ or } x = 0$   
 $H_R: p = p_R \text{ or } x = 1$

When we see  $n$  samples  $\{x_1, \dots, x_n\}$

Calculate total log likelihood ratio as evidence  
for decision

$$S_n = \log \frac{\prod_{i=1}^n P_R(x_i)}{\prod_{i=1}^n P_L(x_i)}$$

$$= \sum_{i=1}^n \log P_R(x_i) - \sum_{i=1}^n \log P_L(x_i)$$

Considering independence of samples.

$$S_n = S_{n-1} + \log \frac{P_R(x_n)}{P_L(x_n)} = S_{n-1} + \log \lambda_n$$

(calculated incrementally when new data points come in sequentially)

Stopping criteria #1 (fixed time)  
 make decision based on  $s_n$  when we've collected  
 $n$  samples

accept  $H_R$  if  $s_n > 0$

accept  $H_L$  if  $s_n < 0$   
 accept  $H_R$  with probability  $\frac{1}{2}$  if  $s_n = 0$

Significance level / desired error rate

$$\alpha = \frac{1}{1 + \exp(|s_n|)}$$

Stopping criteria #2 : Thresholding

assume equal probability to make false positive }  
 decision & to make false negative decision & denote }  
 with  $\alpha$

accept hypothesis  $H_R$  if  $s_n \geq b$   
 $H_L$  if  $s_n \leq a$

thresholds determined by -

$$a = \log \frac{\alpha}{1-\alpha} ; \quad b = \log \frac{1-\alpha}{\alpha}$$

$$(a = -b)$$

As a drift diffusion model -

- assume different gaussian observation models  
conditioned on discrete latent variable  $z$

$$p_L(x|z=0) = N(\mu_L, \sigma_L^2)$$

$$p_R(x|z=1) = N(\mu_R, \sigma_R^2)$$

log likelihood ratio for  $x_i$  (single data pt)

$$\log \Lambda_i = \log \frac{\sigma_L}{\sigma_R} - 0.5 \left[ \frac{(x_i - \mu_R)^2}{\sigma_R^2} - \frac{(x_i - \mu_L)^2}{\sigma_L^2} \right]$$

assume true data generating distribution ( $P_R$ )  
 (without loss of generality)

express  $x_i$  as  $\mu_R + \sigma_R \varepsilon$

$$\log \lambda_i = \left( \log \frac{\sigma_L}{\sigma_R} + 0.5 \frac{(\mu_L - \mu_R)^2}{\sigma_L^2} \right) +$$

$$\left( \frac{\mu_R - \mu_L}{\sigma_L} \varepsilon - 0.5 \left[ 1 - \left( \frac{\sigma_R}{\sigma_L} \right)^2 \right] \varepsilon^2 \right)$$

diffusion part

std gaussian

drift part

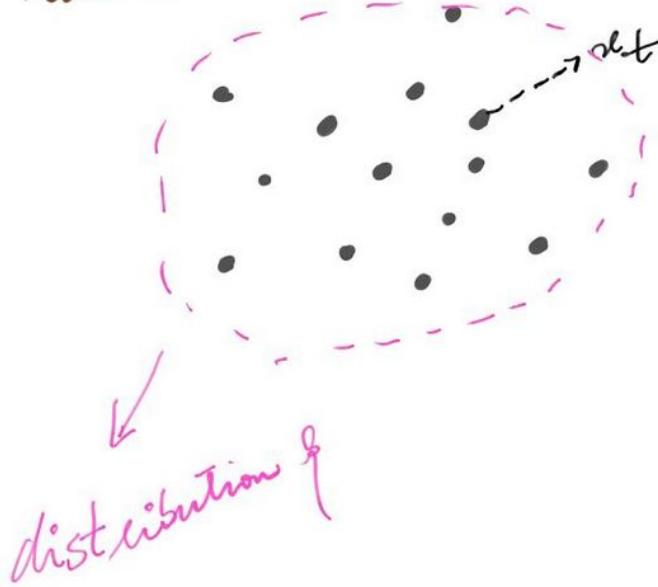
let  $\sigma_L = \sigma_R$

Classical drift diffusion equation  
 (analytical solutions for mean and expected auto covariance)

$$\log N_i = 0.5 \frac{(M_R - M_L)^2}{\sigma_L^2} + \frac{M_R - M_L}{\sigma_L^2} \varepsilon$$

$$\varepsilon \sim N(0,1)$$

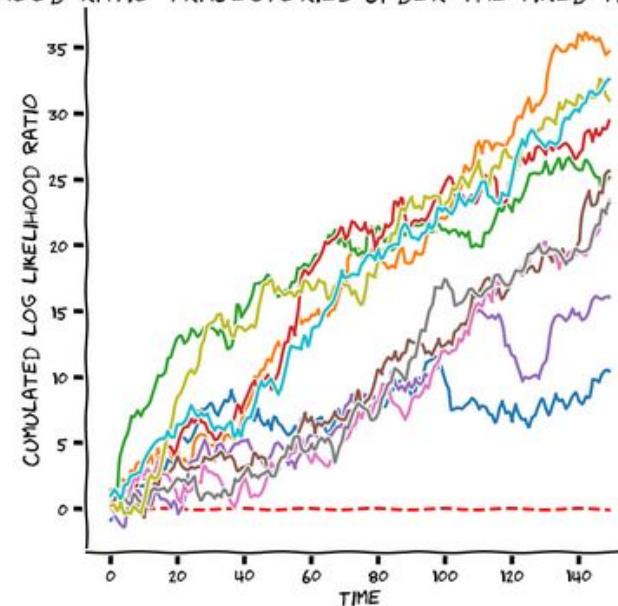
random dot motion task.

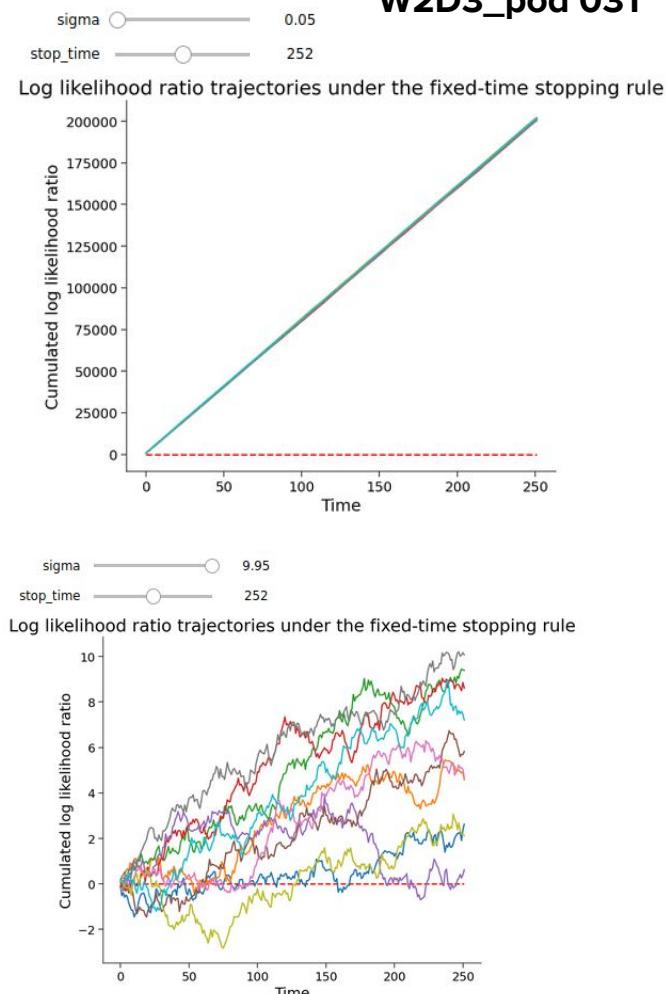
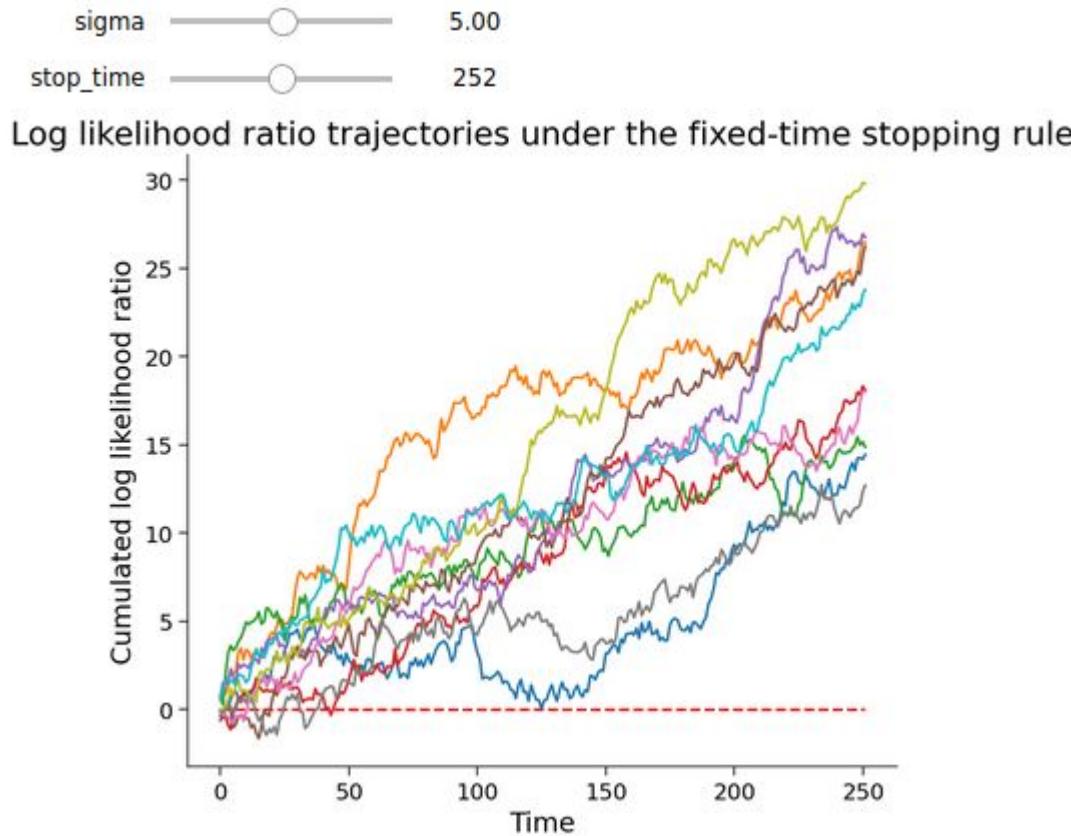


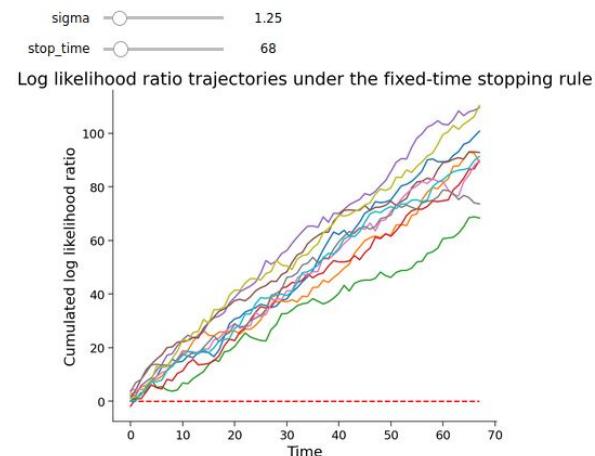
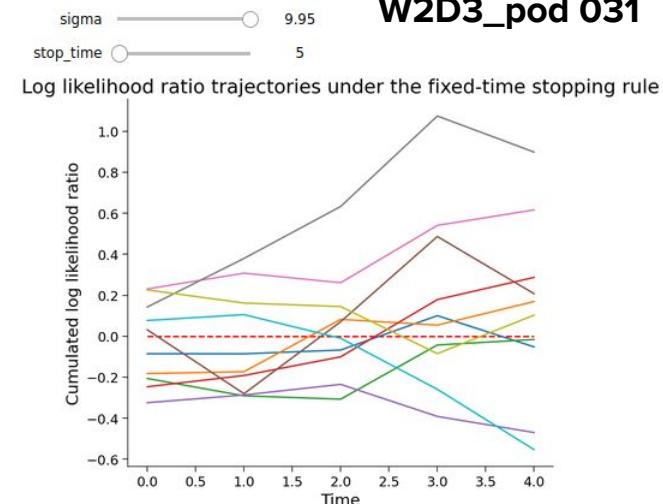
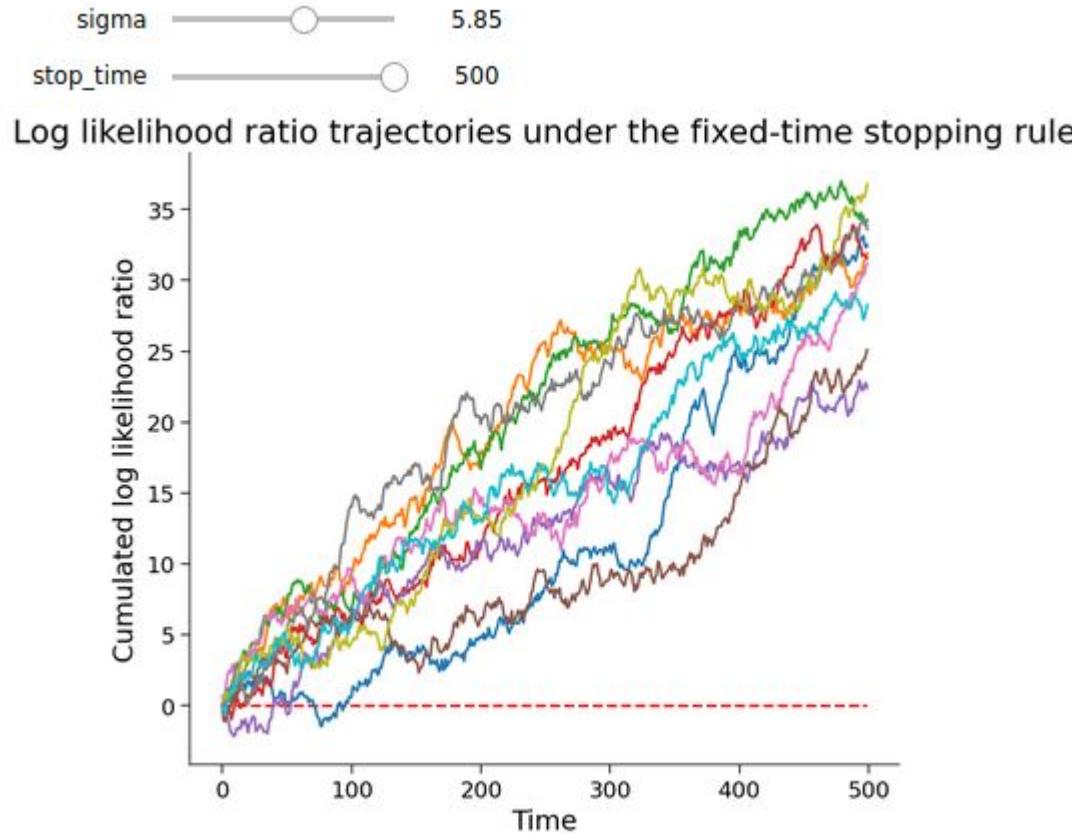
$$p_L = \mathcal{N}(-\mu, \sigma^2)$$

$$p_R = \mathcal{N}(\mu, \sigma^2)$$

LOG LIKELIHOOD RATIO TRAJECTORIES UNDER THE FIXED-TIME STOPPING RULE





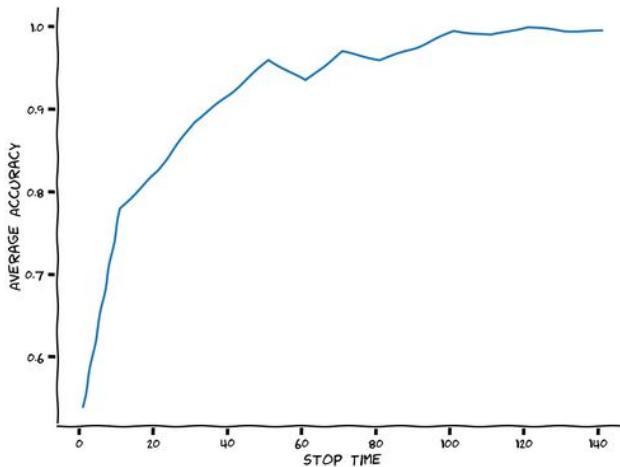


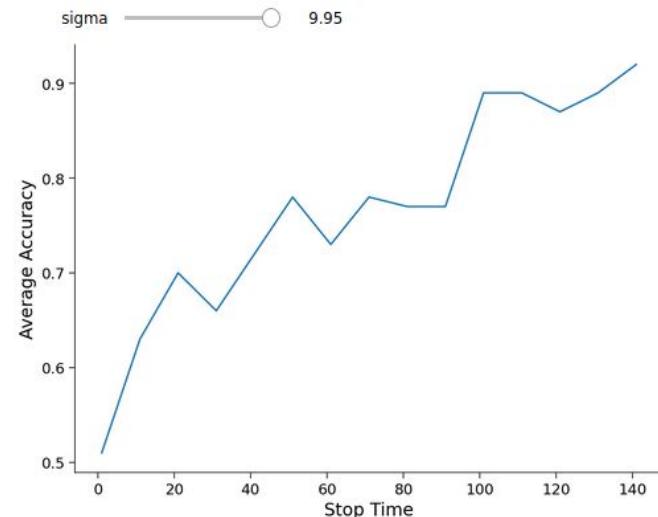
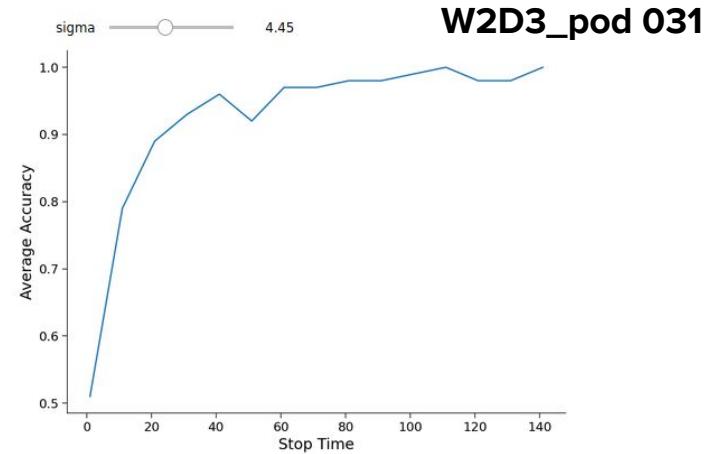
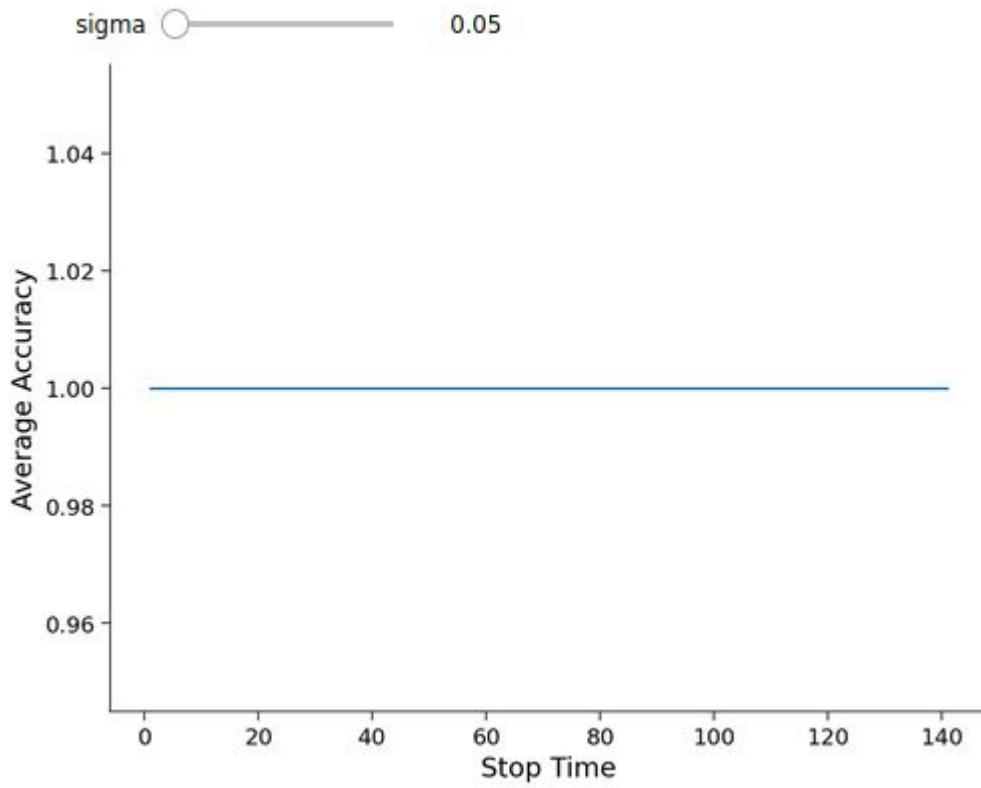
# Observations

If you stop taking samples too early, (e.g., make a decision after only seeing 5 samples), or there's a huge amount of observation noise that buries the signal, you are likely to be driven by observation noise to a negative cumulated log likelihood ratio and thus make a wrong decision.

Accuracy: proportion of correct trials across  
repeated simulations.

$$\frac{\# \text{ correct decisions}}{\# \text{ total simulation runs}}$$





Simulating drift diffusion model with fixed thresholds

- define desired error rate

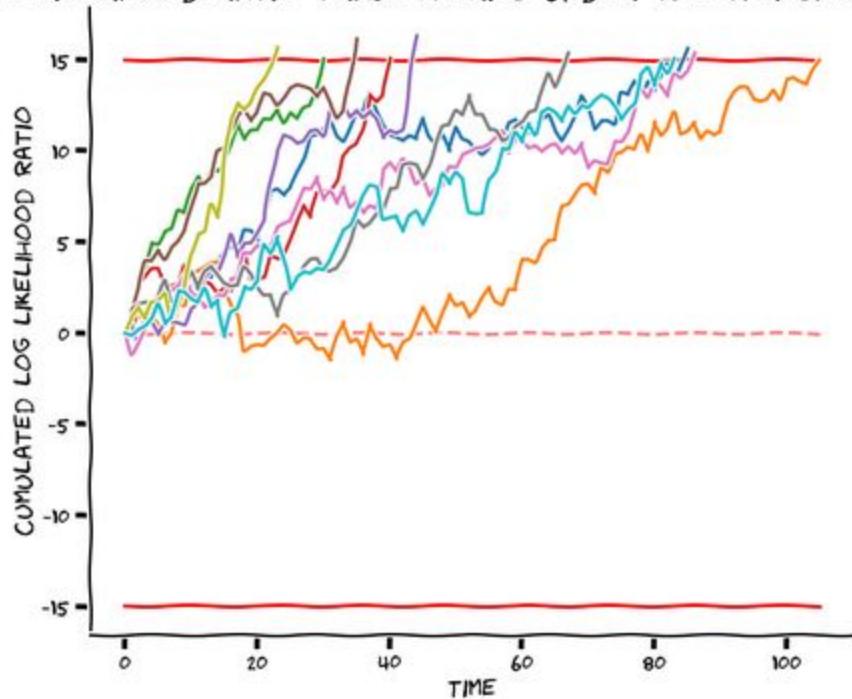
(make measurements until error rate is reached)

{Experimental evidence suggests evidence accumulation / thresholding strategy happens @ neuronal level!}

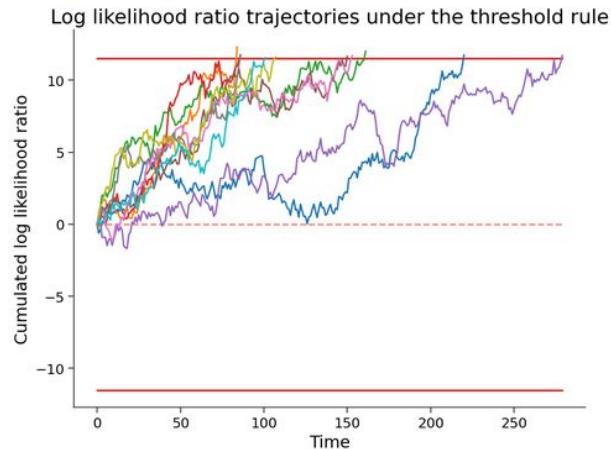
$$\text{th}_L = \log \frac{\alpha}{1 - \alpha} = -\text{th}_R$$

$$\text{th}_R = \log \frac{1 - \alpha}{\alpha} = -\text{th}_L$$

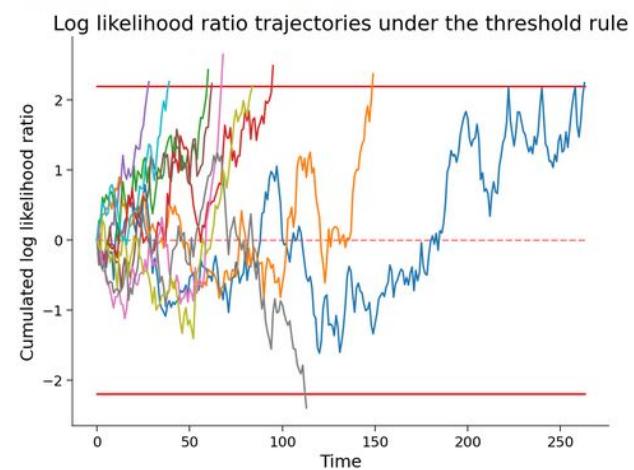
## LOG LIKELIHOOD RATIO TRAJECTORIES UNDER THE THRESHOLD RULE



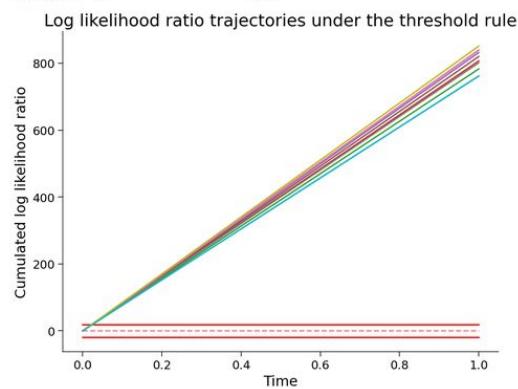
sigma  5.00  
log10\_alpha  -5.00



sigma  9.95  
log10\_alpha  -1.00



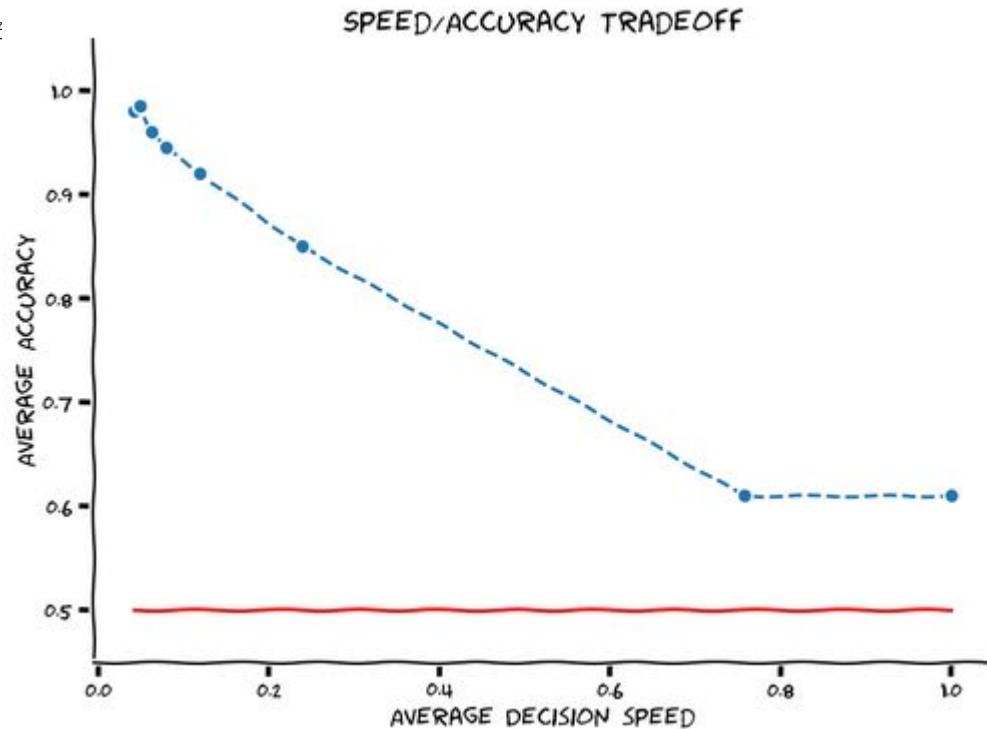
sigma  0.05  
log10\_alpha  -8.00



The faster you make a decision, the lower your accuracy often is. This phenomenon is known as the **speed/accuracy tradeoff**.

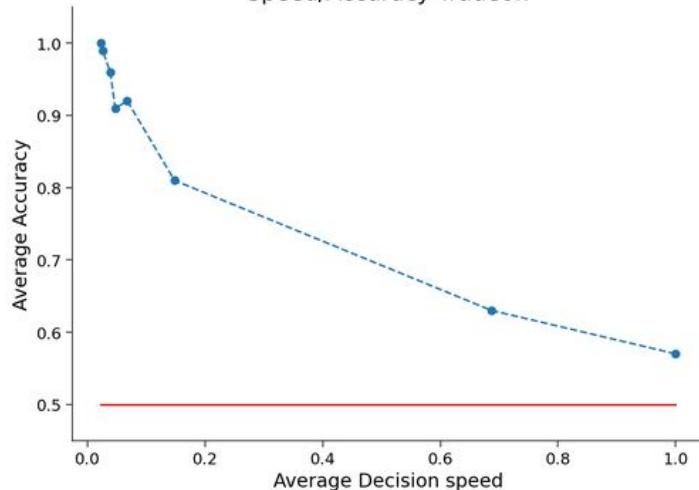
Study how average decision "speed" ( $1/\text{length}$ ) changes with average decision accuracy.

We use speed rather than accuracy because in real experiments, subjects can be incentivized to respond faster or slower; it's much harder to precisely control their decision time or error threshold.



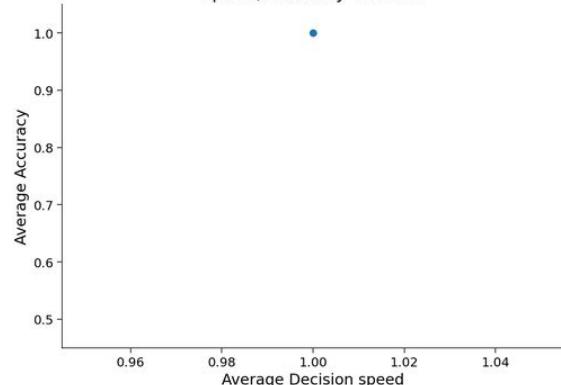
sigma  5.00

Speed/Accuracy Tradeoff



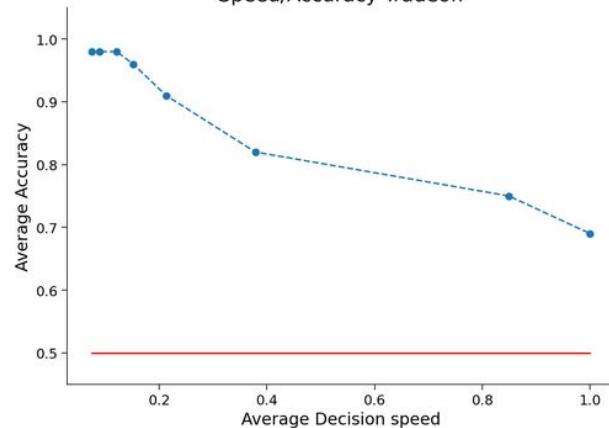
sigma  0.05

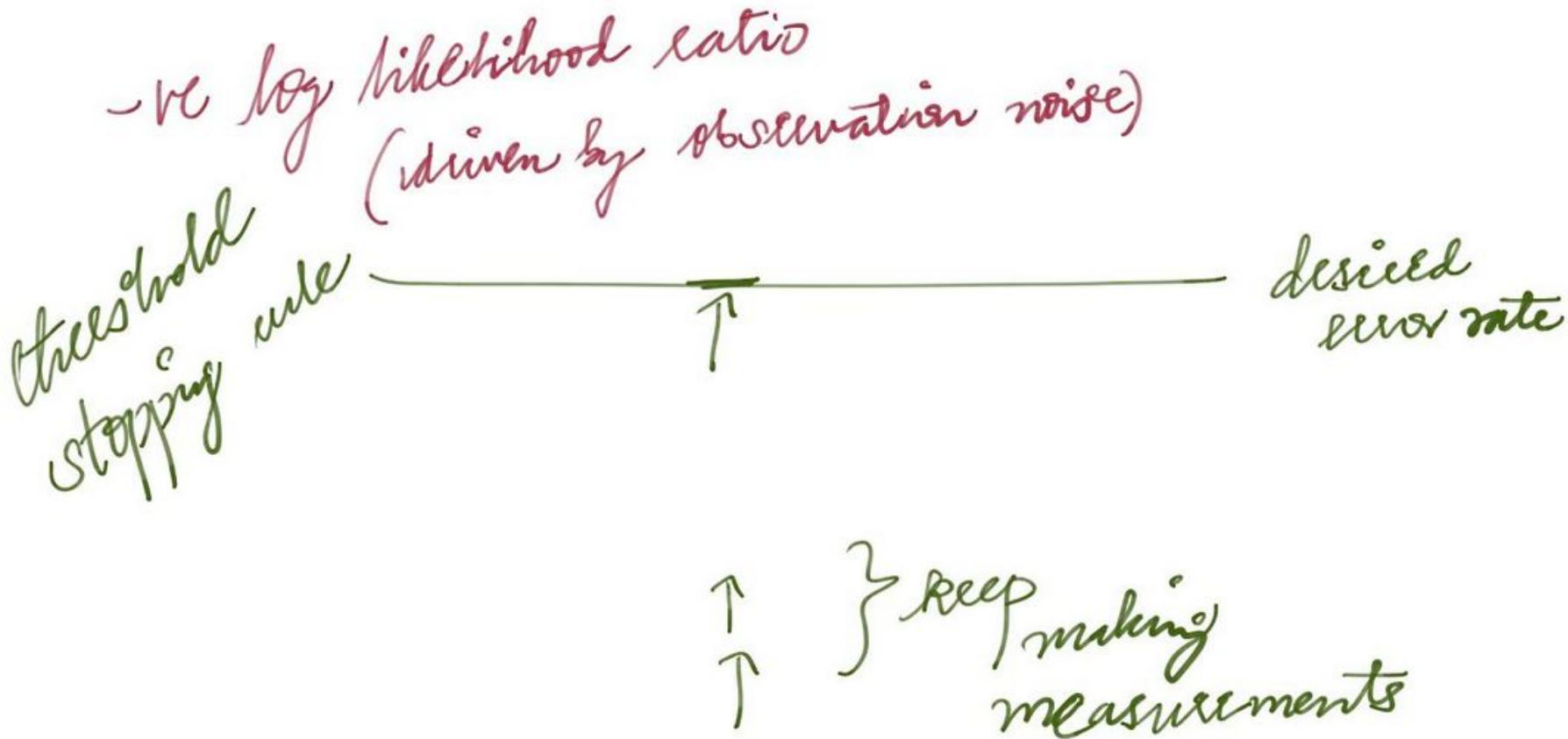
Speed/Accuracy Tradeoff



sigma  9.95

Speed/Accuracy Tradeoff





# Summary

Simulation of Drift Diffusion Models to perform decision making:

- Calculate individual sample evidence as the log likelihood ratio of two candidate models, accumulate evidence from new data points, and make decision based on current evidence
- Run repeated simulations to get an estimate of decision accuracies
- Implement the thresholding stopping rule where we can control our error rate by taking adequate amounts of data, and calculate the evidence threshold from desired error rate
- Explore and gain intuition about the speed/accuracy tradeoff for perceptual decision making

# *Tutorial #2*

# *Explanations*

# Hidden Markov Models

# Objective

- Simulate a Hidden Markov Model and observe how changing the transition probability and observation noise impact samples
- look at how uncertainty increases as we make future predictions without evidence (from observations) and how to gain information from the observations.
- Calculate how predictive probabilities propagates in a Markov Chain with no evidence.
- Combine new evidence and prediction from past evidence to estimate latent states.

hidden markov model :

binary latent variable :  $x_t \in \{0, 1\}$

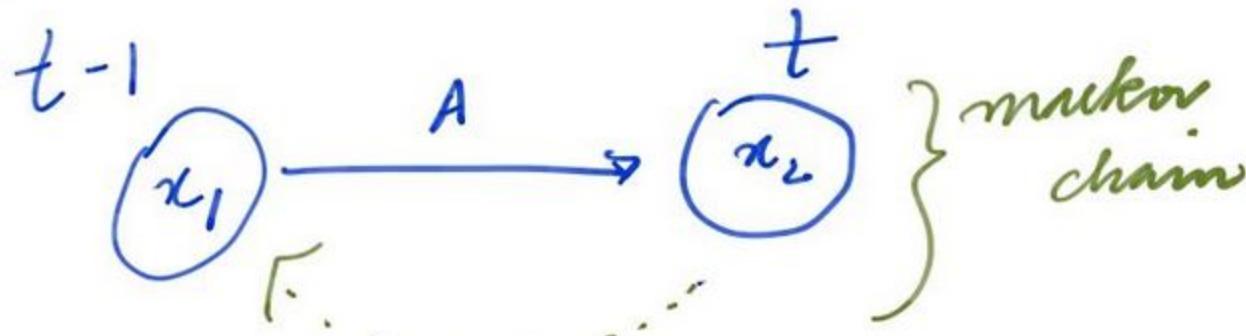
1D gaussian emission model :  $y_t | x_t \sim N(\mu_x, \sigma_{x_t}^2)$

## *Hidden markov models*

*Hidden Markov Models are a class of models that allow us to reason about the dynamics of a set of unobserved states that lead to the changing sensory inputs or data we observe.*

*Organisms and neural systems often are thought to transition between a set of discrete states (up/down states, sleep/wake, etc.) which may only be indirectly observable through their impact on neural activity.*

hidden markov models



dependency is  
markov property

latent state - not fixed over time  
 probabilistically switch/jump to different state at each time step.

$p(\text{state at time } t)$  determined by state at  $t-1$ .  
 $\hookrightarrow$  Markov/markovian

Quantitatively,

$$P(x_t = j) = \sum_i a_{ij} P(x_{t-1} = i)$$

transition matrix {  
 transition probability {  
 probability of jumping from  $i$  to  $j$

latent states  $x_t$ : can't be observed

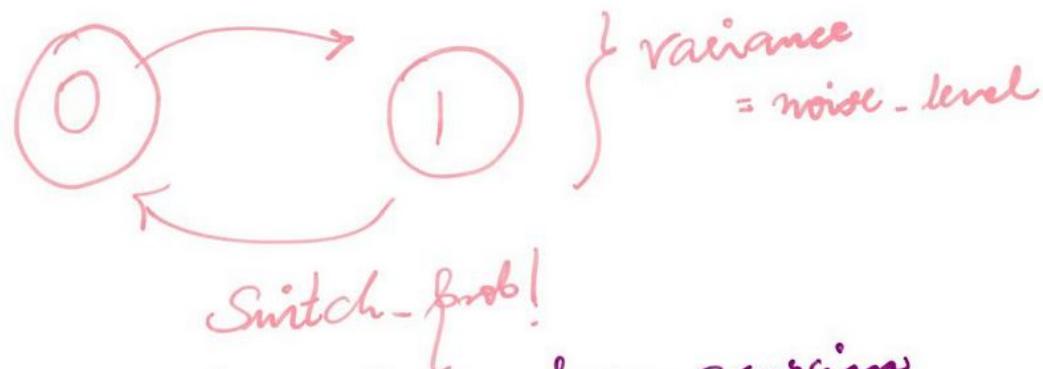
observe noisy measurement  $y_t$  generated from  $x_t$ .

Emission models:

gaussian distribution  $y_t | x_t \sim N(\mu_{x_t}, \sigma_{x_t}^2)$   
(continuous observables)

poisson distribution  $y_t | x_t \sim \text{pois}(\lambda)$   
(discrete observables)

two state HMM with gaussian emissions.



each state emits observations drawn from gaussian

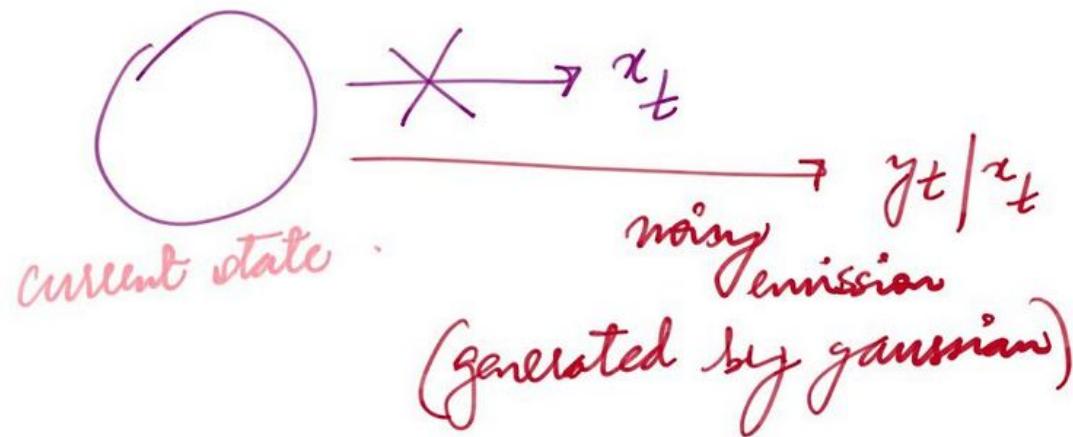
with  $\mu=1$  for state = 0 }  
 $\mu=-1$  for state = 1 }

State transitions:

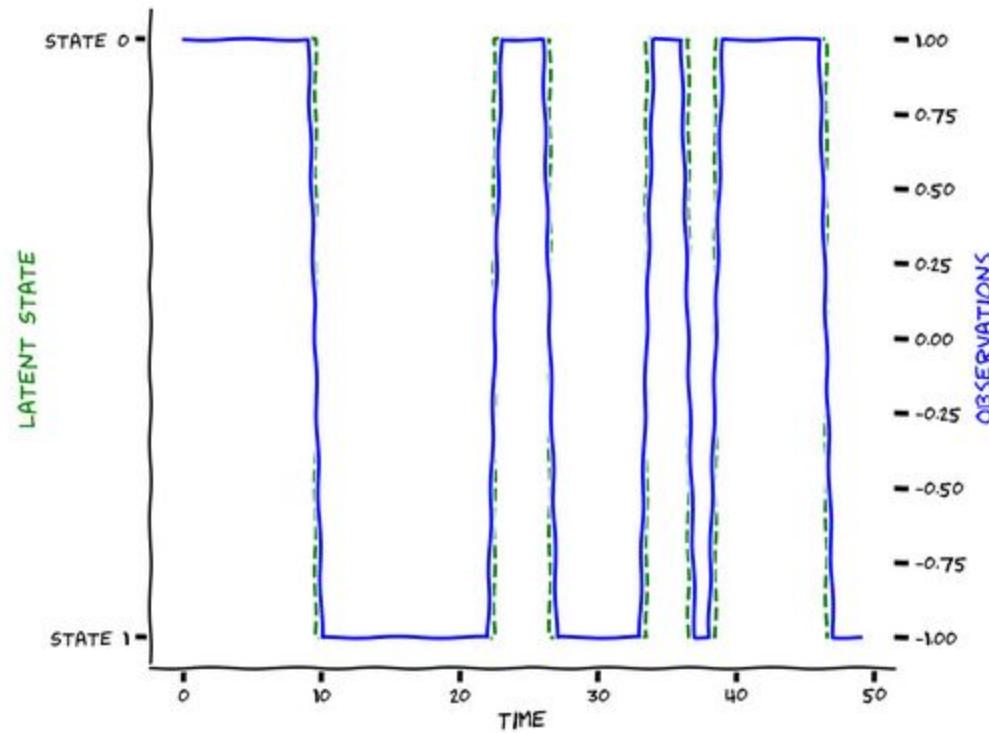
transition matrix  $A_{ij} = \begin{pmatrix} p_{\text{stay}} & p_{\text{switch}} \\ p_{\text{switch}} & p_{\text{stay}} \end{pmatrix}$

$$p_{\text{stay}} = 1 - p_{\text{switch}}$$

hidden part of HMM -

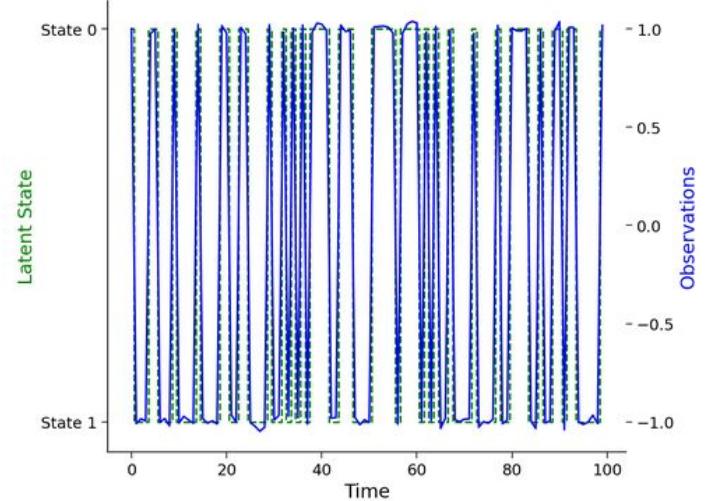


State-obs-variance = noise-level

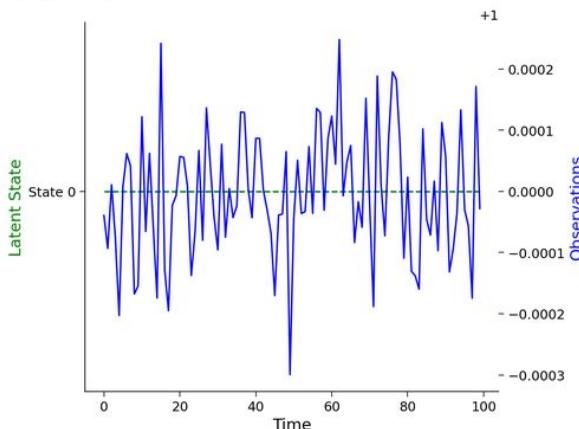


# W2D3\_pod 031

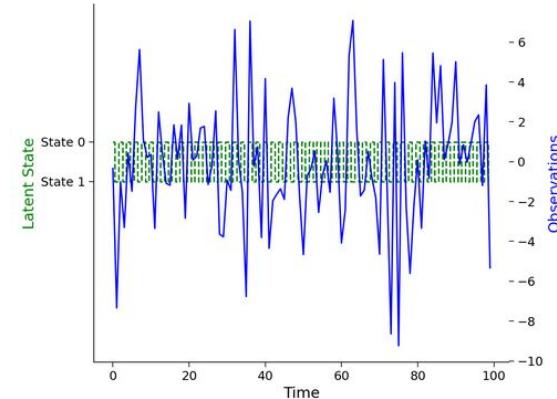
switch\_prob   
log10\_noi...



switch\_prob   
log10\_noi...



switch\_prob   
log10\_noi...



latent state (changes)

(state  
0)

- - - - -

certainty level  $x$

certainty level  $y$

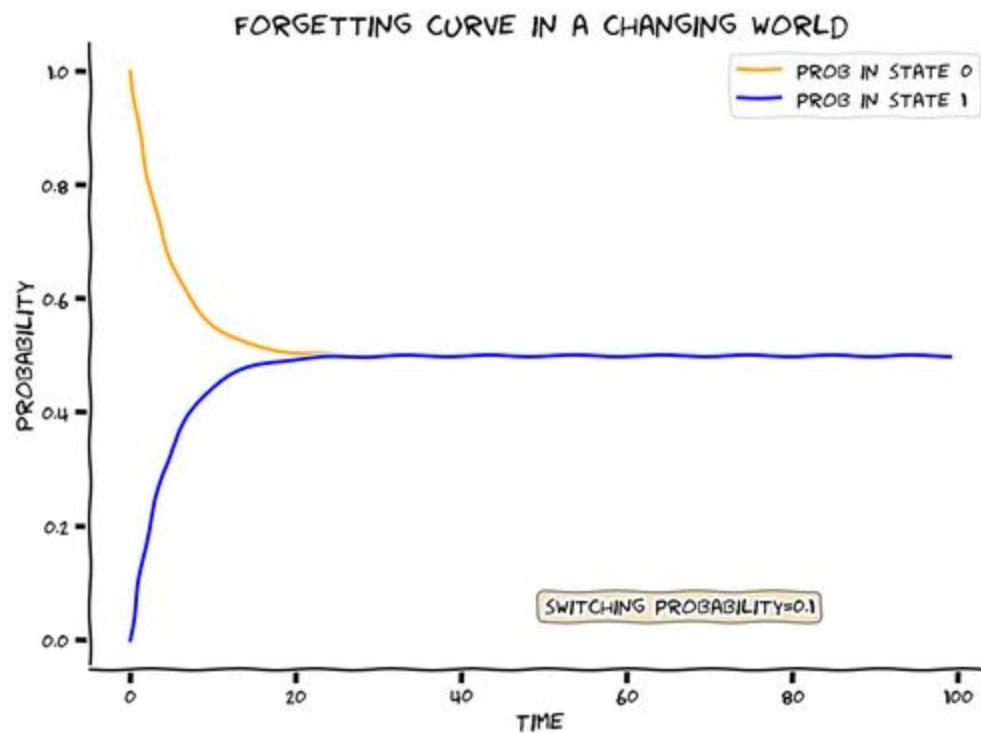
$x > y$

markov chain: influence of current state will }  
decay over time }

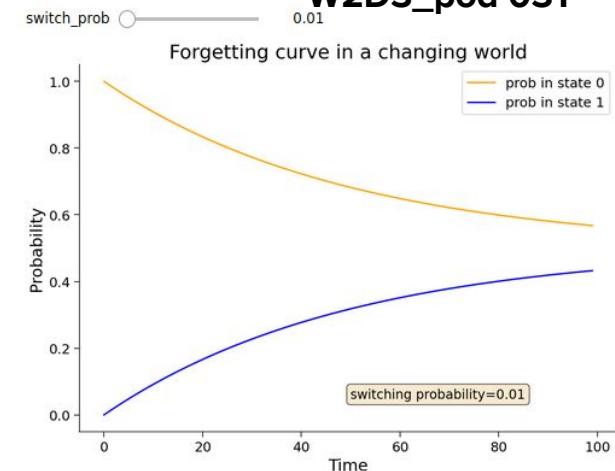
When we try to make predictions of future states }  
in a markov chain based on current knowledge }  
without future evidence .

forget current state information when }  
predicting future states  
without any observation

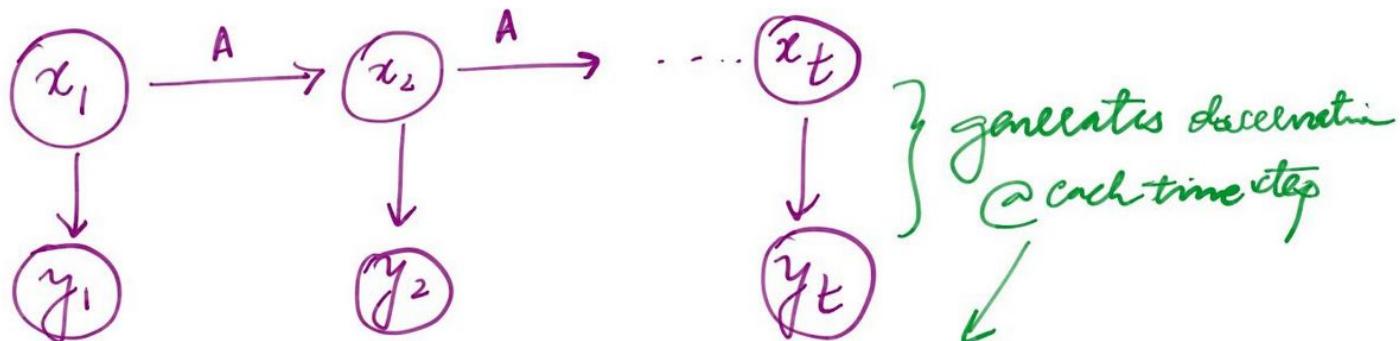
Assumption:  $x_0 = 0$ ; imposed prior probabilities }  
 $p(x_0) = [1, 0]$



**W2D3\_pod 031**



How switching probabilities affects the information loss.



base info/gain uncertainty  
 exponentially when predicting into future  
 without further evidence as marker chain  
 diffuses through exponential number of possible paths.

evidence to improve state estimate

## incorporating evidence into inference

posterior marginal distribution  $p(x_t | y_{1:t})$

→ recursively when evidence  $y_t$  comes in @ each time  
using forward pass of forward backward algorithm.

Suppose we know posterior marginal of  $x_{t-1}$  given all  
observations upto  $t-1$

When new observation  $y_t$ :

By Bayes rule & Markov property:

$$\phi(x_t | y_{1:t}) \propto \phi(x_t | y_{1:t-1}) P(y_t | x_t)$$

*prediction*

*new data*  
*likelihood*

for two gaussian  
observation models

$$\text{posterior } (x_t | y_{1:t}) \propto \text{prediction } (x_t | y_{1:t-1})$$

• likelihood  $(y_t | x_t)$

Algorithm :

create-model ( $0.1$ ,  $0.5$ )  
 switch prob., noise level

calculate marginal posterior distribution

$f(x_t | y_{1:t-1})$  @ time  $t$

from last posterior  $f(x_{t-1} | y_{1:t-1})$  @ time  $t-1$

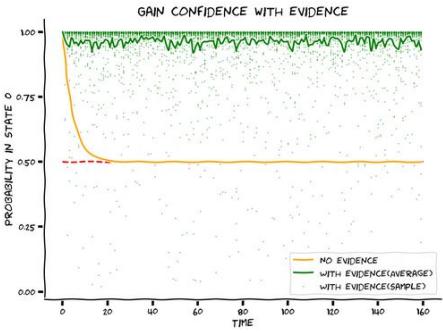
→ calculate predictive distribution

$$f(x_t = j | y_{1:t-1}) = \sum_i A_{ij} f(x_{t-1} = i | y_{1:t-1})$$

→ calculate likelihood of new data

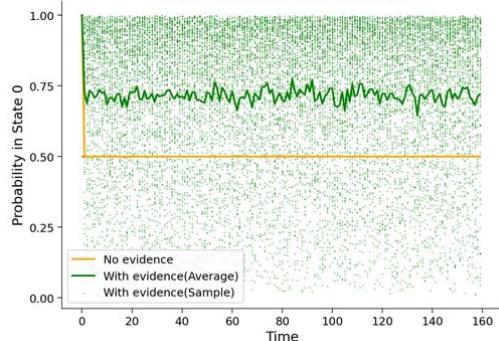
→ multiply likelihood & prediction element wise  
 normalize

Plot of information loss due to diffusion together with the information recovered/uncertainty reduced due to evidence. The difference between the former and the latter is the amount of uncertainty that still remains because of observation noise.



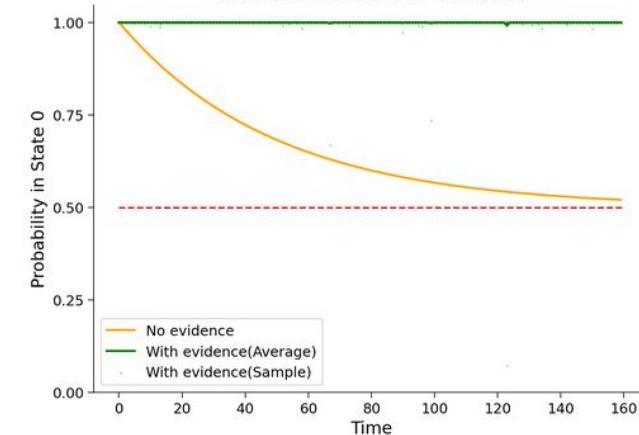
switch\_prob 0.50  
noise\_level 1.50  
nsample 100  
T 160

Gain confidence with evidence



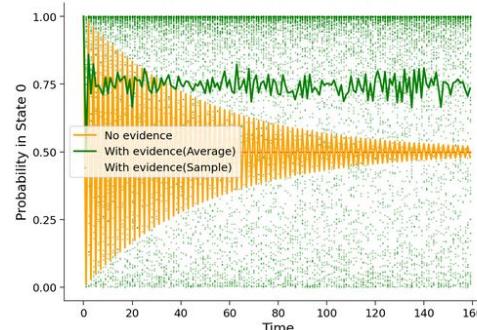
switch\_prob 0.01  
noise\_level 0.10  
nsample 100  
T 160

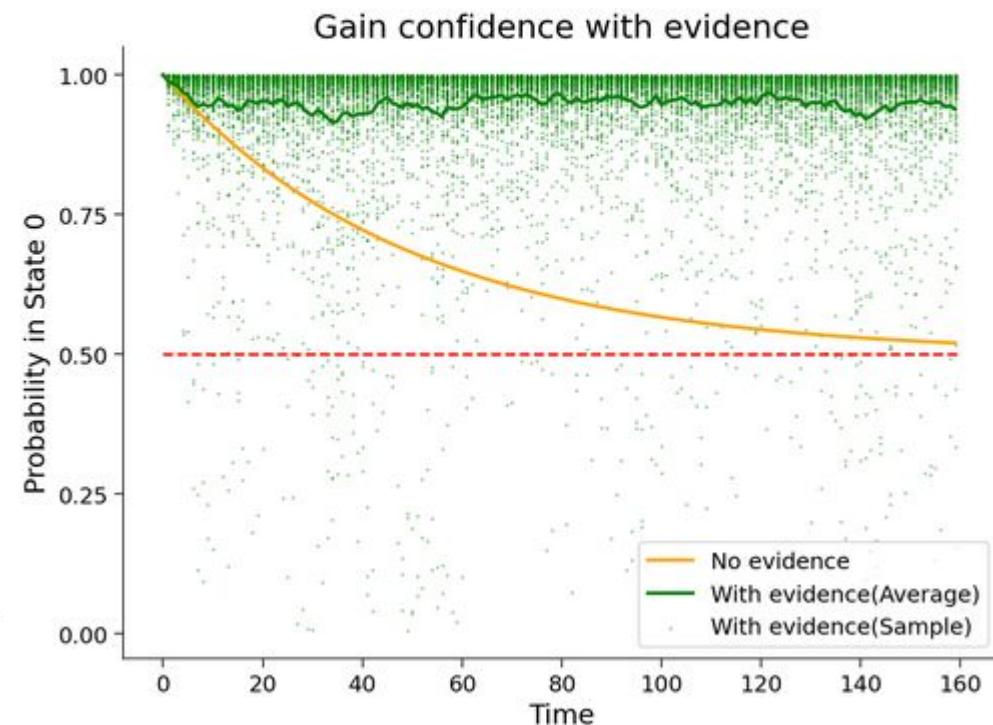
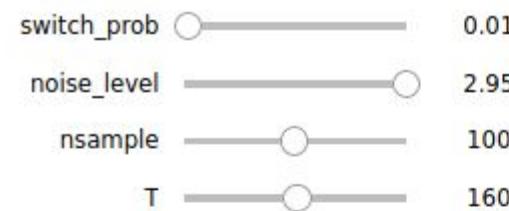
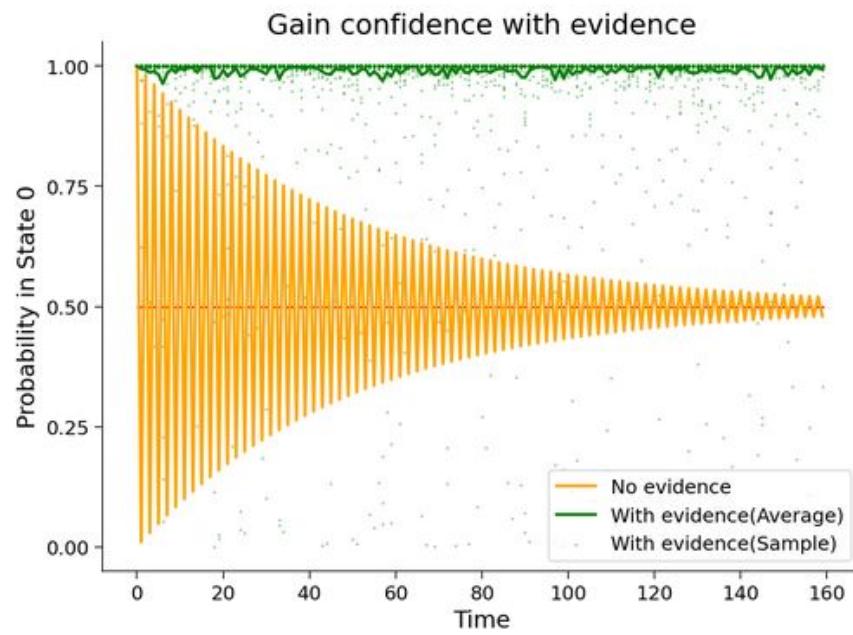
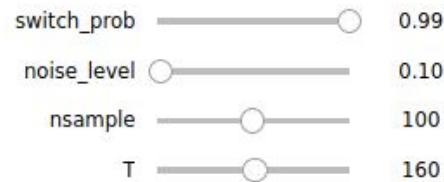
Gain confidence with evidence



switch\_prob 0.99  
noise\_level 2.95  
nsample 100  
T 160

Gain confidence with evidence





# OBSERVATIONS

If `switch_prob` or `noise_level` is large, some sample inference dots fall below 0.5. This means we are making false inferences about which latent state we are in.

In this exercise, let's make a forward inference of a random state sequence rather than a constant one by observing its noisy Gaussian outputs.

# forward inference of HMM

① hmm with five probabilities  $(0.5, 0.5)$

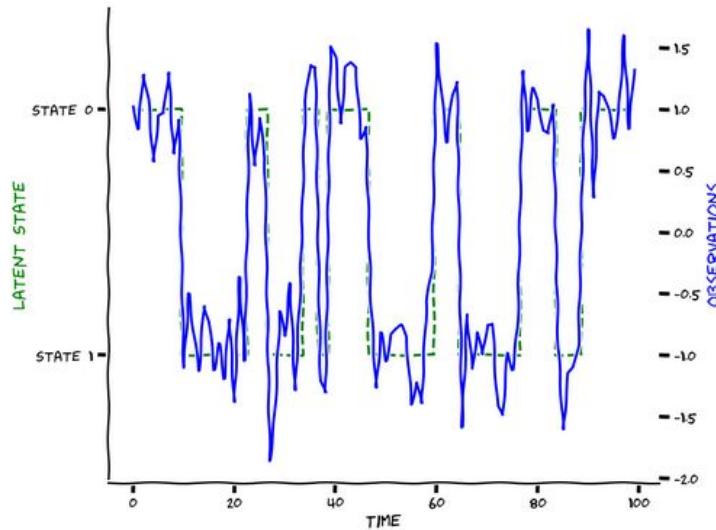
Switching probabilities  $0.1$

noisy end  $0.1$

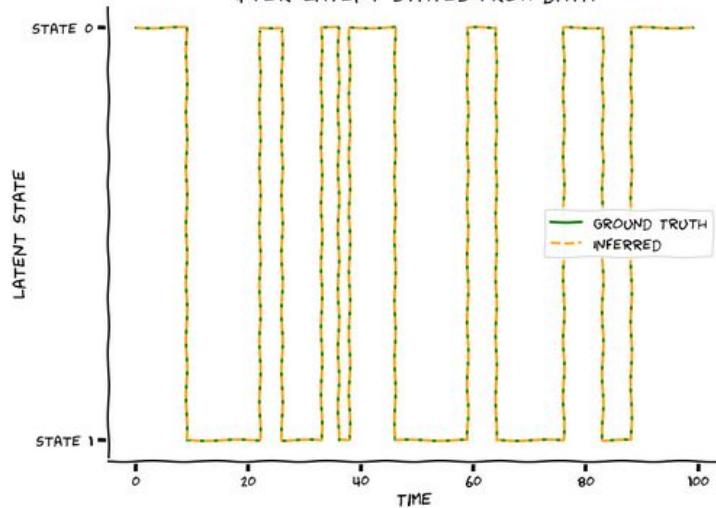
② generate sample sequence along with observations

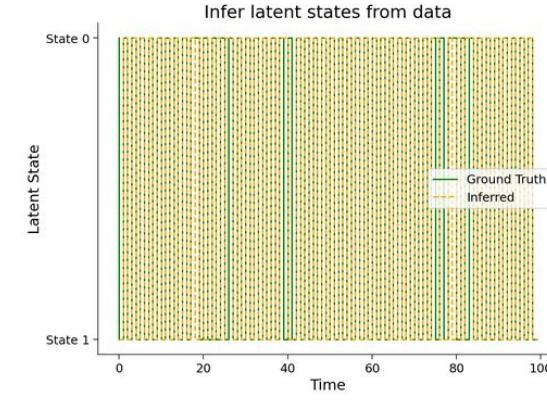
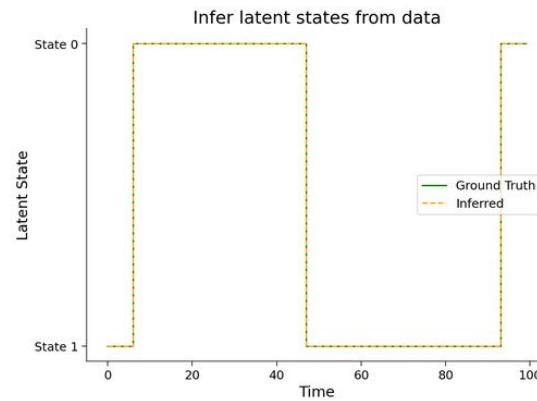
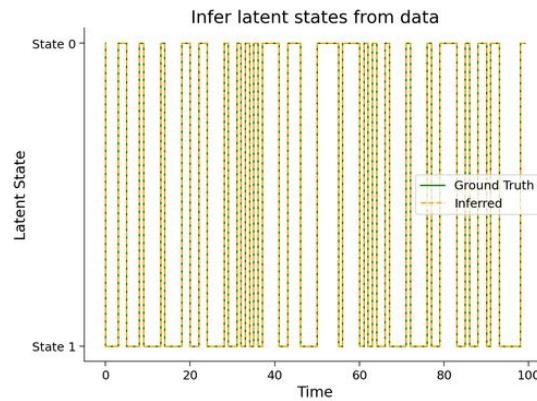
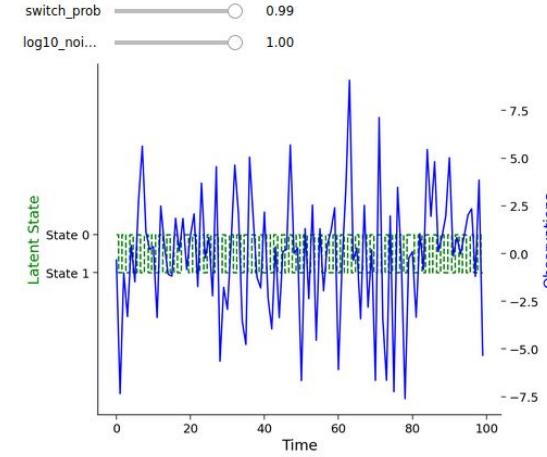
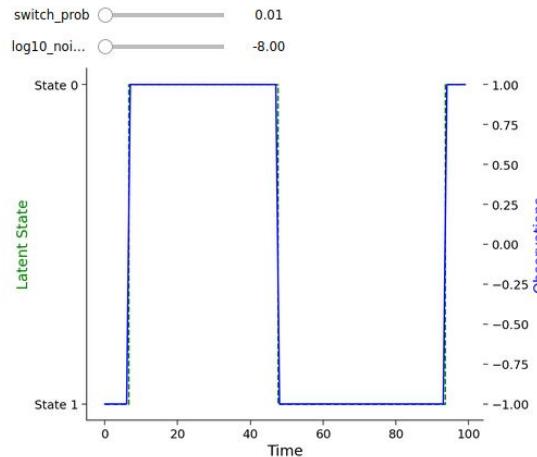
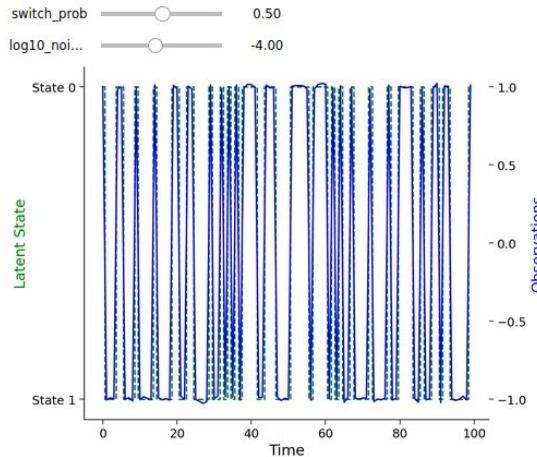
③ calc. posterior probabilities.

④ visualise inferred state sequence  
with ground truth.



## INFER LATENT STATES FROM DATA





# *Tutorial #2 Bonus Explanations*

## Objective

Perform parameter estimation of an HMM using the EM algorithm.

Implementation of an HMM of a network of Poisson spiking neurons:

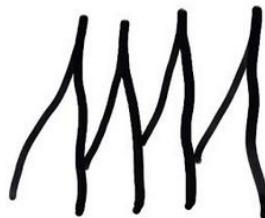
- Implementation of the forward-backward algorithm
- E-step and M-step
- Learning parameters using the EM algorithm
- Intuition of how the EM algorithm monotonically increase data likelihood

(Run for poison spiking neurons case!)

## thalamic relay neurons

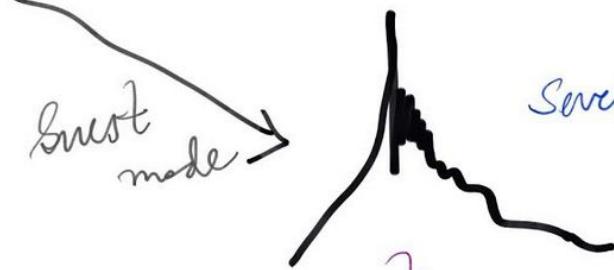
modes differentially  
encode how neurons  
relay information from sensory receptor to cortex

tonic mode



T type calcium channels

burst mode



Several action potentials  
in rapid succession

Statistical approaches let us recover hidden state of Calcium  
Channels freely from spiking activity

problem formulation:

- Network of  $C$  neurons
- Switches between  $K$  states
- neuron  $c$ : firing rate -  $\lambda_i^c$

state transitions :  $K \times K$  transition matrix  $A_{ij}$

initial probability vector  $\pi^0$   
(length:  $K$  at  $t=1$ )

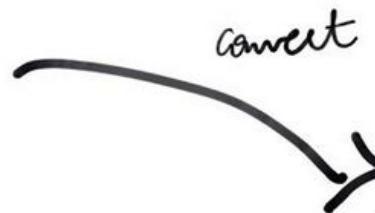
$y_t^c$  = no of spikes for cell  $c$  in time bin  $t$

- ① generate random state sequence from hidden markov chain
- ② generate different trials of spike trains for each cell  
assuming they use the same underlying sequence.

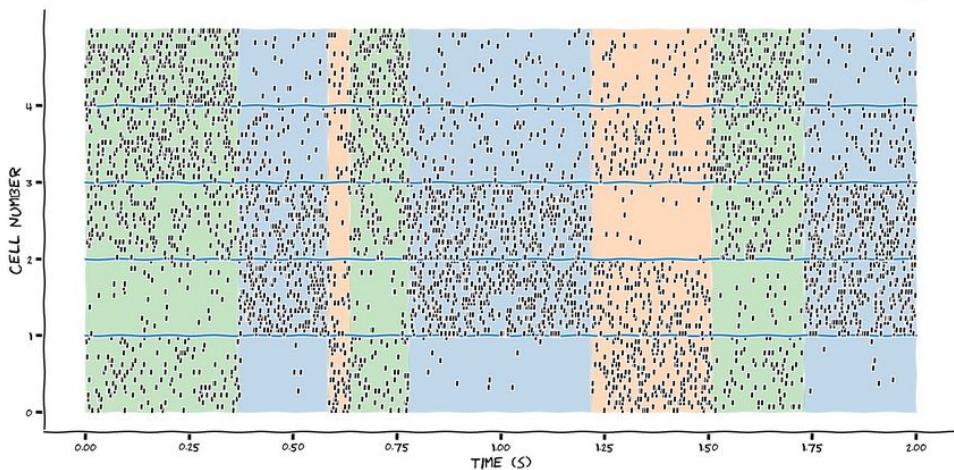
→ Simulate with model / simulation parameters  
initialise the model

→ convert given state sequence to corresponding  
spike rates for all cells at a time  
(visualise all spike trains)

given state sequence



one hot coding  $x_f$   
(sequence of length  $T$   
shape  $(T, K)$ )



## EM Algorithm (for optimisation)

Step #1: generate data

generate n-trials of observations, each one with own randomly generated sequence.

$$y_{\text{shape}} = (\text{n\_trial} = 300, T = 1000, c = 5)$$

find optimal values of parameters that maximizes data likelihood  
→ infeasible to integrate all latent variables  $x_{1:T}$   
(Time needed: Exponential to T)

Alternate solution: EM algorithm

(expectation maximisation algorithm)

data likelihood doesn't decrease after each EM cycle.

Objective -

① recursive equations for  
forward / backward probabilities  
 $\alpha_i(t)$  &  $\beta_i(t)$ .

② Singleton marginal distribution

$$\gamma_i(t) = P_\theta(x_t = i \mid y_{1:T})$$

pairwise marginal distribution

$$\xi_{ij}(t) = P_\theta(x_t = i, x_{t+1} = j \mid y_{1:T})$$

③ closed form solutions of updated values  
to increase data likelihood.

forward backward algorithm

forward pass: calculate forward probabilities  
joint probability of  $x_t$ , current past data  $y_{1:t}$

$$a_i(t) = p(y_t | x_i = t) \sum_j A_{ji} a_j(t-1)$$

backward probabilities: likelihood of observing all future data points given current state  $x_t$

$$b_i(t) = \sum_j p_o(y_{t+1} | x_{t+1} = j) b_j(t+1) A_{ij}$$

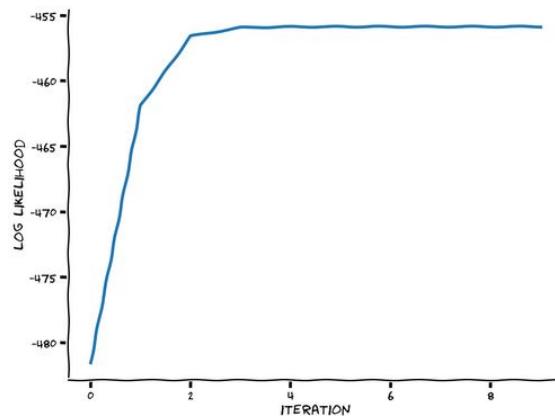
Singletor and pairwise marginal distributions.

$$\gamma_i(t) = p_\theta(x_t=i \mid Y_{1:T}) = \frac{a_i(t)b_i(t)}{p_\theta(Y_{1:T})}$$

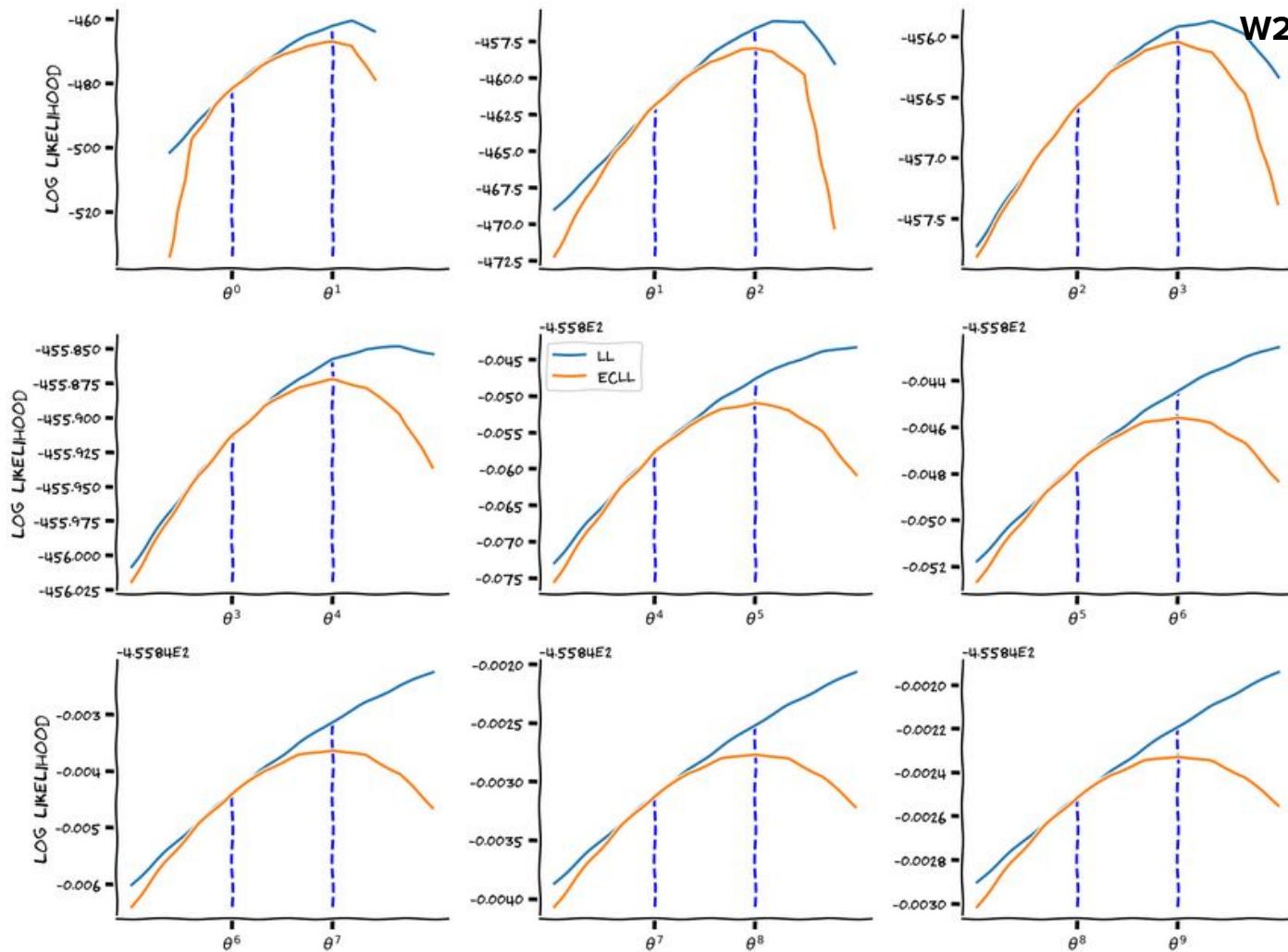
$$\xi_{ij}(t) = p_\theta(x_t=i, x_{t+1}=j \mid Y_{1:T}) = \frac{b_j(t+1)p_\theta(y_{t+1} \mid x_{t+1}=j) \cdot a_i(t)}{p_\theta(Y_{1:T})}$$

where  $p_\theta(Y_{1:T}) = \sum_i a_i(T)$

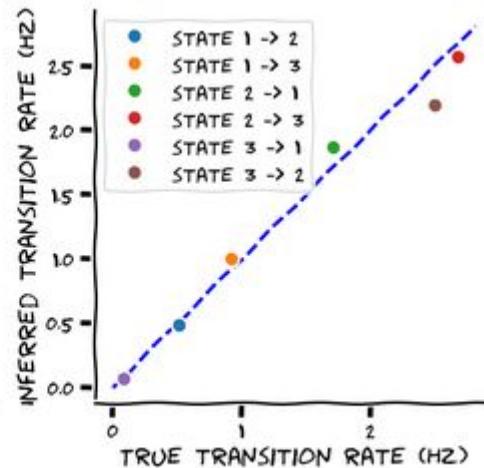
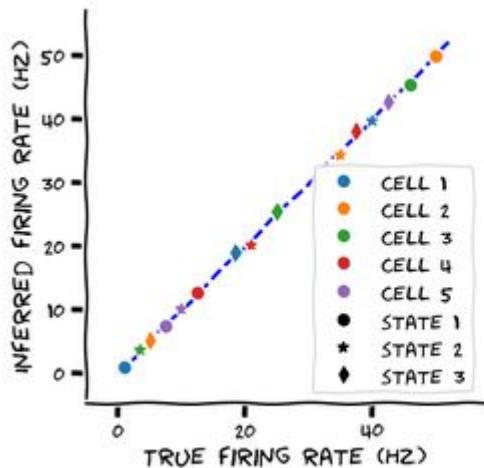
M-step log likelihood versus  
expected complete log  
likelihood (ECLL)



W2D3\_pod 031



plot the (sorted) learnt parameters with true parameters to see if we successfully recovered all the parameters



# *Tutorial #3*

# *Explanations*

# *Linear Dynamic Systems and Kalman filters - Objective*

*Infer a latent model when states are continuous.*

- Review linear dynamical systems
- Implementation of the Kalman filter
- Exploring how the Kalman filter can be used to smooth data from an eye-tracking experiment

latent state variable

$$s_t = F s_{t-1} + G_t \cdot t \quad (D \text{ dimensions})$$

measured/observed variable

$$y_t = H s_t + \eta_t \quad (N \text{ dimensions})$$

by principle of dimension reduction -  $D < N$

Gaussian noise terms -

$$G_t \sim N(0, Q)$$

$$\eta_t \sim N(0, R)$$

$$s_0 \sim N(\mu_0, \Sigma_0)$$

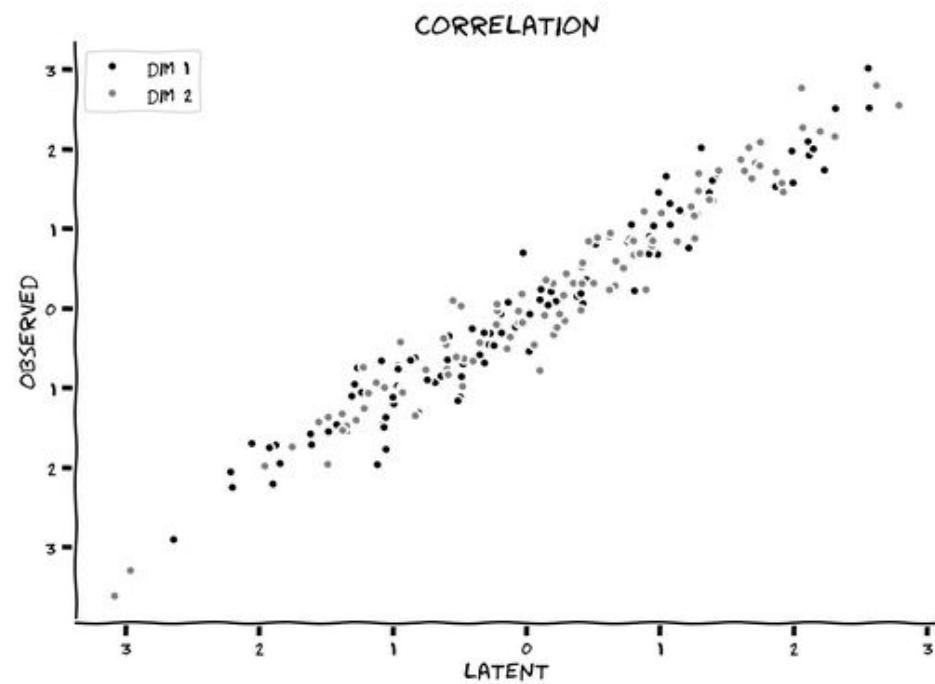
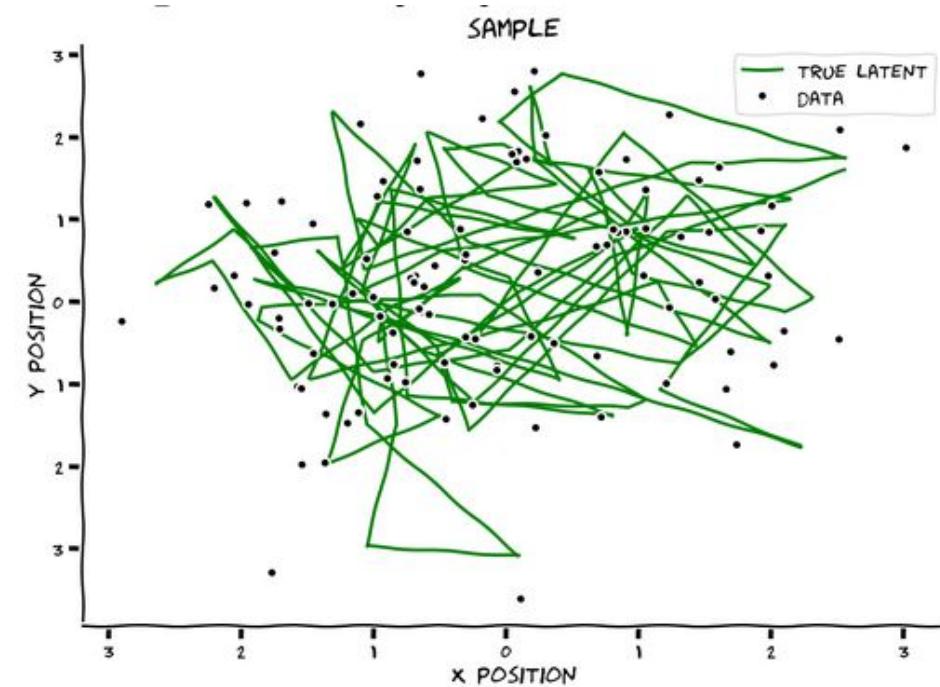
$\Rightarrow s_t$  &  $y_t$  are gaussian  
markov chain: state at  $t$  is conditionally independent  
given  $t^{-1}$

# Sampling

```
# task dimensions
n_dim_state = 2
n_dim_obs = 2

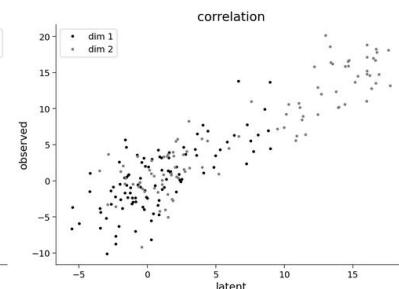
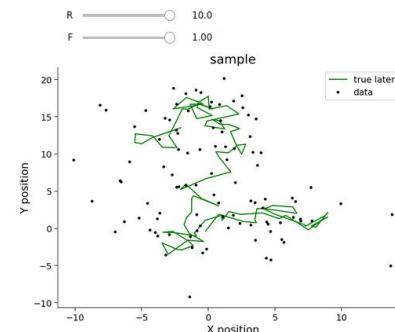
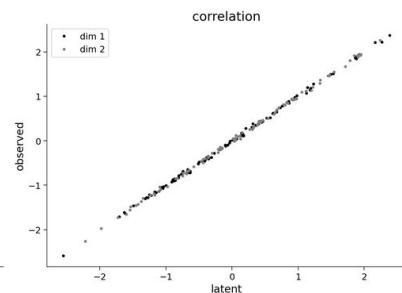
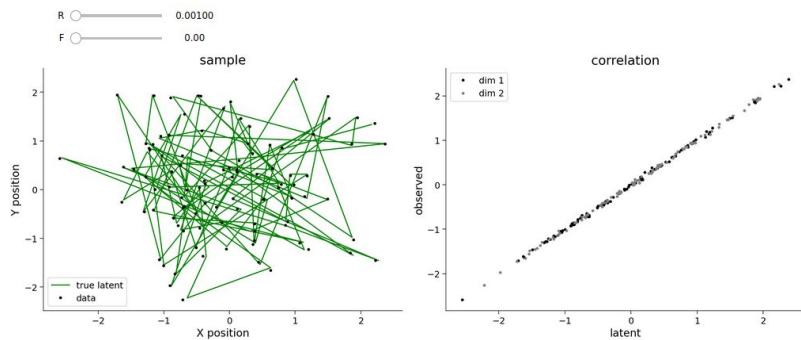
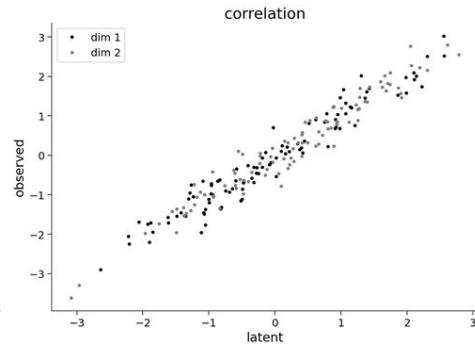
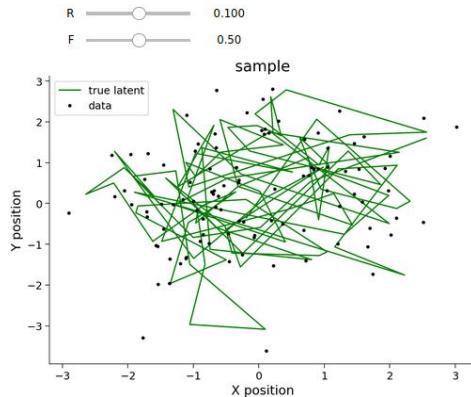
# initialize model parameters
params = {
    'F': 0.5 * np.eye(n_dim_state),  # state transition matrix
    'Q': np.eye(n_dim_obs),  # state noise covariance
    'H': np.eye(n_dim_state),  # observation matrix
    'R': 0.1 * np.eye(n_dim_obs),  # observation noise covariance
    'mu_0': np.zeros(n_dim_state),  # initial state mean
    'sigma_0': 0.1 * np.eye(n_dim_state),  # initial state noise covariance
}
```

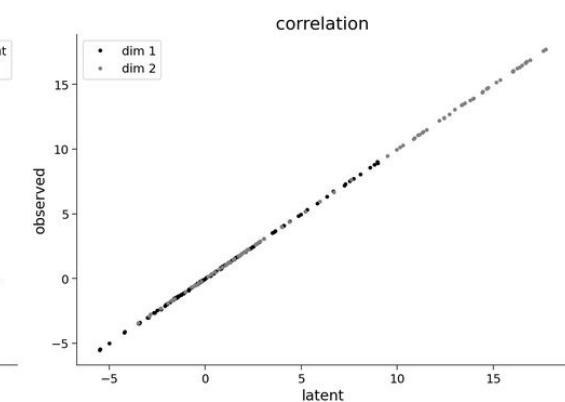
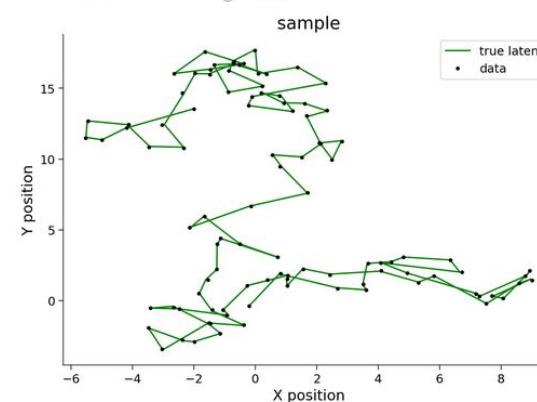
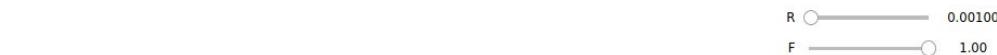
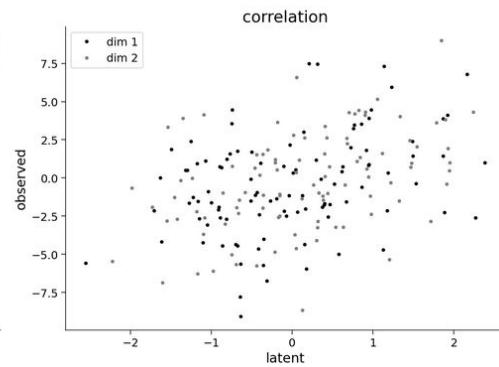
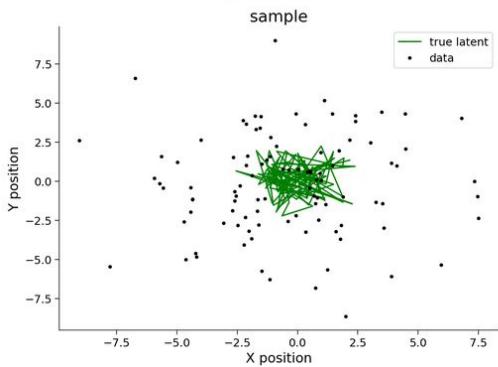
*Note:* Parameter dictionary (`params`) is used but as the number of parameters increases, it can be beneficial to condense them into a data structure. The trade-off is that we have to know what is in the data structure to use the values, rather than function signature.



## Adjusting System Dynamics

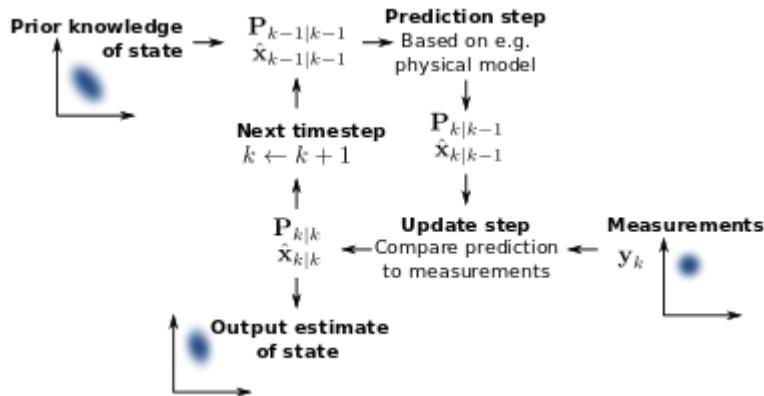
parameters of a linear dynamical system: Reduce observation noise  $R$  and respective temporal dynamics  $F$





The Kalman filter keeps track of the estimated state of the system and the variance or uncertainty of the estimate. The estimate is updated using a state transition model and measurements

In statistics and control theory, Kalman filtering is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe.



Obtain estimates of latent state by running filtering 0 to n.

$$s_t^{\text{pred}} \sim N(\hat{u}_t^{\text{pred}}, \hat{\Sigma}_t^{\text{pred}})$$

$$\begin{aligned} \hat{u}_1^{\text{pred}} &= F \hat{u}_0 \\ \hat{u}_t^{\text{pred}} &= F \hat{u}_{t-1} \end{aligned} \quad \left. \right\}$$

Prediction for  $s_t$  obtained by taking expected value of  $s_{t-1}$  and projecting it forward one step using transition probability matrix A

$$\hat{\Sigma}_0^{\text{pred}} = F \hat{\Sigma}_0 F^T + Q$$

noise covariance

$$\hat{\Sigma}_t^{\text{pred}} = F \hat{\Sigma}_{t-1} F^T + Q.$$

Update from observation to obtain  $\hat{\mu}_t^{\text{filter}}$  &  $\hat{\Sigma}_t^{\text{filter}}$

project to observational space

$$y_t^{\text{pred}} \sim N(H\hat{\mu}_t^{\text{pred}}, H\hat{\Sigma}_t^{\text{pred}}H^T + R)$$

update prediction by actual data

$$s_t^{\text{filter}} \sim N(\hat{\mu}_t^{\text{filter}}, \hat{\Sigma}_t^{\text{filter}})$$

$$\hat{\mu}_t^{\text{filter}} = \hat{\mu}_t^{\text{pred}} + K_t (y_t - H\hat{\mu}_t^{\text{pred}})$$

$$\hat{\Sigma}_t^{\text{filter}} = (I - K_t H) \hat{\Sigma}_t^{\text{pred}}$$

Kalman gain matrix

$$K_t = \hat{\Sigma}_t^{\text{pred}} H^+ \left( H \hat{\Sigma}_t^{\text{pred}} H^T + R \right)^{-1}$$

use only latent only prediction }  
to project to observation space }

Compute correction  $\times y_t - H F \hat{x}_{t-1}$  } prediction & data

coefficient of correction = Kalman gain matrix

Note: measurement noise = small & dynamics = fast }  
estimation depends on observed  
data

Gaussian Noise terms.

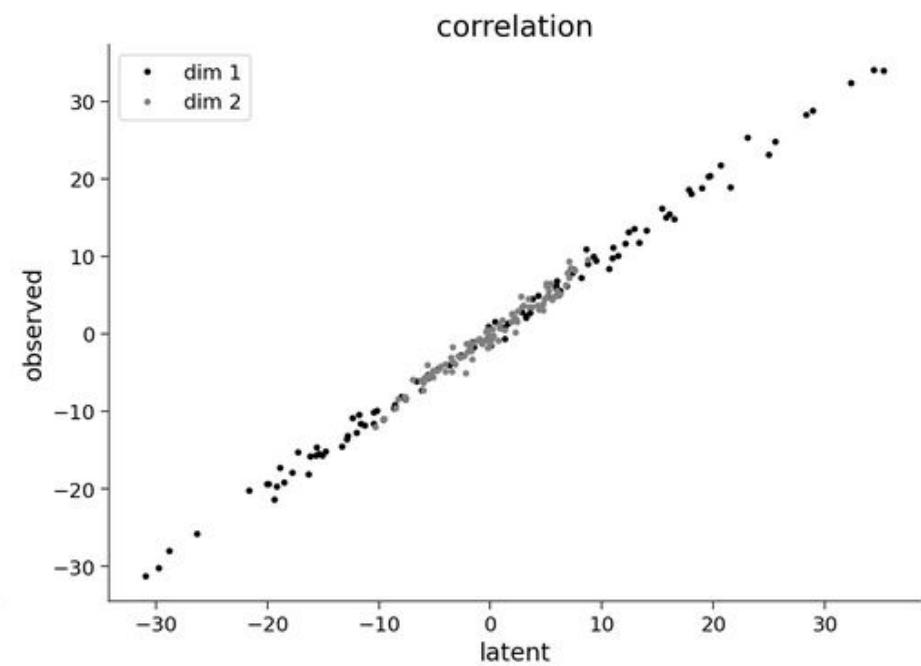
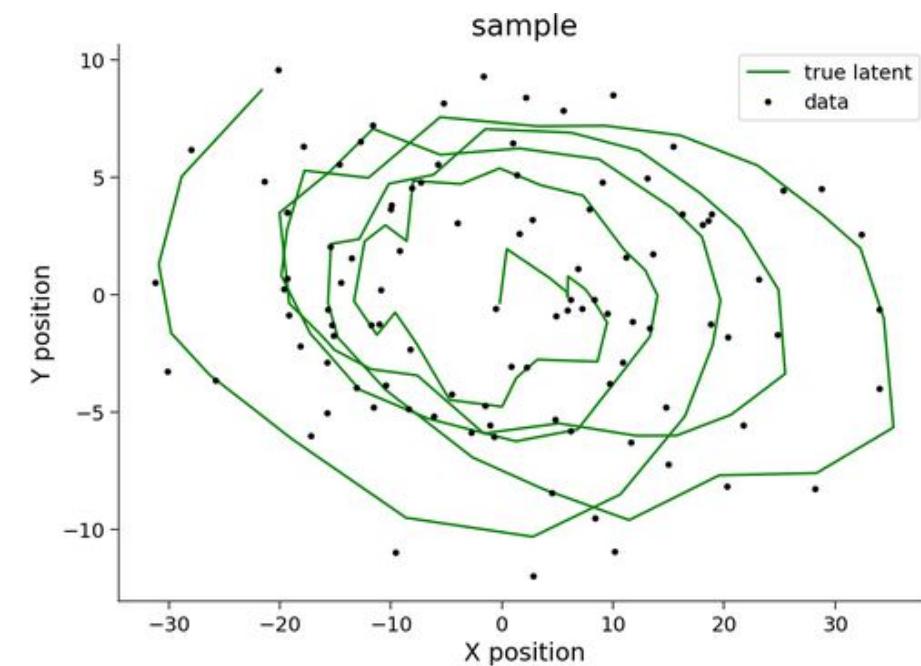
$$\xi_t \sim N(0, Q)$$

state noise covariance

$$\eta_t \sim N(0, R)$$

acceleration noise covariance

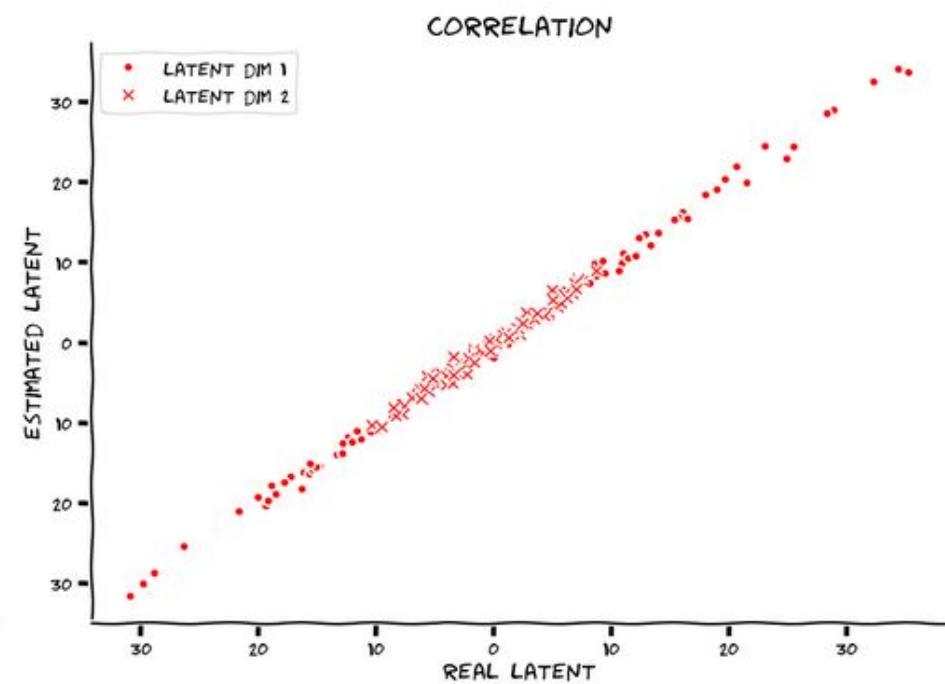
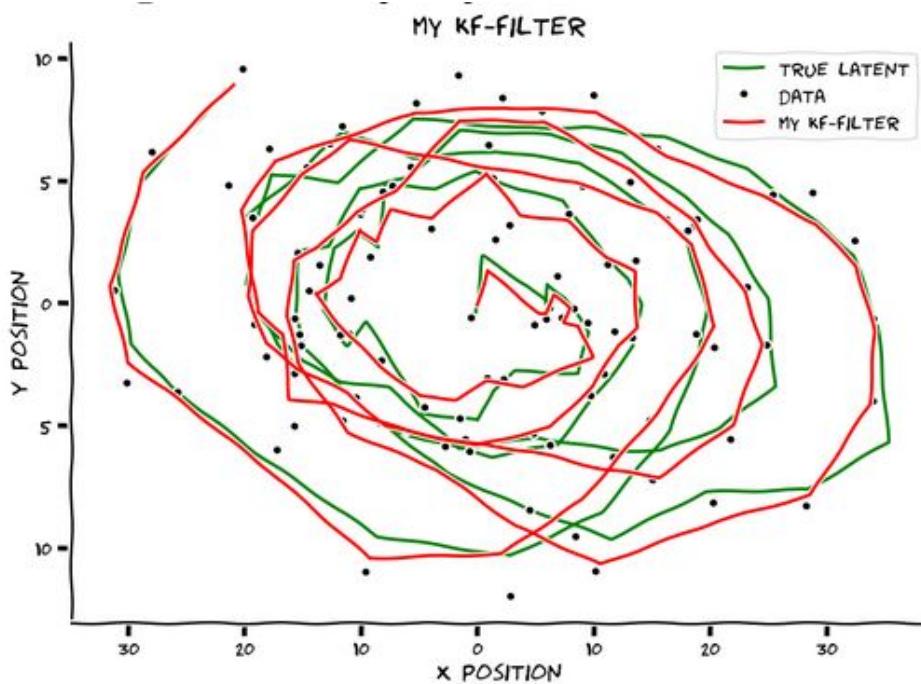
$$S_0 \sim N(\mu_0, \Sigma_0)$$



# Implement kalman filter (forward process)

Kalman gain  $K$   
filter mean  $\mu$   
filter covariance  $\Sigma$   
@ each time step

State tracking array  
↓  
filtering  
↓  
 $K, \mu, \Sigma$



# Drawing Parallels with Biological systems

Kalman filters also are one of the main topics in the field of robotic motion planning and control, and they are sometimes included in trajectory optimization. The Kalman filter also works for modeling the central nervous system's control of movement. Due to the time delay between issuing motor commands and receiving sensory feedback, use of the Kalman filter supports a realistic model for making estimates of the current state of the motor system and issuing updated commands

# *Applications - Kalman filter*

*Tracking eye gaze to get an accurate estimation of where someone is looking on a screen in pixel coordinates. This can be challenging due to the various sources of noise inherent in obtaining these measurements - general accuracy of the eye tracker device, maintenance of calibration over time, changes in ambient light or subject position, eye blinks (interruptions in the data stream).*

subject\_id  -1

image\_id  0



subject\_id  4

image\_id  2



subject\_id  2

image\_id  1



**W2D3\_pod 031**

Use Kalman filtering to give us a better estimate of the true gaze directly from the data with the EM algorithm.

### Estimation with Kalman (model fitting)

- ① provide guess at dimensionality of latent state  
(assume dynamics line up with observation data)  
( $x, y$ ) coordinates
- ② let EM discover the dynamics of:  
transition / observation matrices &  
transition / observation covariance.



Subjects are fixated centre of screen  
 $\Rightarrow$  start state estimate  $m_0$   
+ low initial noise covariance  $\Sigma_0$

F =

```
[[ 1.004 -0.01 ]  
 [ 0.005 0.989]]
```

Q =

```
[[278.016 219.292]  
 [219.292 389.774]]
```

H =

```
[[ 0.999 0.003]  
 [-0.004 1.01 ]]
```

R =

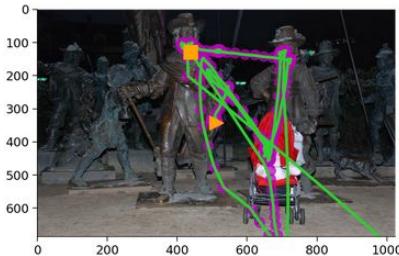
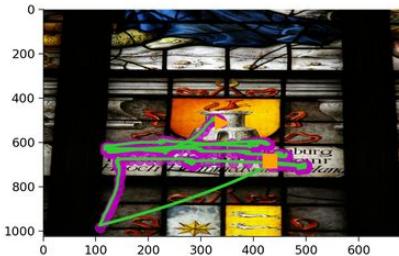
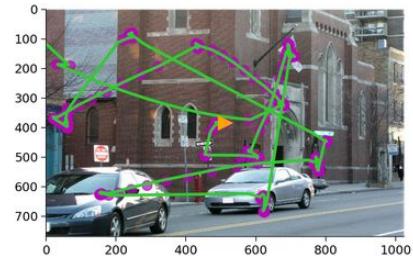
```
[[26.026 19.596]  
 [19.596 26.745]]
```

both the state and observation matrices are close to the identity matrix, which means the x- and y-coordinate dynamics are independent of each other and primarily impacted by the noise covariances.

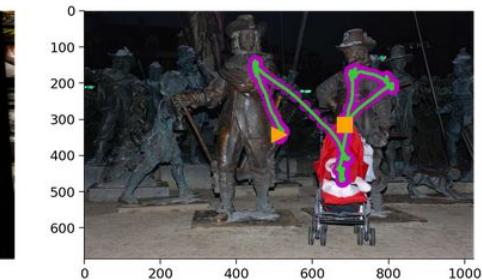
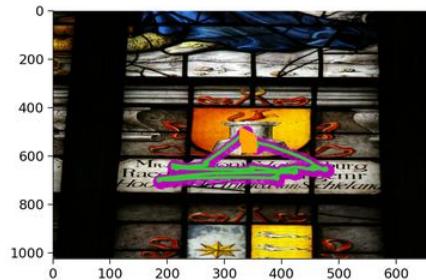
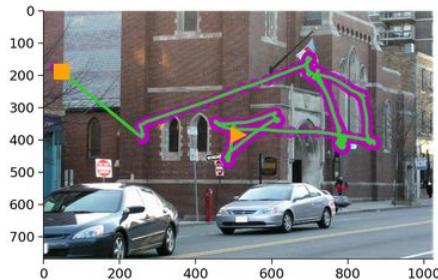
use this model to smooth the observed data from the subject. In addition to the source image, see how this model works with the gaze recorded by the same subject on the other images as well, or even with different subjects.

subject\_id 

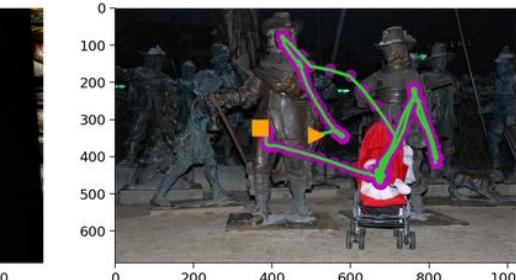
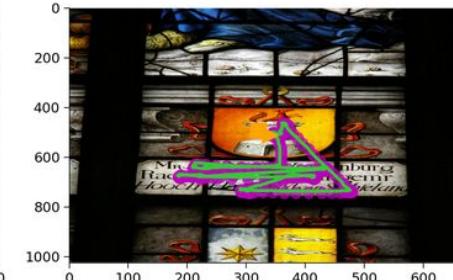
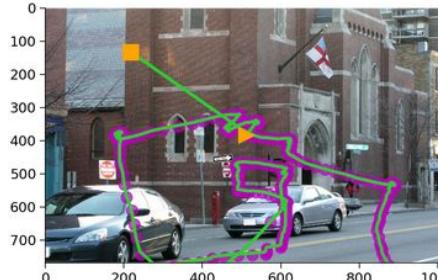
1

subject\_id 

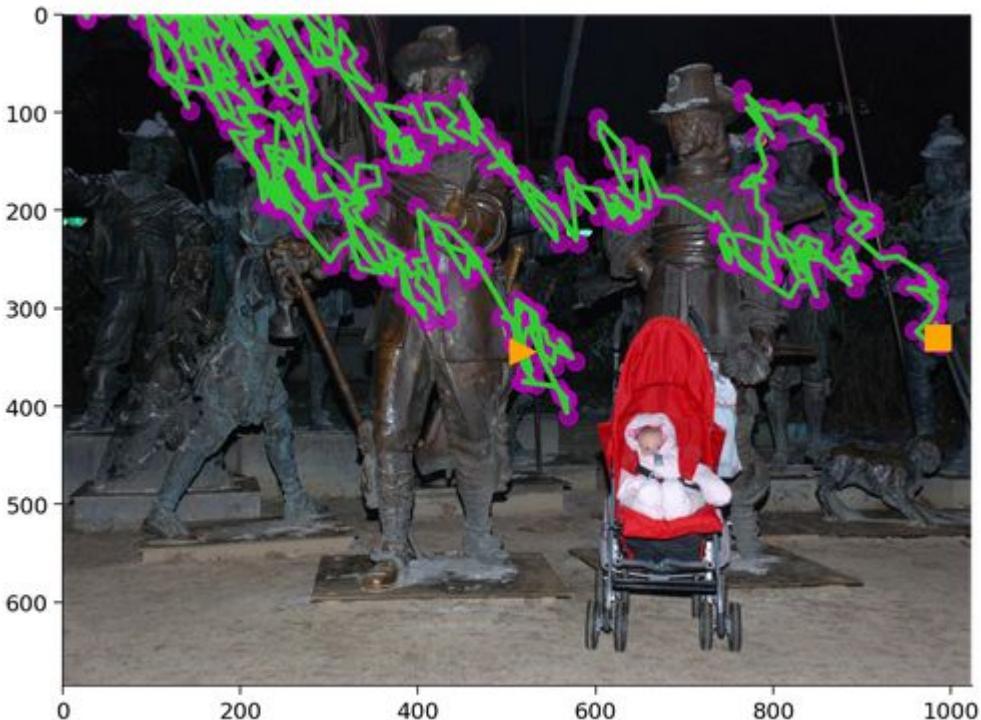
0

subject\_id 

4



Original task was to use this data to help develop models of visual salience. While our Kalman filter is able to provide smooth estimates of observed gaze data, it's not telling us anything about *why* the gaze is going in a certain direction. In fact, if we sample data from our parameters and plot them, we get what amounts to a random walk.



# *Observations*

The model is given no other observed data beyond the pixels at which gaze was detected. We expect some other aspect to be driving the latent state of where to look next.

# Summary

In summary, while the Kalman filter is a good option for smoothing the gaze trajectory itself, especially if using a lower-quality eye tracker or in noisy environmental conditions, a linear dynamical system may not be the right way to approach the much more challenging task of modeling visual saliency.

# *Tutorial #3 Bonus Explanations*

Review Gaussian joint, marginal and conditional distributions.

Assume:

$$\mathbf{z} = [x^T \ y^T]^T$$

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

Marginal distributions:

$$x \sim N(a, A)$$

$$y \sim N(b, B)$$

Conditional distributions:

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

$$y|x \sim N\left(b + C^TA^{-1}(x - a), B - C^TA^{-1}C\right)$$

# Kalman Smoothing

Obtain estimates by propagating from  $y_T$  back to  $y_0$   
 using results of forward pass ( $\hat{\mu}_t^{\text{filter}}$ ,  $\hat{\Sigma}_t^{\text{filter}}$ ,  $P_t = \hat{\Sigma}_{t+1}^{\text{pred}}$ )

$$s_t \sim N(\hat{\mu}_t^{\text{smooth}}, \hat{\Sigma}_t^{\text{smooth}})$$

$$\hat{\mu}_t^{\text{smooth}} \sim \hat{\mu}_t^{\text{filter}} + J_t (\hat{\mu}_{t+1}^{\text{smooth}} - F \hat{\mu}_t^{\text{filter}})$$

$$\hat{\Sigma}_t^{\text{smooth}} = \hat{\Sigma}_t^{\text{filter}} + J_t (\hat{\Sigma}_{t+1}^{\text{smooth}} - P_t) J_t^T$$

$$J_t = \hat{\Sigma}_t^{\text{filter}} F_t^{T-1} P_t^{-1}$$

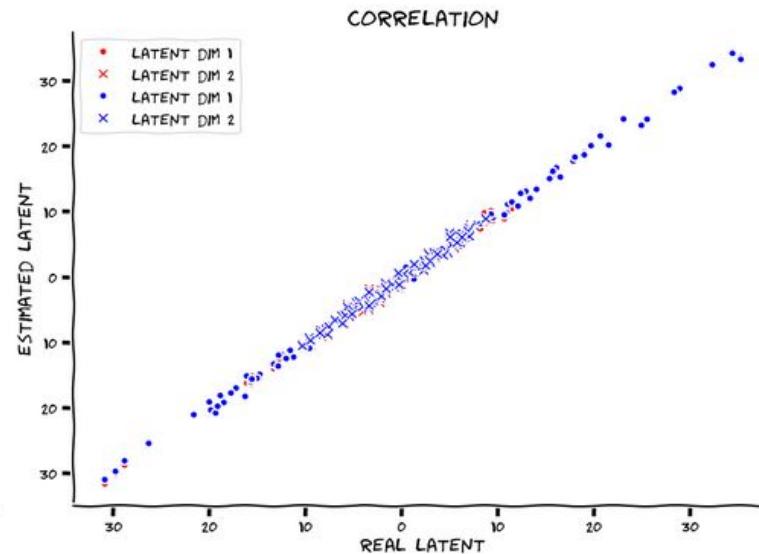
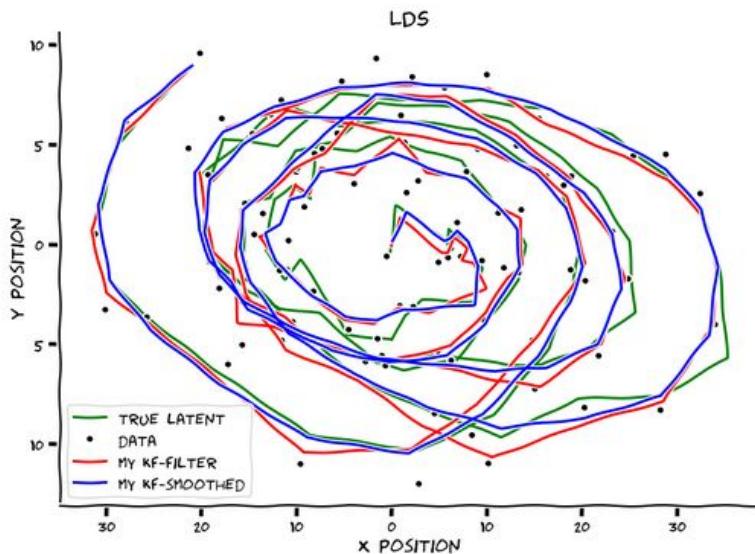
final estimate for  $z_t$

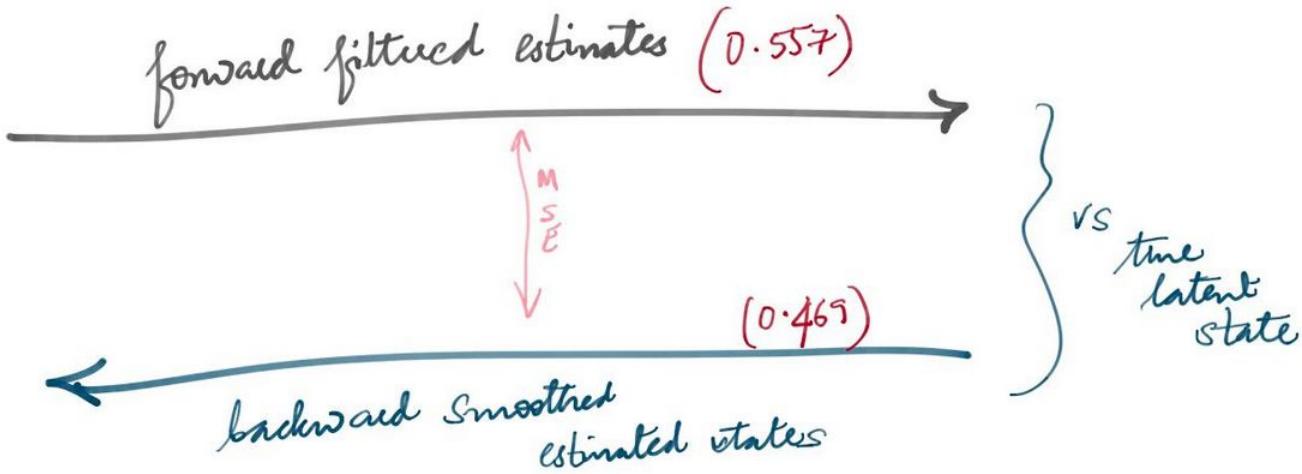
$$\hat{\mu}_t = \hat{\mu}_t^{\text{smooth}}$$

$$\hat{\Sigma}_t = \hat{\Sigma}_t^{\text{smooth}}$$

implement Kalman smoothing (backward) process

• compute smoothed mean / covariance &  $\hat{S}_f$  values



forward vs backward

Smoothed better than filtered as backward pass can use forward pass estimates & correct them given collected data.

*Use Kalman filtering alone, without smoothing:* As Kalman filtering only depends on already observed data (i.e. the past) it can be run in a streaming, or on-line, setting. Kalman smoothing relies on future data as it were, and as such can only be applied in a batch, or off-line, setting.

So,

*Kalman filtering for real-time corrections and  
Kalman smoothing for already-collected data.*

EM algorithm

maximise  $\log p(y|\theta)$

marginalise latent state (not tractable)

$$f(y|\theta) = \int p(y, s|\theta) dy$$

add probability distribution  $q(s)$  which  
approximate latent state distribution

$$\log p(y|\theta) - \int_s q(s) dy$$

joint distribution of  $y$  and  $s$ .

$$L(q, \theta) + KL(q(s) || f(s|y), \theta)$$

conditional distribution of  $s|y$

## Expectation step (Kalman filter + smoother)

- fixed parameters
  - good approximation
- $q(s) : \max_{\alpha} \text{lower bound } L(\alpha, \theta)$   
with respect to  $q(s)$

## Maximisation step

- parameters : fixed
- change parameters to max lower bound  
 $L(q, \theta)$

M step

posterior marginals obtained from smoothing

$$\left\{ \begin{array}{l} \hat{\mu}_t^{\text{smooth}} \\ \hat{\Sigma}_t^{\text{smooth}} \end{array} \right.$$

$$E(s_t) = \hat{\mu}_t$$

$$E(s_t s_{t-1}^T) = J_{t-1} \hat{\Sigma}_t + \hat{\mu}_t \hat{\mu}_{t-1}^T$$

$$E(s_t s_t^T) = \hat{\Sigma}_t + \hat{\mu}_t \hat{\mu}_t^T$$

parameter update

$$\text{initial : } \mu_0^{\text{new}} = E(s)$$

$$Q_0^{\text{new}} = E(s_0 s_0^T) - E(s_0) E(s_0^T)$$

hidden / latent state

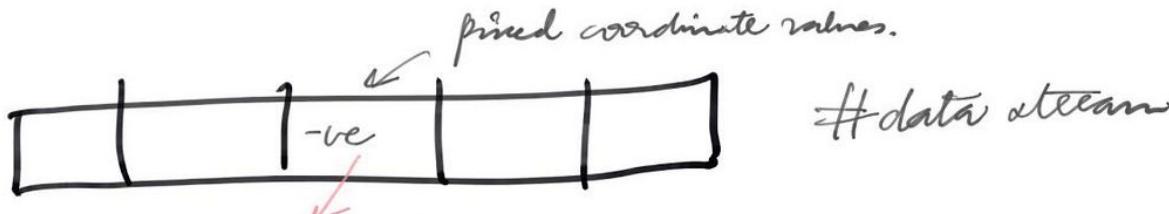
$$F^{\text{new}} = \left( \sum_{t=2}^N E(s_t s_{t-1}^T) \right) \left( \sum_{t=2}^N E(s_{t-1} s_{t-1}^T) \right)^{-1}$$

$$\hat{Q}^{\text{new}} = \frac{1}{T-1} \sum_{t=2}^N E(s_t s_t^T) - F^{\text{new}} E(s_{t-1} s_{t-1}^T) \\ - E(s_t s_{t-1}^T) F^{\text{new}} + \left( F^{\text{new}} E(s_{t-1} s_{t-1}^T) \cdot (F^{\text{new}})^T \right)$$

Observed/measured space

$$H^{\text{new}} = \left( \sum_{t=1}^N y_t E(s_t^\top) \right) \left( \sum_{t=1}^N E(s_t s_t^\top) \right)^{-1}$$

$$R^{\text{new}} = \frac{1}{T} \sum_{t=1}^N y_t y_t^\top - H^{\text{new}} E(s_t) y_t^\top - y_t E(s_t^\top) H^{\text{new}} \\ + H^{\text{new}} E(s_t s_t^\top) H^{\text{new}}$$



### issue mitigation

- ① delete values.

con: integrity of consistent time step between samples: lost

- ② masked arrays

additional embedded boolean masking array that indicates which elements should be masked (ignored while processing)

preferred {