Welcome!
#pod-031
Week #2, Day 3

(Reviewed by: Deepak)

neuromatch academy

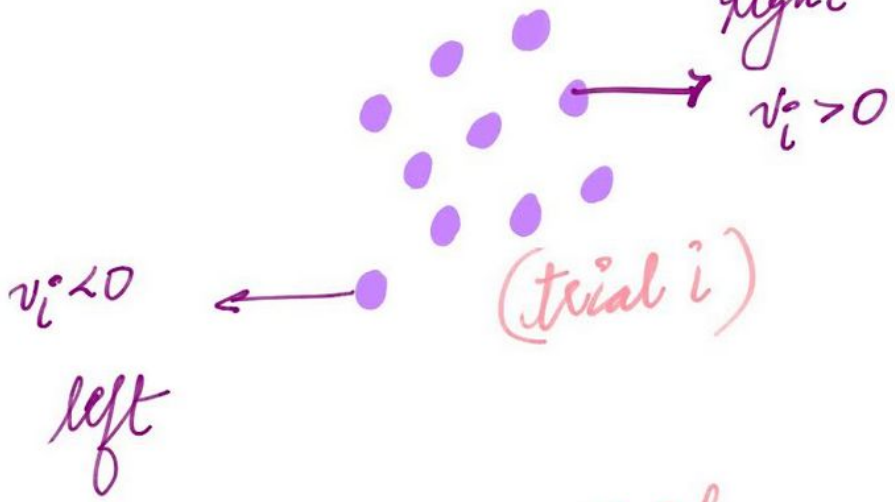
facebook Reality Labs

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



$P_L\{\mathcal{N}(-1, \sigma^2)$

accepting hypothesis H_L

$P_R\{\mathcal{N}(+1, \sigma^2)$

accepting hypothesis H_R

Set of all v_i are generated by -

true / data generating / fixed probability distribution

P_L / P_R : 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.

Why log likelihood?

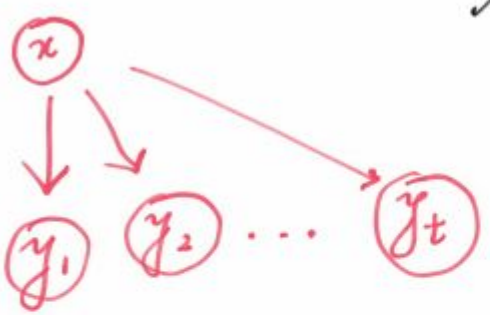
In statistics, the **likelihood-ratio test** assesses the goodness of fit of two competing statistical models based on the ratio of their likelihoods,

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_1

or accepting hypothesis H_2.

choose more evidence before making a decision.



receive sequence of independent samples from distribution f

$f \in \{f_0, f_1\}$

determined by binary latent variable x & need to test b/w

$H_L : f = f_L$ if $x = 0$

$H_R : f = f_R$ if $x = 1$

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

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 $S_n > 0$

H_L if $S_n < 0$

accept H_R with probability $1/2$ 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 α

accept hypothesis H_R if $S_n \geq b$

H_L if $S_n \leq a$

thresholds determined by -

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

$(a = -b)$

As a drift diffusion model –

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

$$p_L(x \mid z=0) = \mathcal{N}(\mu_L, \sigma_L^2)$$

$$p_R(x \mid z=1) = \mathcal{N}(\mu_R, \sigma_R^2)$$

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 $\mu_R + \sigma_R \varepsilon$ ← std gaussian

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

drift part

diffusion part

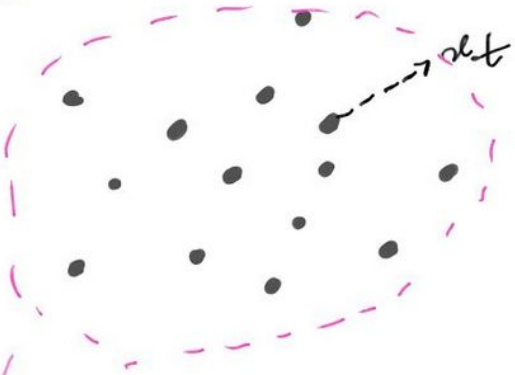
let $\sigma_L = \sigma_R$

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

$$\log \Lambda_i = 0.5 \frac{(\mu_R - \mu_L)^2}{\sigma_L^2} + \frac{\mu_R - \mu_L}{\sigma_L^2} \varepsilon$$

$$\varepsilon \sim \mathcal{N}(0,1)$$

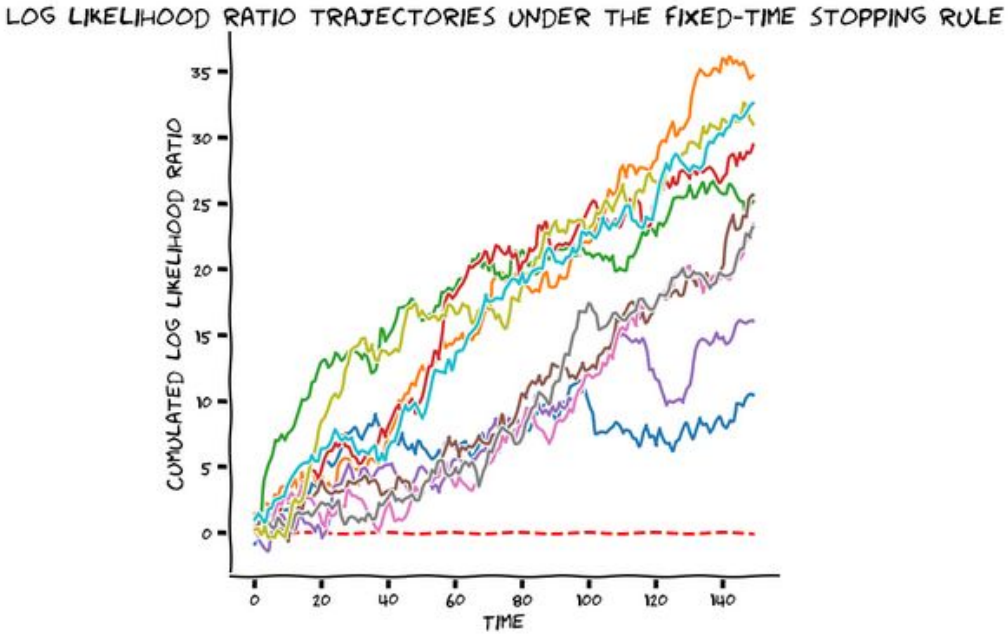
random dot motion task.

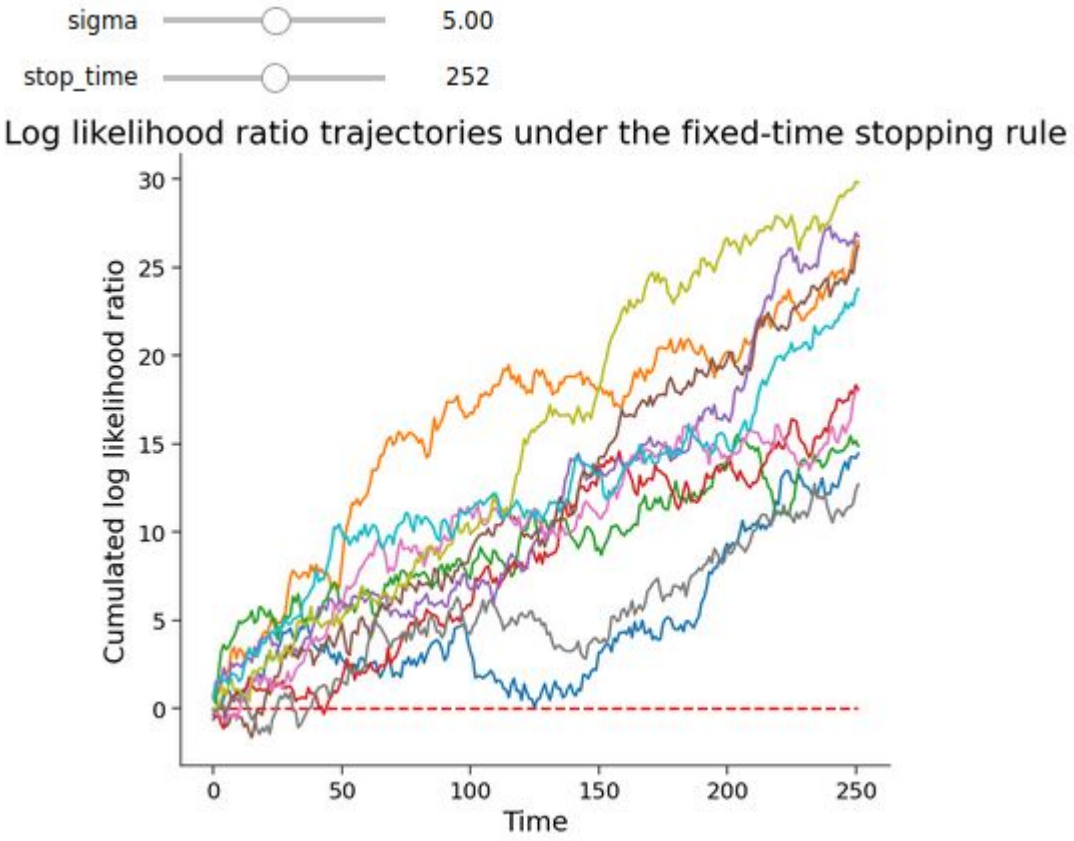


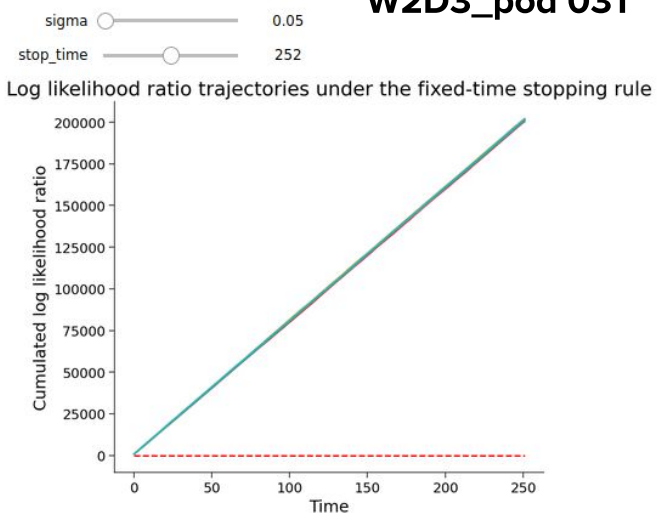
distribution

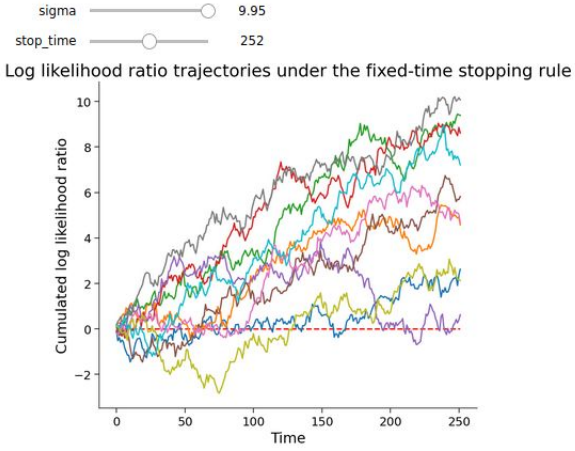
$p_L = N(-\mu, \sigma^2)$

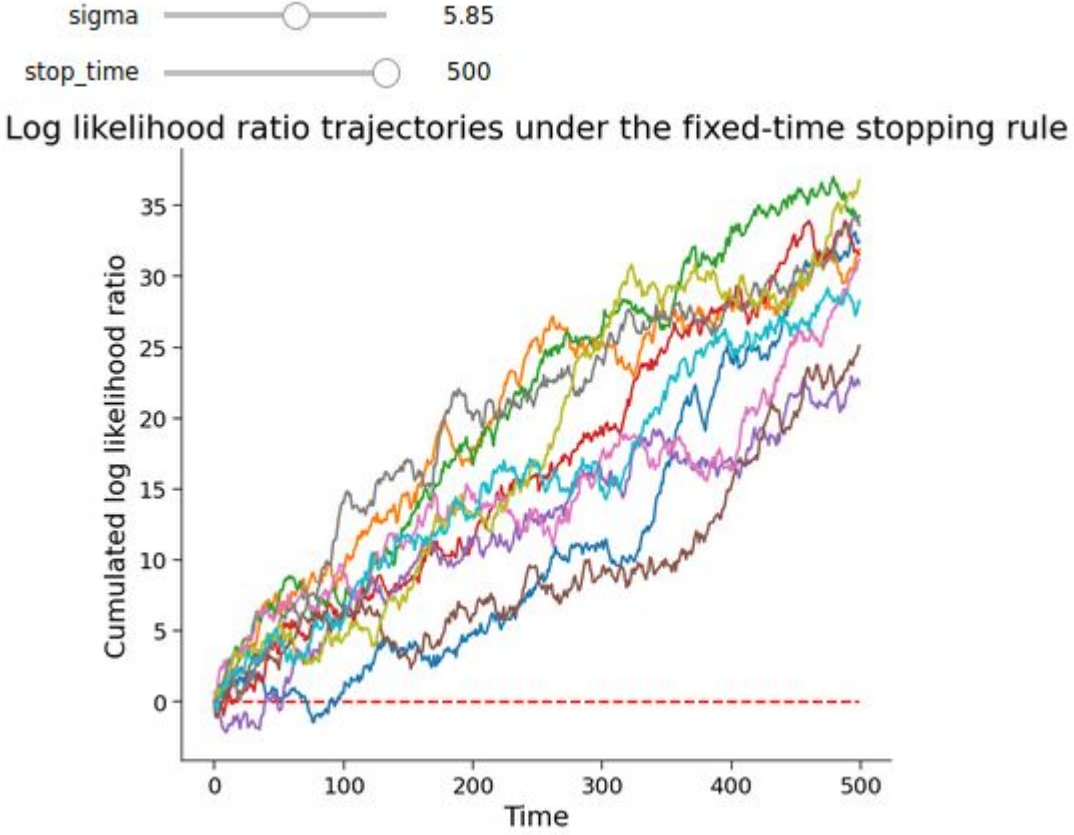
$p_R = N(\mu, \sigma^2)$

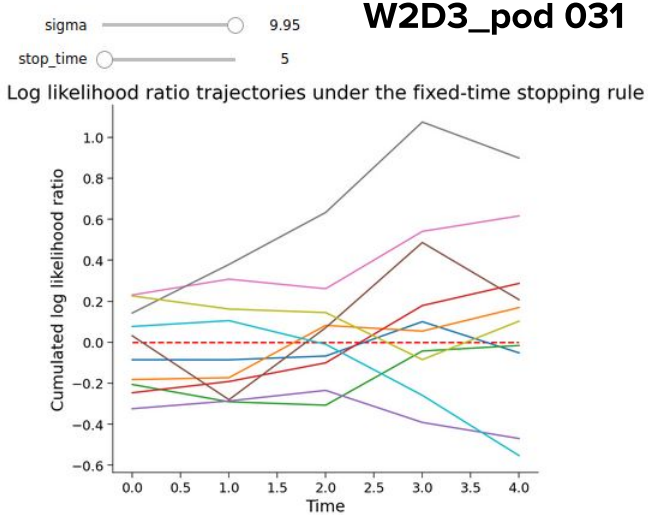


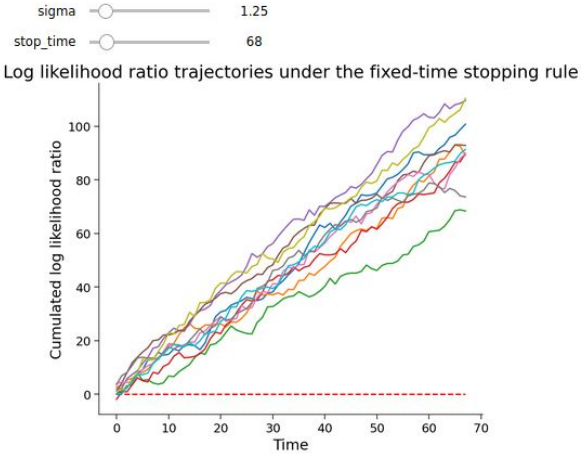

sigma
5.00
stop_time
252
Log likelihood ratio trajectories under the fixed-time stopping rule
Cumulated log likelihood ratio
Time


sigma
0.05
stop_time
252
Log likelihood ratio trajectories under the fixed-time stopping rule
Cumulated log likelihood ratio
Time


sigma
9.95
stop_time
252
Log likelihood ratio trajectories under the fixed-time stopping rule
Cumulated log likelihood ratio
Time


sigma
5.85
stop_time
500
Log likelihood ratio trajectories under the fixed-time stopping rule
Cumulated log likelihood ratio
Time


sigma
9.95
stop_time
5
Log likelihood ratio trajectories under the fixed-time stopping rule
Cumulated log likelihood ratio
Time

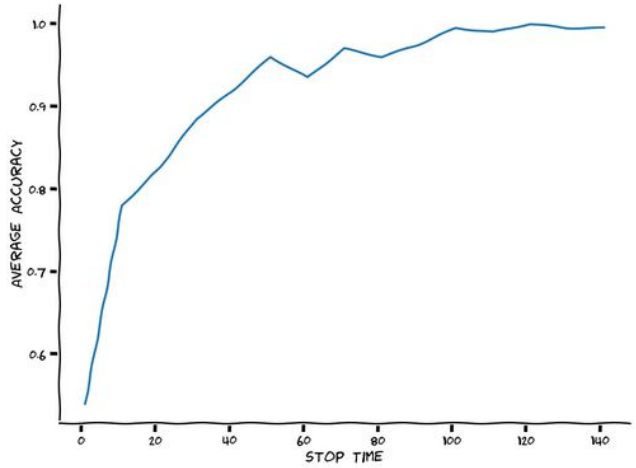

sigma
1.25
stop_time
68
Log likelihood ratio trajectories under the fixed-time stopping rule
Cumulated log likelihood ratio
Time

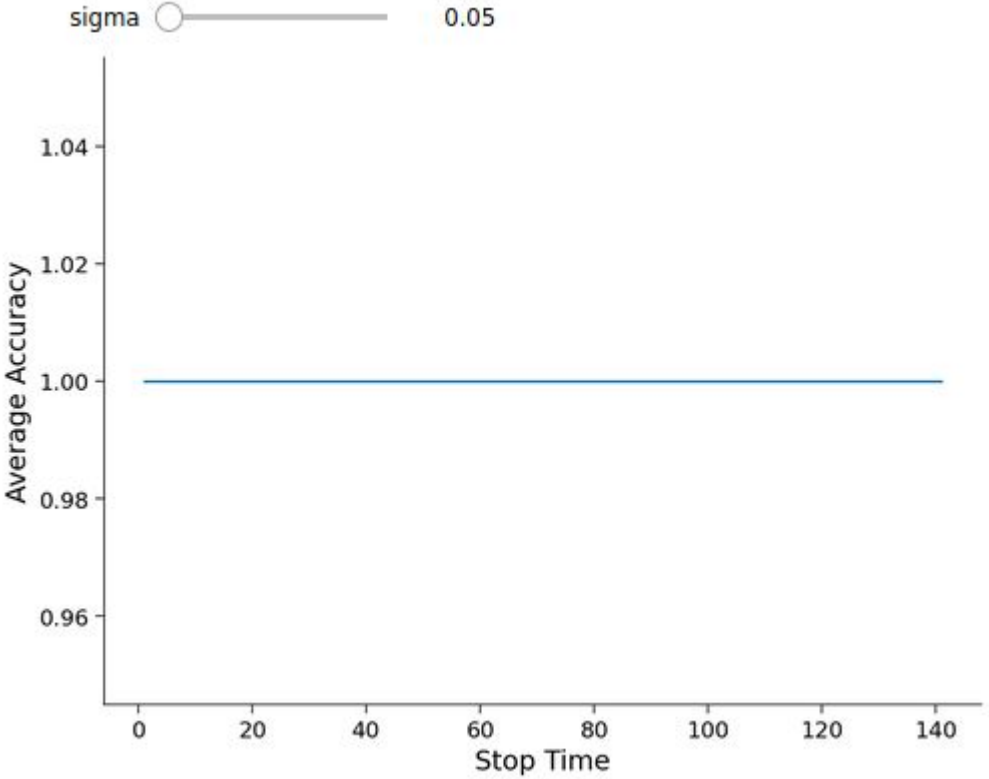
Observations

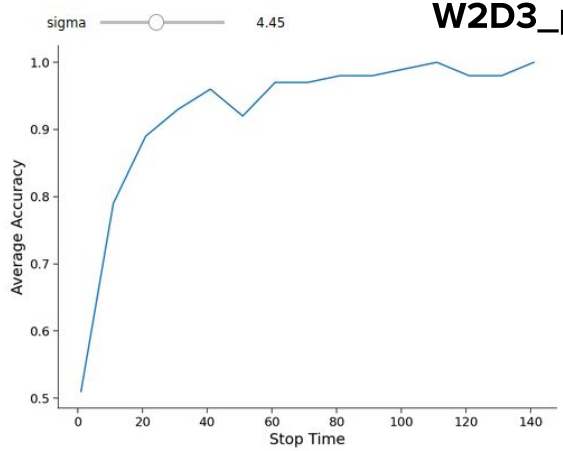
If you stop taking samples too early, (e.g., make a decision after only seeing 5 samples), or there's a huge amount of observation noise that buries the signal, you are likely to be driven by observation noise to a negative cumulated log likelihood ratio and thus make a wrong decision.

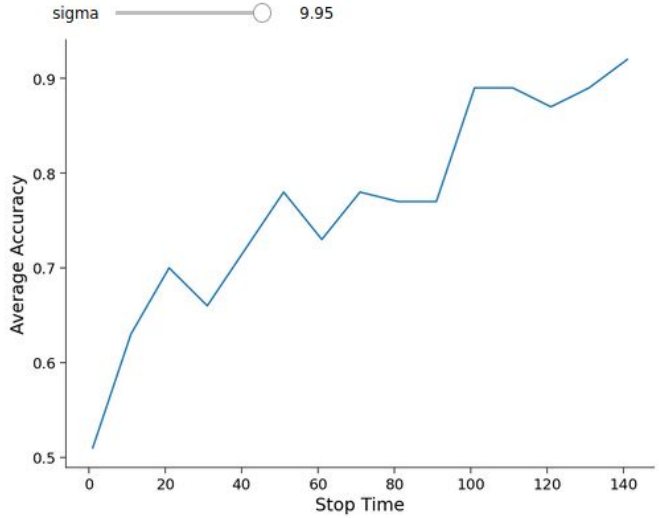
Accuracy: proportion of correct trials across repeated simulations.

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




sigma
0.05
Average Accuracy
1.04
1.02
1.00
0.98
0.96
0
20
40
60
80
100
120
140
Stop Time


sigma
4.45
Average Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100
120
140
Stop Time


sigma
9.95
Average Accuracy
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100
120
140
Stop Time

Simulating drift diffusion model with fixed thresholds

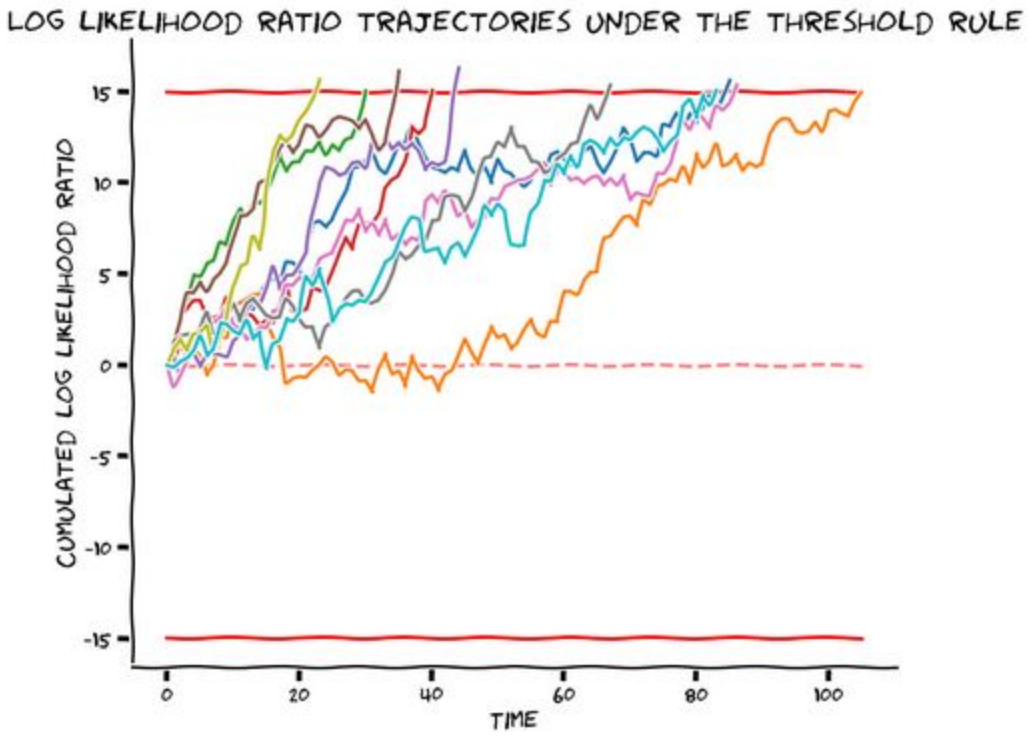
- define desired error rate

(make measurements until error rate is reached)

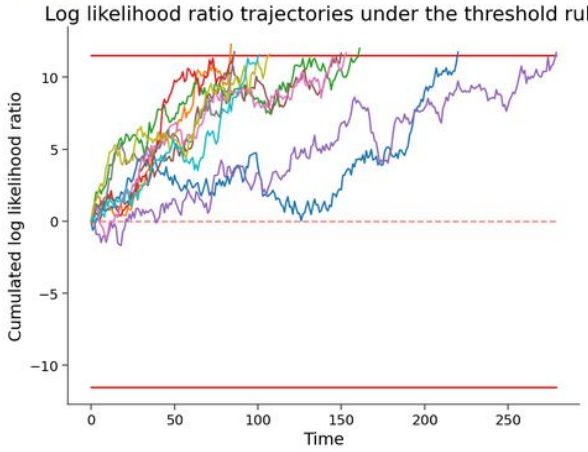
{Experimental evidence suggests evidence accumulation / thresholding strategy happens @ neuronal level!}

$$th_L = \log \frac{\alpha}{1-\alpha} = -th_R$$

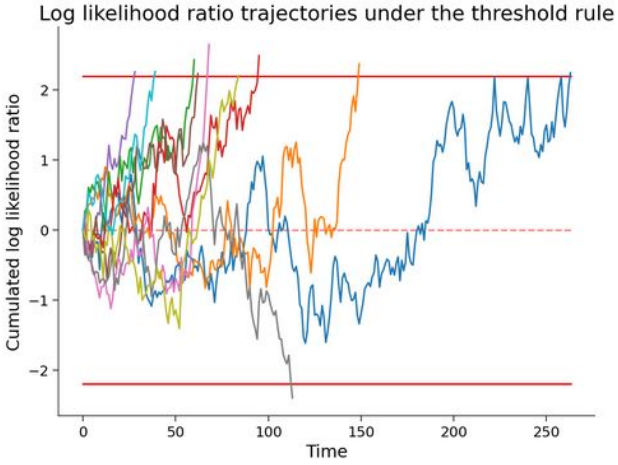
$$th_R = \log \frac{1-\alpha}{\alpha} = -th_L$$


LOG LIKELIHOOD RATIO TRAJECTORIES UNDER THE THRESHOLD RULE
CUMULATED LOG LIKELIHOOD RATIO
15
10
5
0
-5
-10
-15
0
20
40
60
80
100
TIME

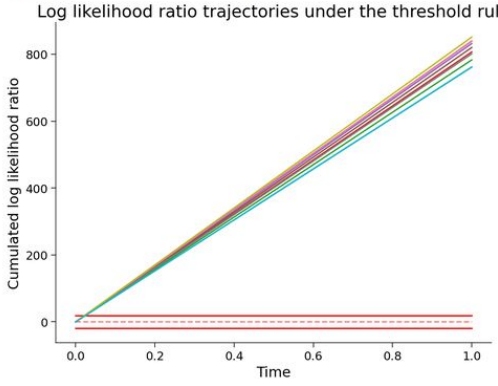
sigma 5.00
log10_alpha -5.00



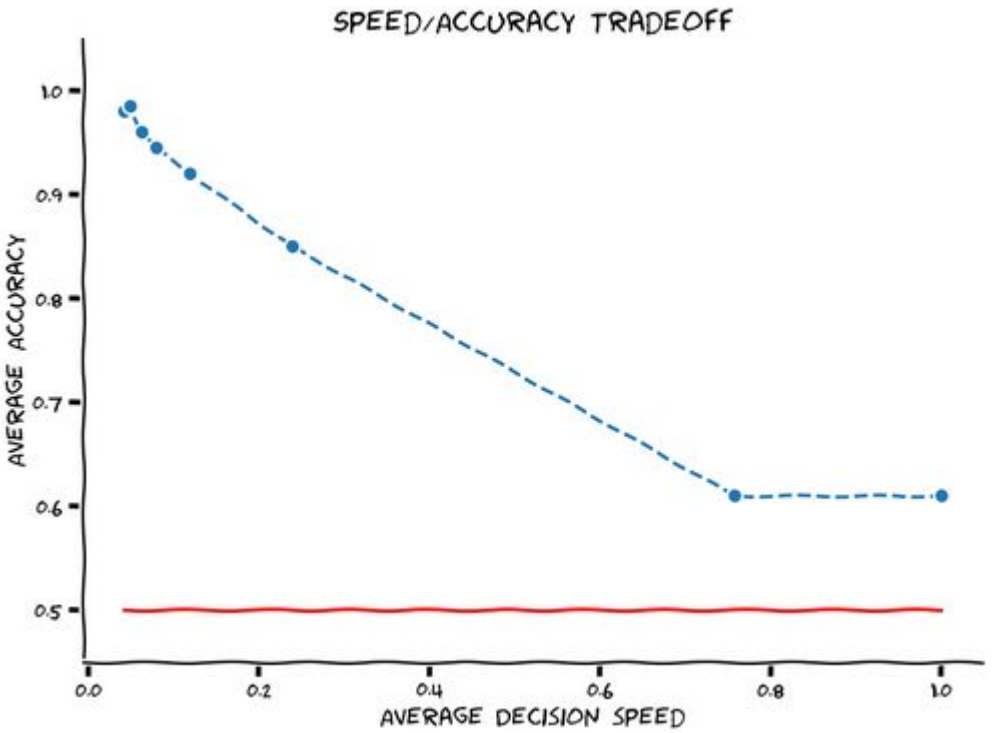
sigma 9.95
log10_alpha -1.00

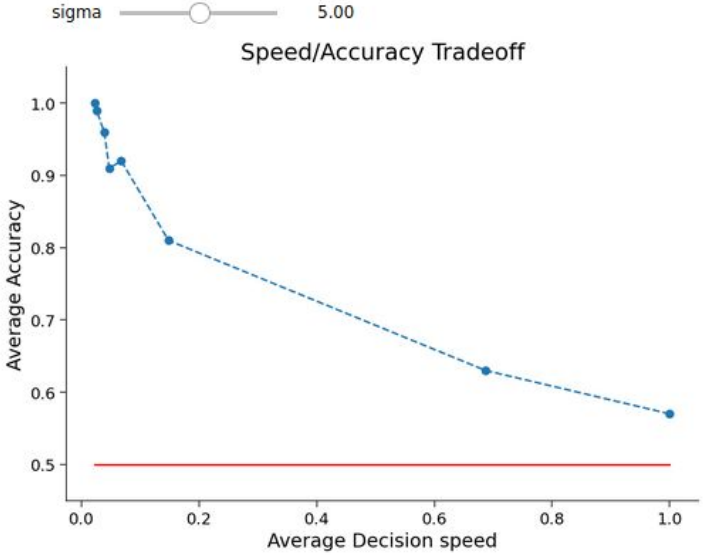


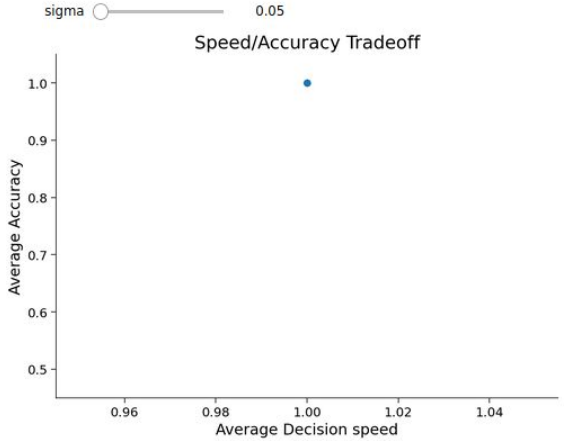
sigma 0.05
log10_alpha -8.00

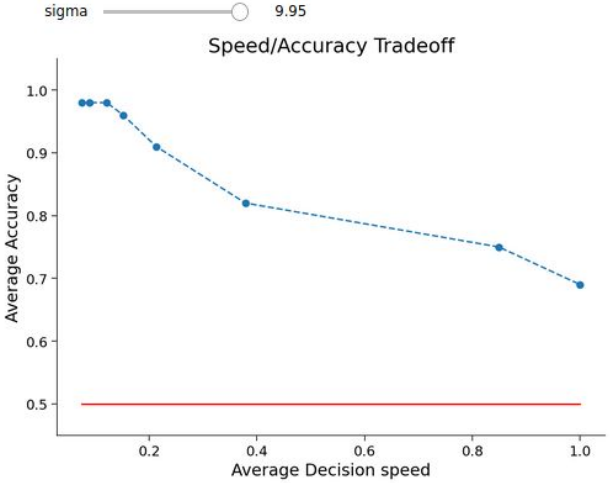


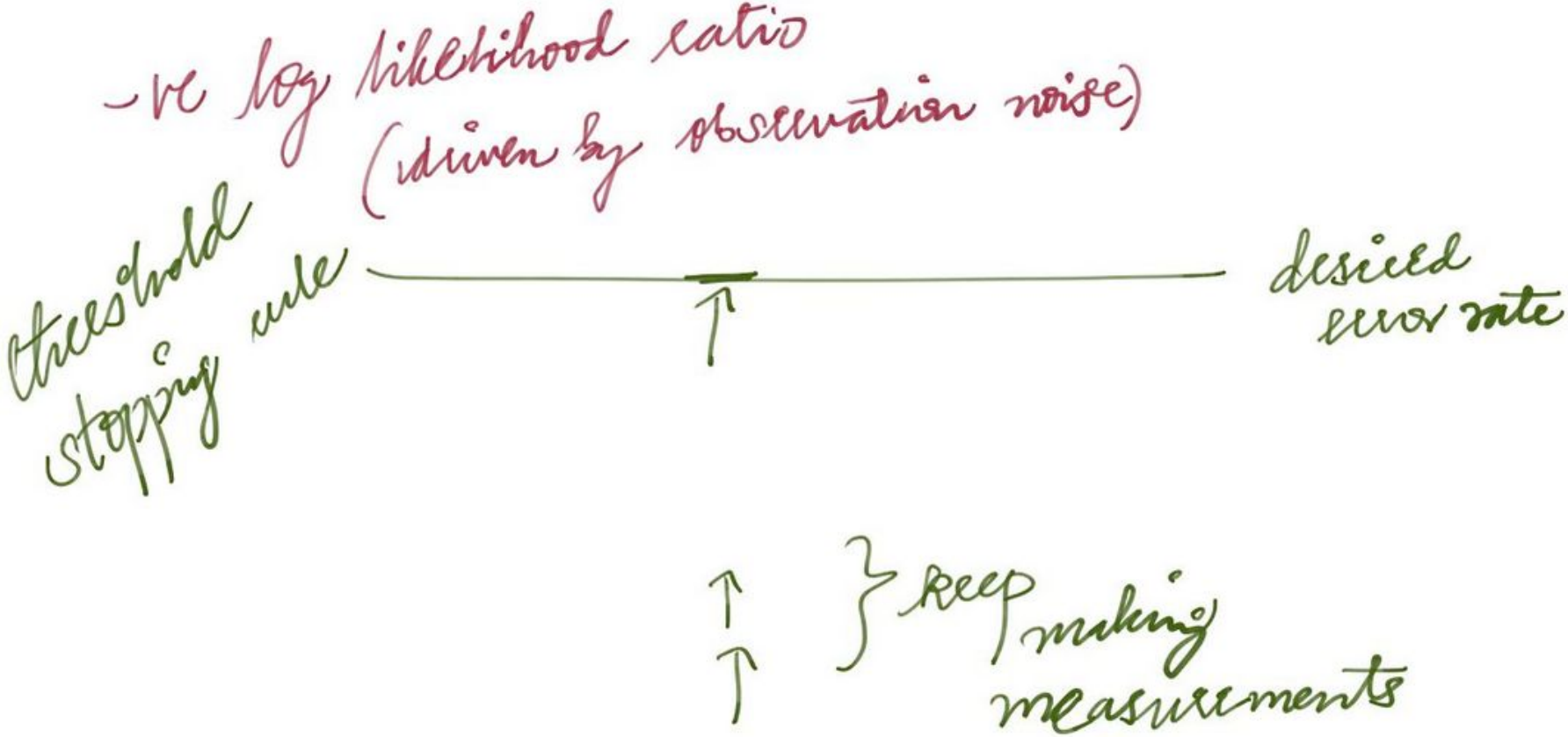
The faster you make a decision, the lower your accurac
often is. This phenomenon is known as the
speed/accuracy tradeoff.
Study how average decision "speed" (1/length)
changes with average decision accuracy.
We use speed rather than accuracy because
in real experiments, subjects can be incentivized
to respond faster or slower; it's much harder to
precisely control their decision time or error
threshold.




sigma
5.00
Speed/Accuracy Tradeoff
Average Accuracy
Average Decision speed


sigma
0.05
Speed/Accuracy Tradeoff
Average Accuracy
Average Decision speed


sigma
9.95
Speed/Accuracy Tradeoff
Average Accuracy
Average Decision speed


-ve log likelihood ratio
(driven by observation noise)
threshold
stopping rule
desired
error rate
keep making
measurements

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: $x_t \in \{0, 1\}$

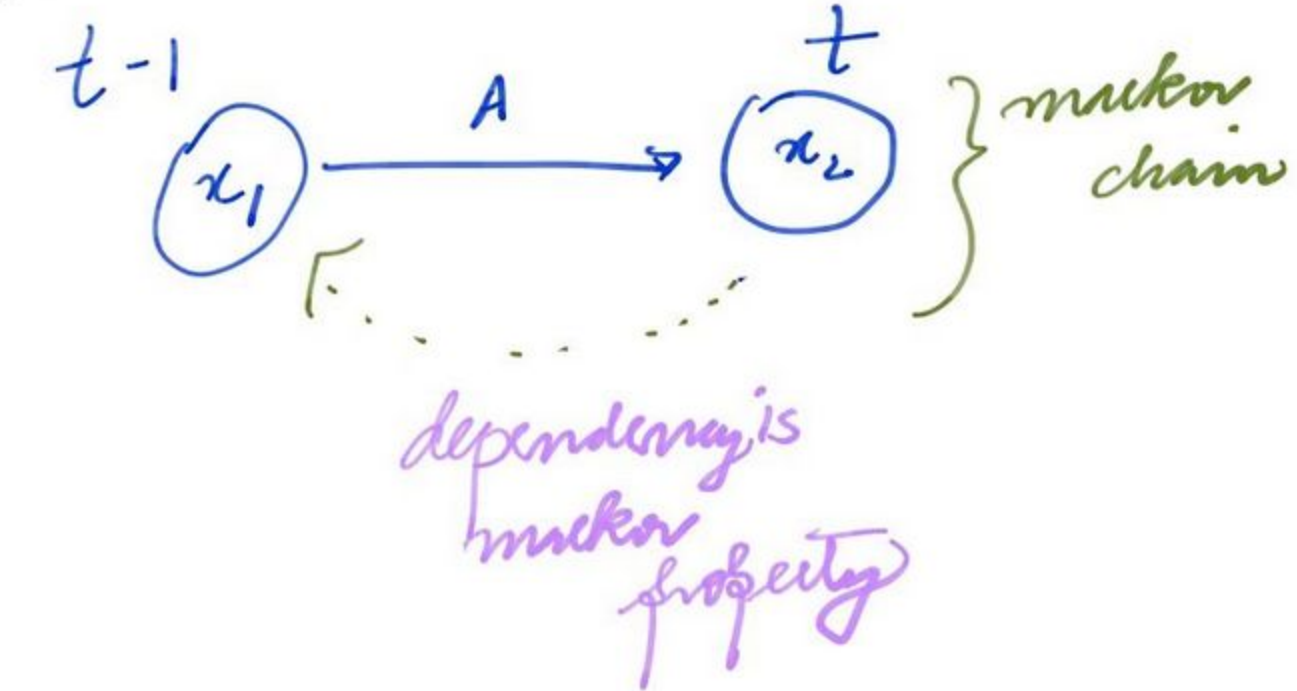
1D gaussian emission model: $y_t | x_t \sim \mathcal{N}(\mu_x, \sigma^2_{x_t})$

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



latent state – not fixed over time

probabilistically switch/jump to different state at each time step.

p(state @ time t) determined by state at t-1.

↳ Markov/markovian

Quantitatively,

$$p(x_t = j) = \sum_i A_{ij}\, p(x_{t-1} = i)$$

transition matrix

transition probability: probability of jumping states from i to j

latent states x_t : can't be observed

observe noisy measurement y_t generated from x_t.

Emission models :

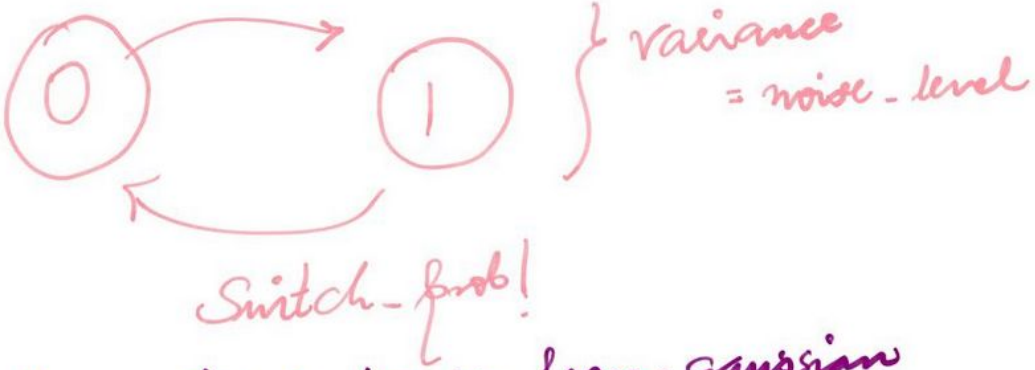
gaussian distribution $y_t | x_t \sim \mathcal{N}(\mu_{x_t}, \sigma^2_{x_t})$

(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

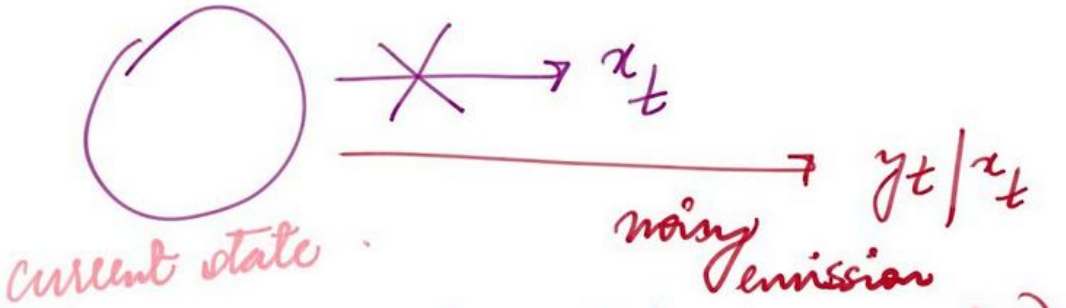
$\mu = -1$ for state = 1

State transitions:

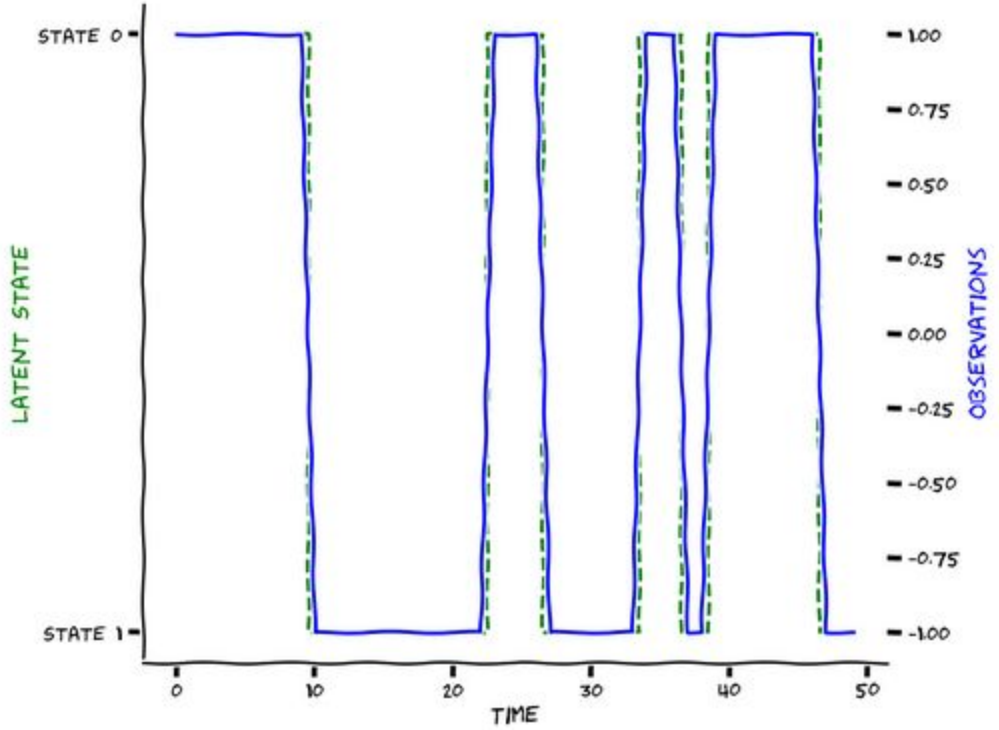
transition matrix $A_{ij} = \begin{pmatrix} p_{stay} & p_{switch} \\ p_{switch} & p_{stay} \end{pmatrix}$

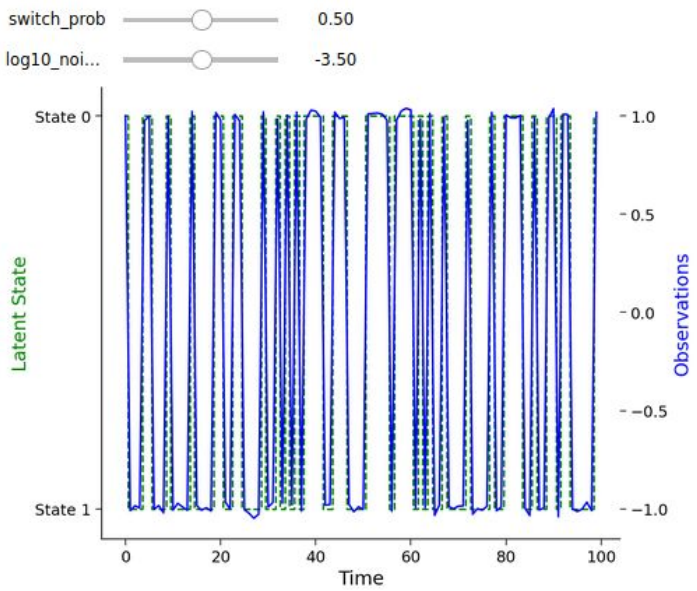
$p_{stay} = 1 - p_{switch}$

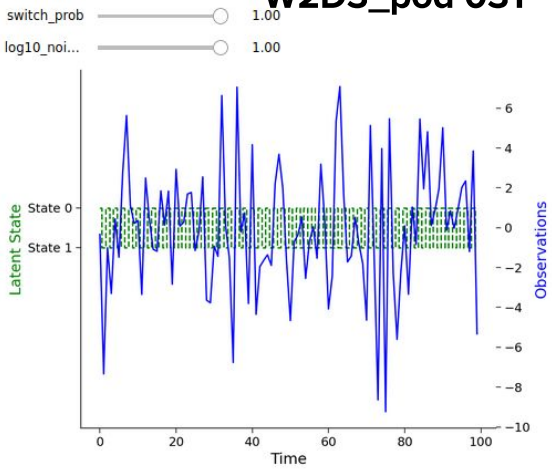
Hidden part of HMM -

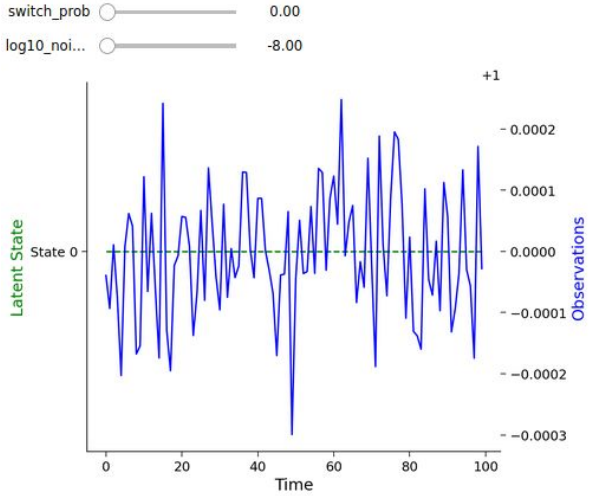


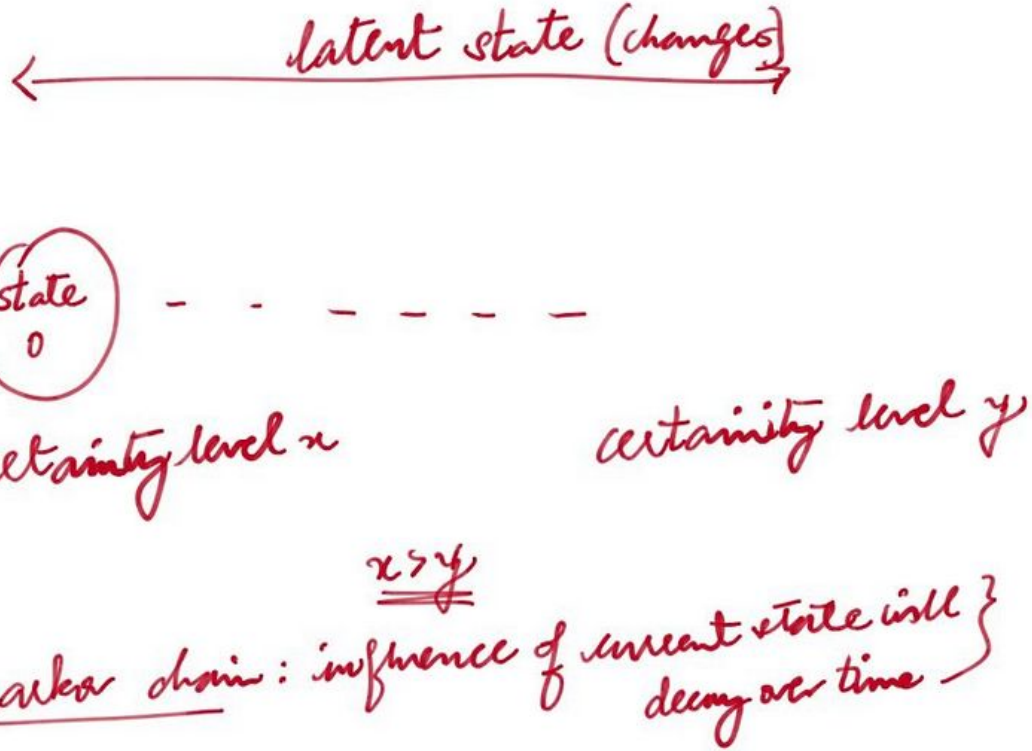
State_obs_variance = noise_level


STATE 0
STATE 1
LATENT STATE
1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
OBSERVATIONS
0
10
20
30
40
50
TIME


switch_prob
0.50
log10_noi...
-3.50
State 0
State 1
Latent State
Observations
1.0
0.5
0.0
−0.5
−1.0
0
20
40
60
80
100
Time


switch_prob
1.00
log10_noi...
1.00
State 0
State 1
Latent State
Observations
6
4
2
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
Time

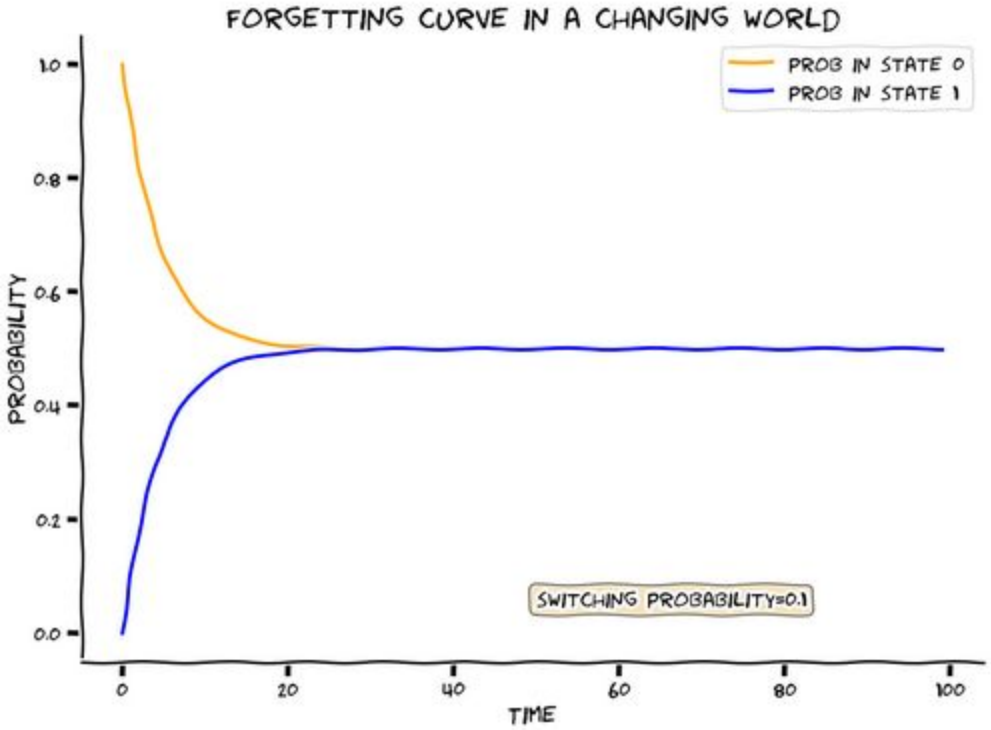

switch_prob
0.00
log10_noi...
-8.00
+1
State 0
Latent State
Observations
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0003
0
20
40
60
80
100
Time

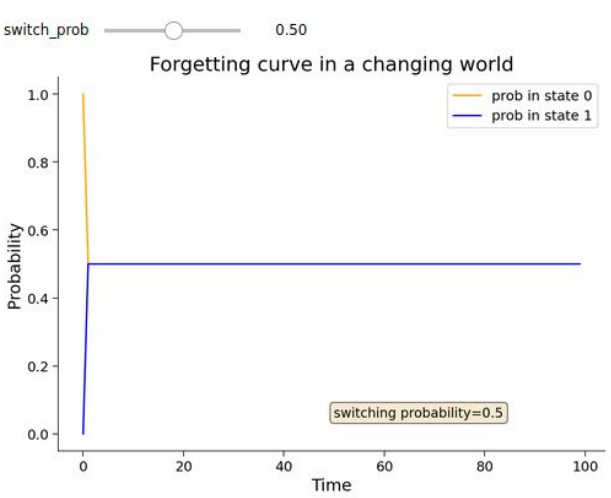

latent state (changes)
state 0
certainty level x
certainty level y
x > y
markov chain: influence of current state is ll decay over time

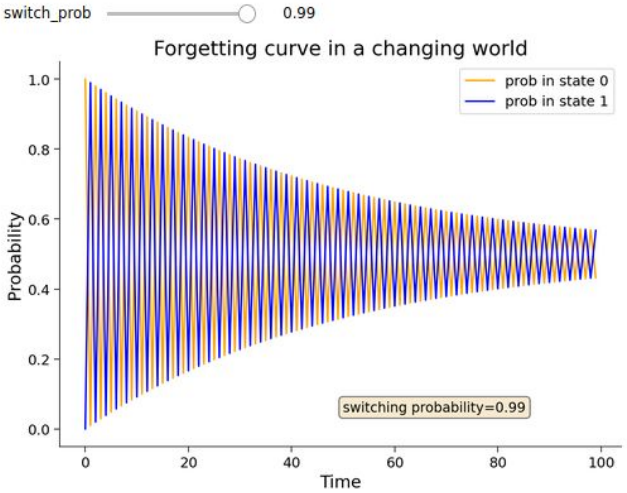
When we try to make predictions of future states in a markov chain based on current knowledge without future evidence.

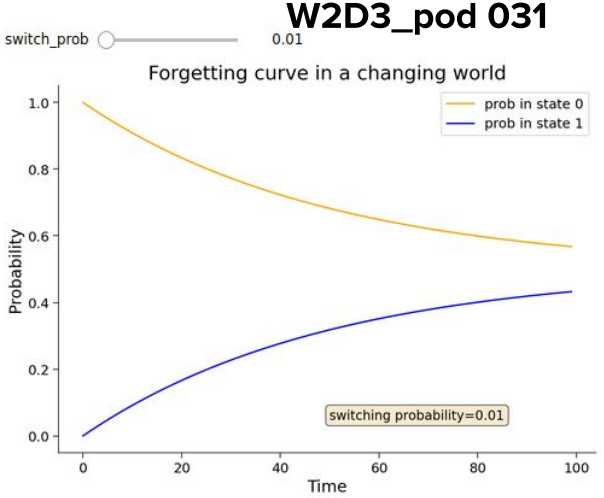
forget current state information when predicting future states without any observation

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

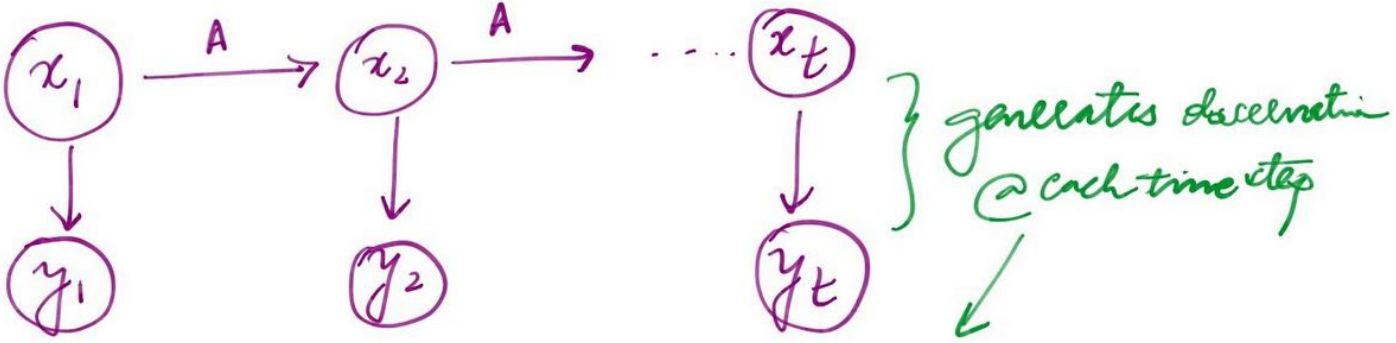

FORGETTING CURVE IN A CHANGING WORLD
PROB IN STATE 0
PROB IN STATE 1
PROBABILITY
1.0
0.8
0.6
0.4
0.2
0.0
SWITCHING PROBABILITY=0.1
0
20
40
60
80
100
TIME







How switching probabilities affects the information loss.



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

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:

$$p(x_t \mid y_{1:t}) \propto p(x_t \mid y_{1:t-1})\, p(y_t \mid x_t)$$

$p(x_t \mid y_{1:t-1})$: prediction

$p(y_t \mid x_t)$: new data likelihood for two gaussian observation models

$$\text{posterior}(x_t \mid y_{1:t}) \propto \text{prediction}(x_t \mid y_{1:t-1}) \odot \text{likelihood}(y_t \mid x_t)$$

Algorithm :

create_model (switch prob 0.1, noise level 0.5)

calculate marginal posterior distribution

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

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

→ calculate predictive distribution

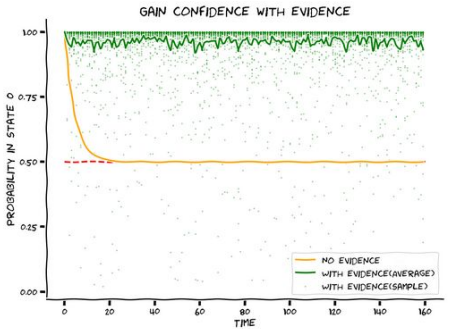
$$p(x_t = j | y_{1:t-1}) = \sum_i A_{ij}\, p(x_{t-1} = i | y_{1:t-1})$$

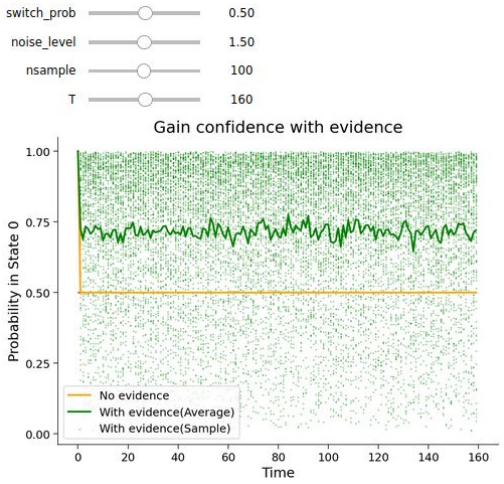
→ calculate likelihood of new data

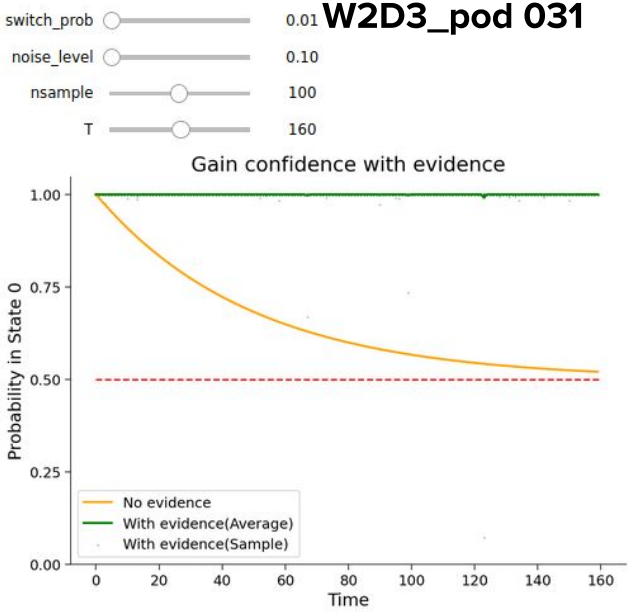
→ multiply likelihood & prediction element wise

normalise

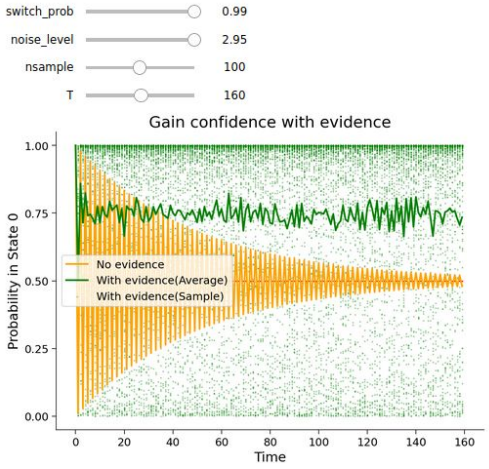
Plot of information loss due to diffusion together with the information recovered/uncertainty reduced due to evidence. The difference between the former and the latter is the amount of uncertainty that still remains because of observation noise.



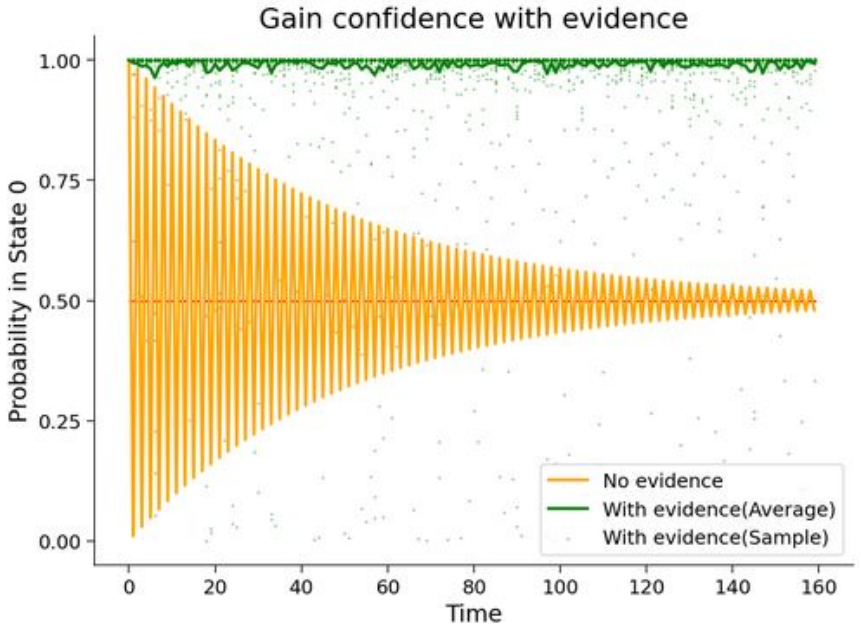

switch_prob 0.50
noise_level 1.50
nsample 100
T 160
Gain confidence with evidence
Probability in State 0
Time
No evidence
With evidence(Average)
With evidence(Sample)


switch_prob 0.01
noise_level 0.10
nsample 100
T 160
Gain confidence with evidence
Probability in State 0
Time
No evidence
With evidence(Average)
With evidence(Sample)

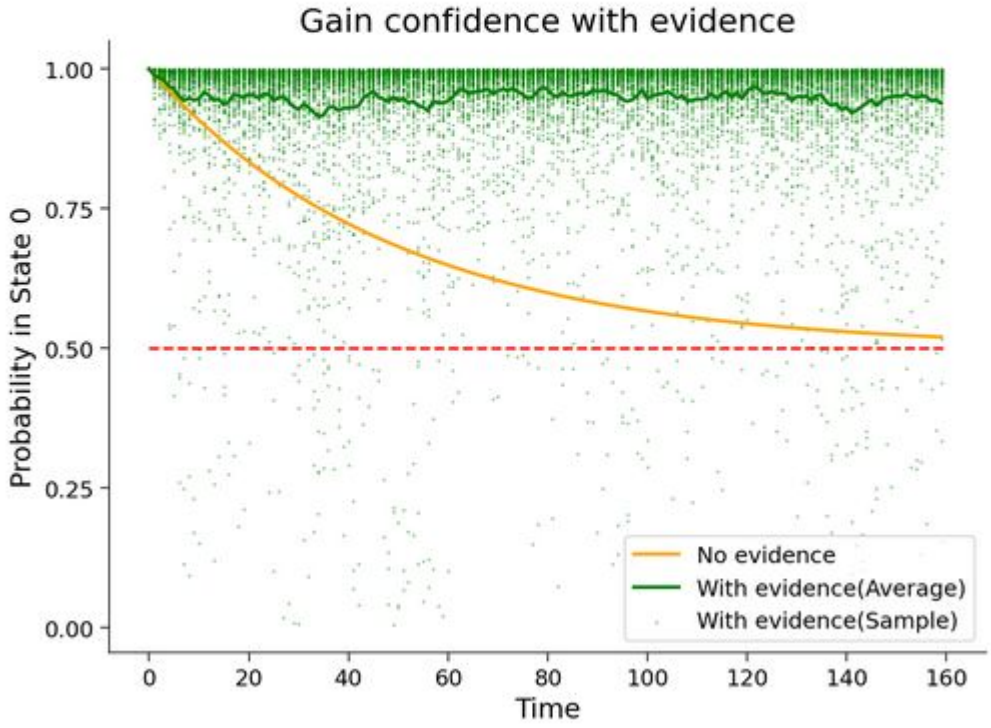
W2D3_pod 031


switch_prob 0.99
noise_level 2.95
nsample 100
T 160
Gain confidence with evidence
Probability in State 0
Time
No evidence
With evidence(Average)
With evidence(Sample)

switch_prob 0.99
noise_level 0.10
nsample 100
T 160



switch_prob 0.01
noise_level 2.95
nsample 100
T 160



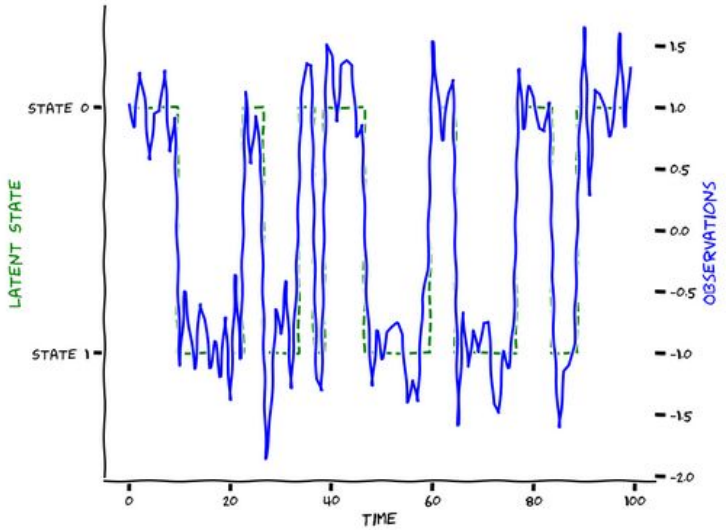
OBSERVATIONS

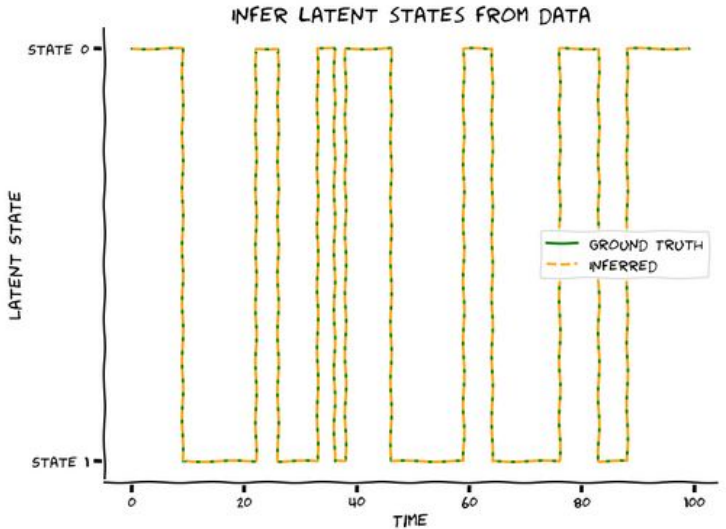
If switch_prob or noise_level is large, some sample inference dots fall below 0.5. This means we are making false inferences about which latent state we are in.

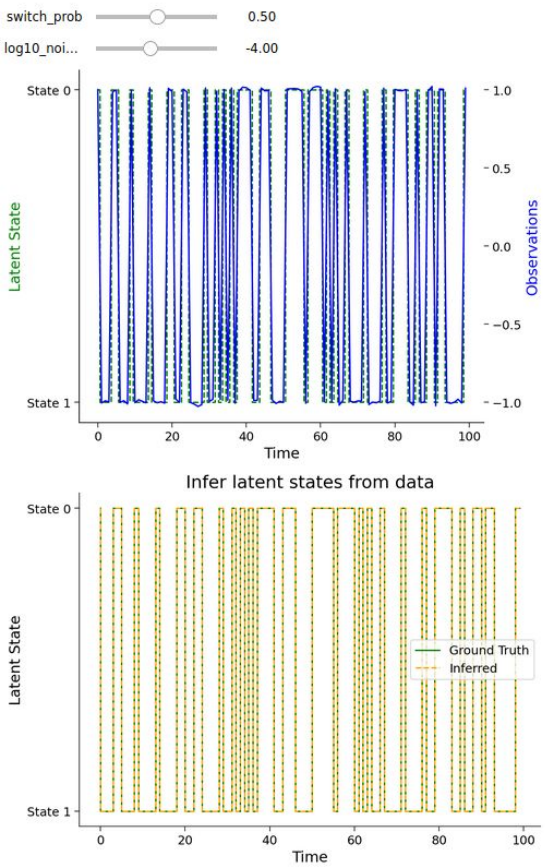
In this exercise, let's make a forward inference of a random state sequence rather than a constant one by observing its noisy Gaussian outputs.

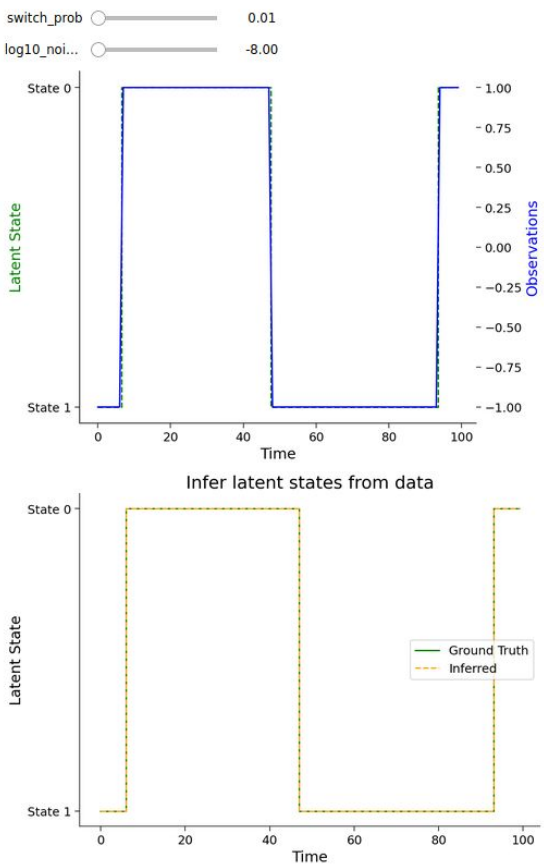
forward inference of HMM

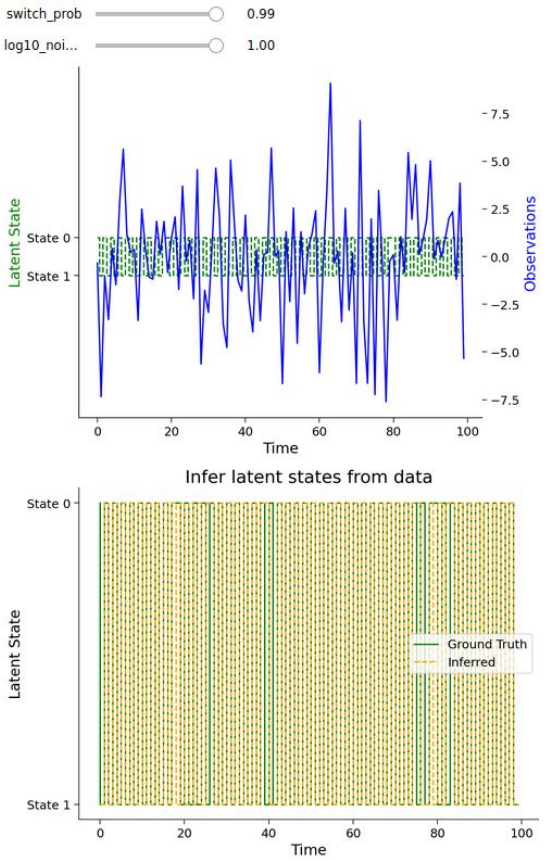
1. hmm with fair probabilities (0.5, 0.5)
   Switching probabilities 0.1
   noisy level 0.1
2. generate sample sequence along with observations
3. Calc. posterior probabilities.
4. visualise inferred state sequence with ground truth.


STATE 0
STATE 1
LATENT STATE
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
OBSERVATIONS
0
20
40
60
80
100
TIME


INFER LATENT STATES FROM DATA
STATE 0
STATE 1
LATENT STATE
GROUND TRUTH
INFERRED
0
20
40
60
80
100
TIME


switch_prob
0.50
log10_noi...
-4.00
State 0
State 1
Latent State
Observations
Time
Infer latent states from data
Ground Truth
Inferred


switch_prob
0.01
log10_noi...
-8.00
State 0
State 1
Latent State
Observations
Time
Infer latent states from data
Ground Truth
Inferred


switch_prob
0.99
log10_noi...
1.00
State 0
State 1
Latent State
Observations
Time
Infer latent states from data
Ground Truth
Inferred

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

(HMM for poisson spiking neurons case!)

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 - λ_i^c

state transitions : K×K transition matrix A_{ij}

initial probability vector ψ
(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 true model

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

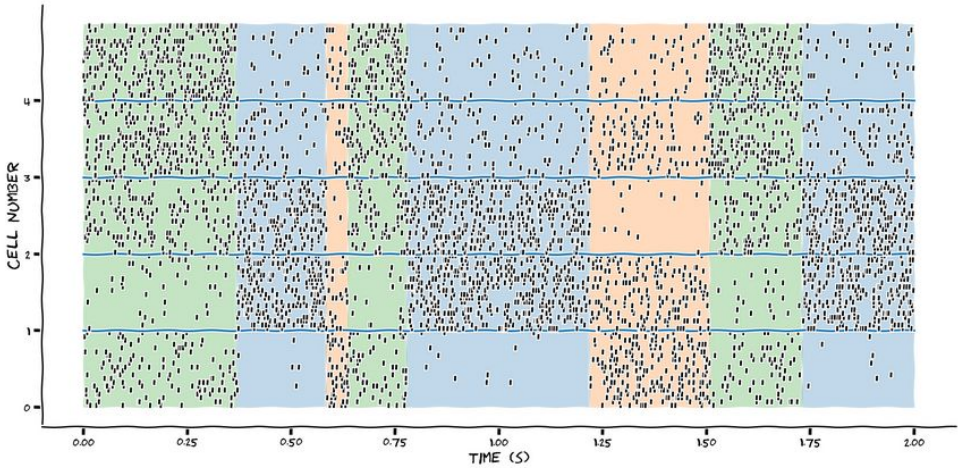
given state sequence

convert

one hot coding of

(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.

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

find optimal values of parameters that maximises 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 $a_i(t)$ & $b_i(t)$.

(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 $y_{1:t}$

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

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

$$b_i(t) = \sum_j f_\theta\left(y_{t+1} \middle| x_{t+1} = j\right) b_j(t+1) A_{ij}$$

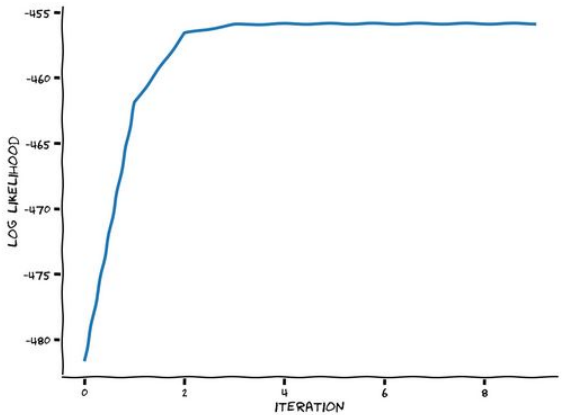
Singleton and pairwise marginal distributions.

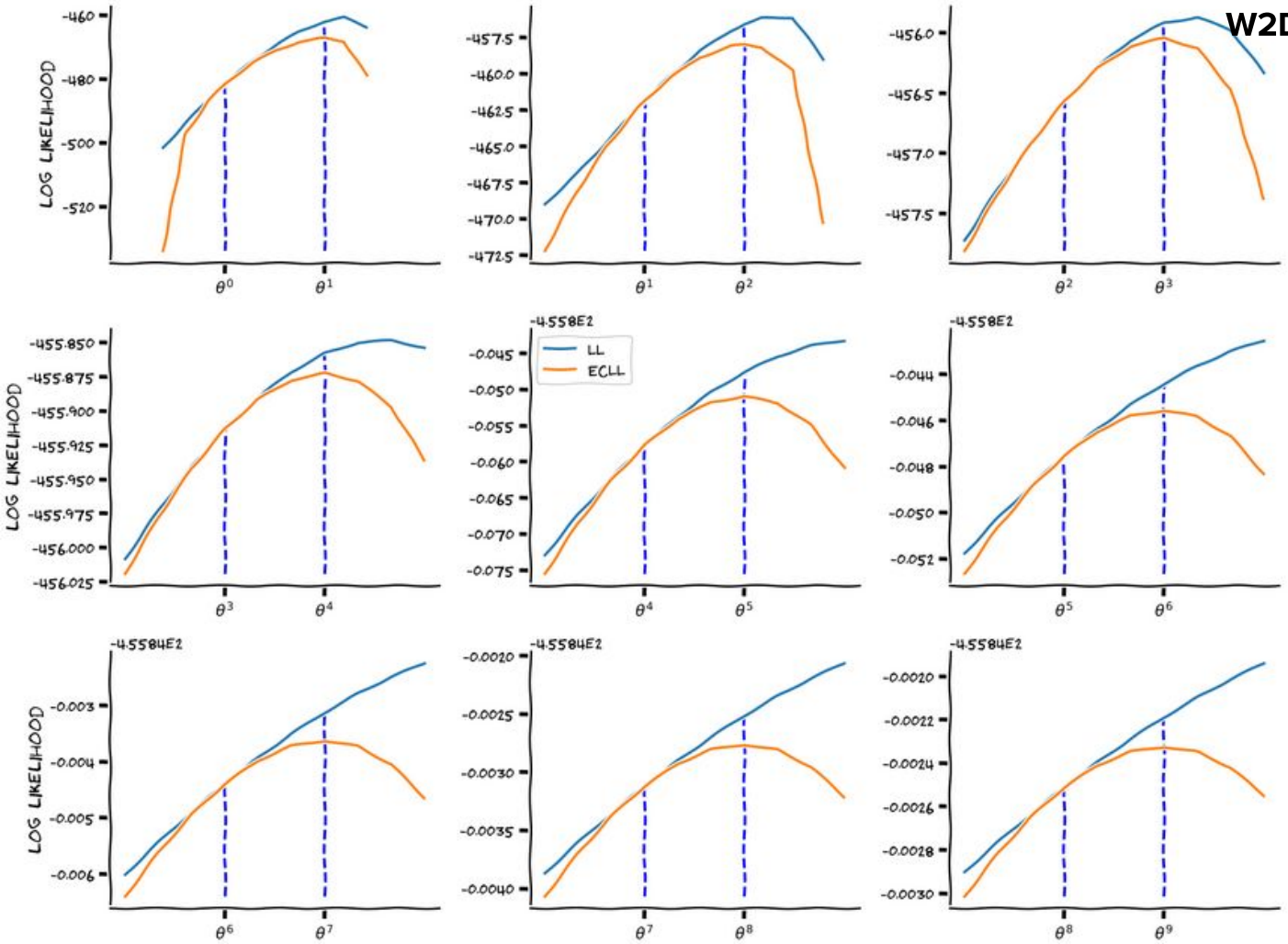
$$\gamma_i(t) = p_\theta(x_t = i \mid Y_{1:T}) = \frac{a_i(t)\, b_i(t)}{p_\theta(Y_{1:T})}$$

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

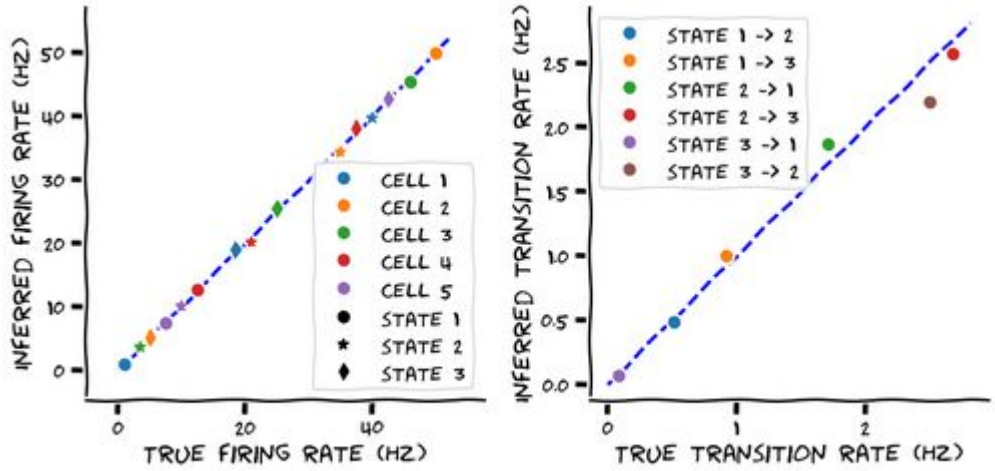
where $p_\theta(Y_{1:T}) = \sum_i a_i(T)$

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




LOG LIKELIHOOD
LL
ECLL
-4.558E2
θ^0 θ^1 θ^2 θ^3 θ^4 θ^5 θ^6 θ^7 θ^8 θ^9

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 $s_t = Fs_{t-1} + \zeta_t$ (D dimensions)

measured/observed variable $y_t = Hs_t + \eta_t$ (N dimensions)

by principle of dimension reduction - $D < N$

Gaussian noise terms -

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

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

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

$\Rightarrow$ s_t & y_t are gaussian

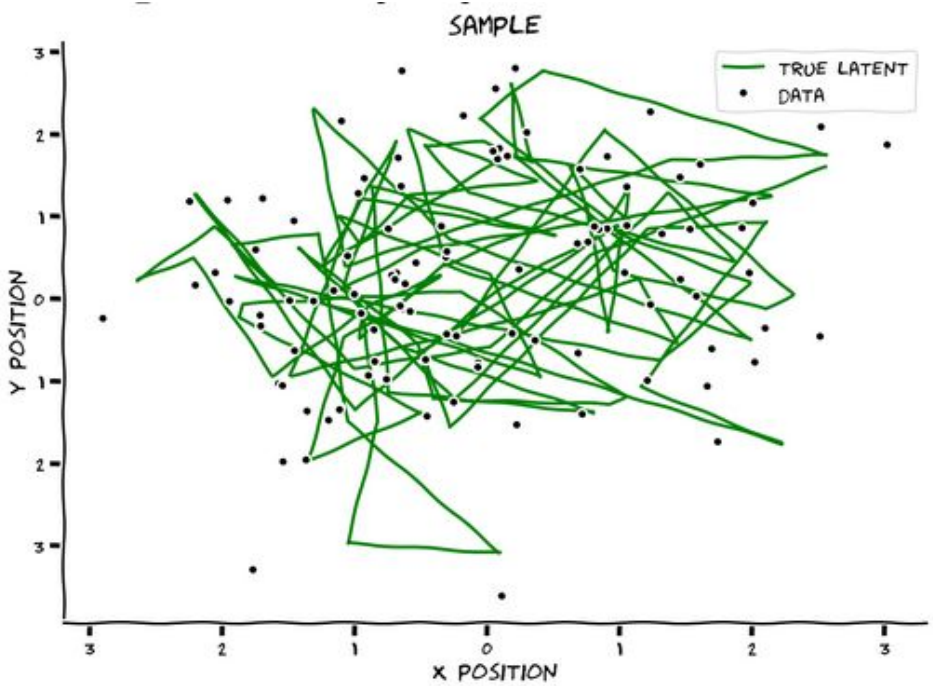
markov chain: state @ t is conditionally independent given t-1

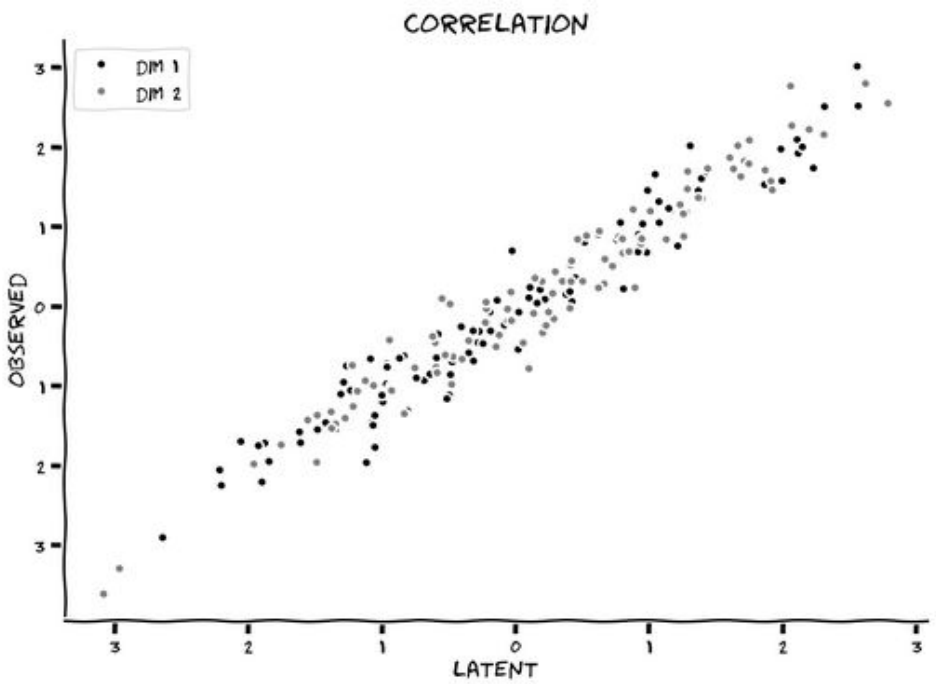
Sampling

```
# 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
}
```

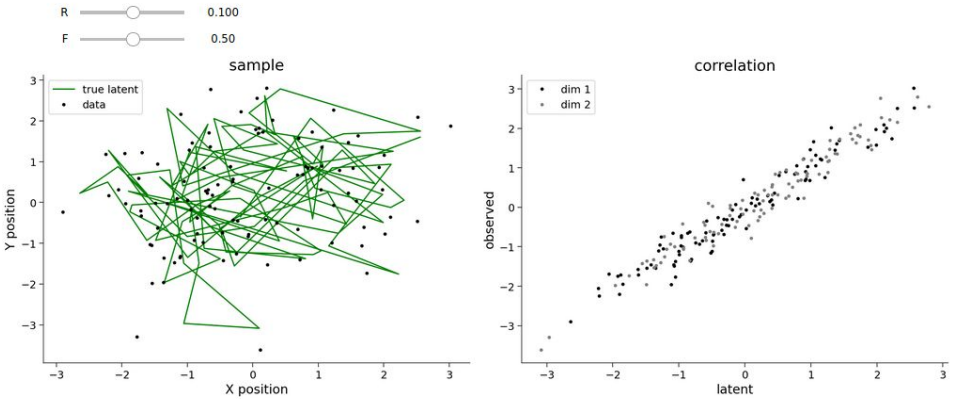
Note: *Parameter dictionary (params) is used but as the number of parameters increases, it can be beneficial to condense them into a data structure. The trade-off is that we have to know what is in the data structure to use the values, rather than function signature.*

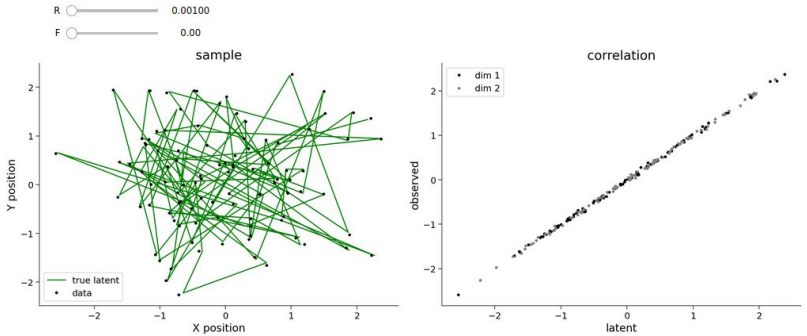

SAMPLE
TRUE LATENT
DATA
Y POSITION
X POSITION
3
2
1
0
1
2
3

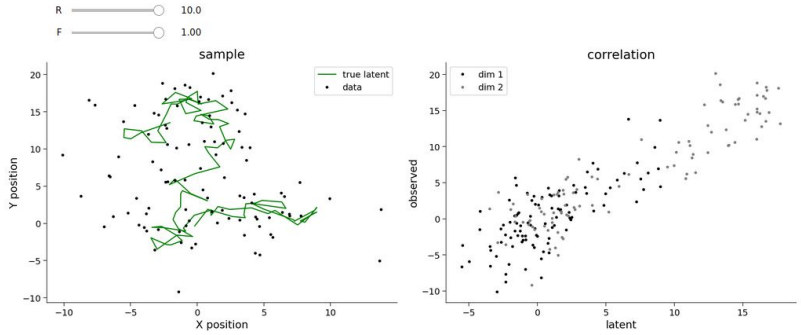

CORRELATION
DIM 1
DIM 2
OBSERVED
LATENT
3
2
1
0
1
2
3

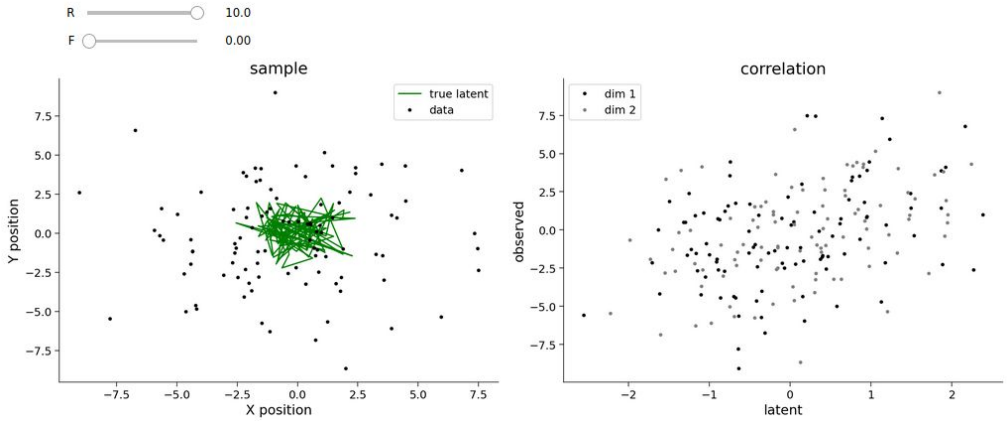
Adjusting System Dynamics

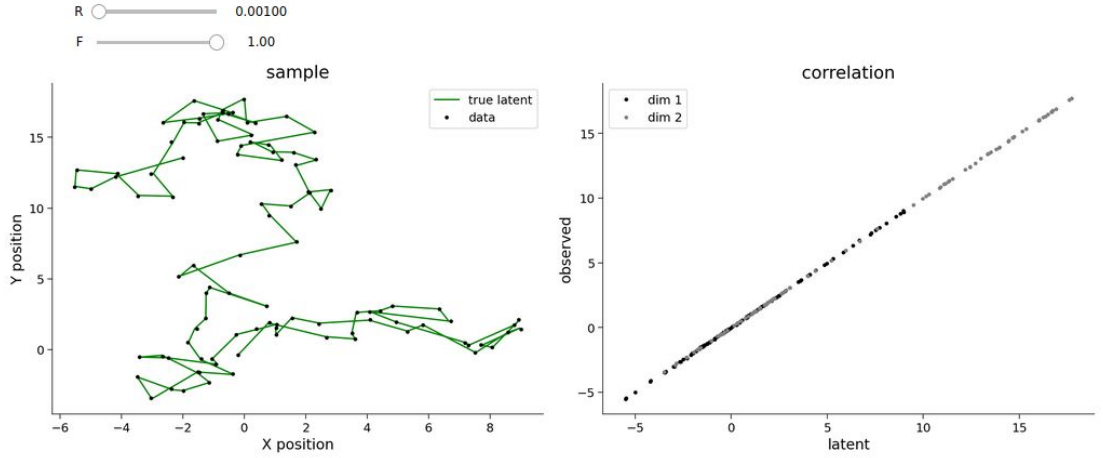
parameters of a linear dynamical system: Reduce observation noise R and respective temporal dynamics F





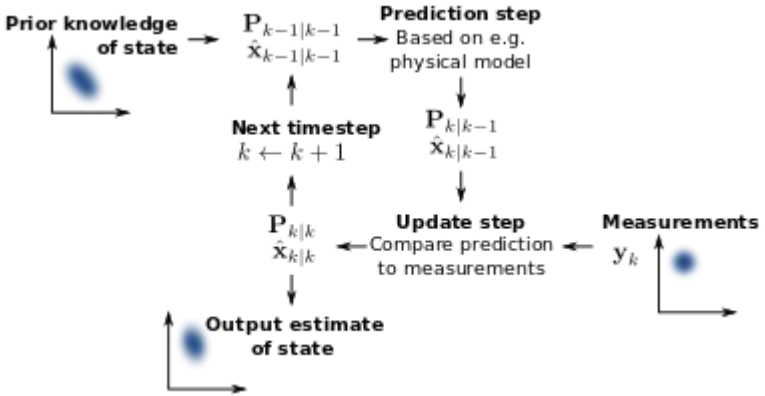



R
10.0
F
0.00
sample
true latent
data
Y position
X position
correlation
dim 1
dim 2
observed
latent


R
0.00100
F
1.00
sample
true latent
data
Y position
X position
correlation
dim 1
dim 2
observed
latent

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.

$$s_t^{pred} \sim N\left(\hat{\mu}_t^{pred}, \hat{\Sigma}_t^{pred}\right)$$

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

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

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

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

Q: noise covariance

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

project to observational space

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

update prediction by actual data

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

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

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

Kalman gain matrix

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

use only latent only prediction to project to observation space

Compute correction $\propto y_t - H F_{\theta_{t-1}}$ } prediction & data

Coefficient of correction = Kalman gain matrix

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

Gaussian Noise terms.

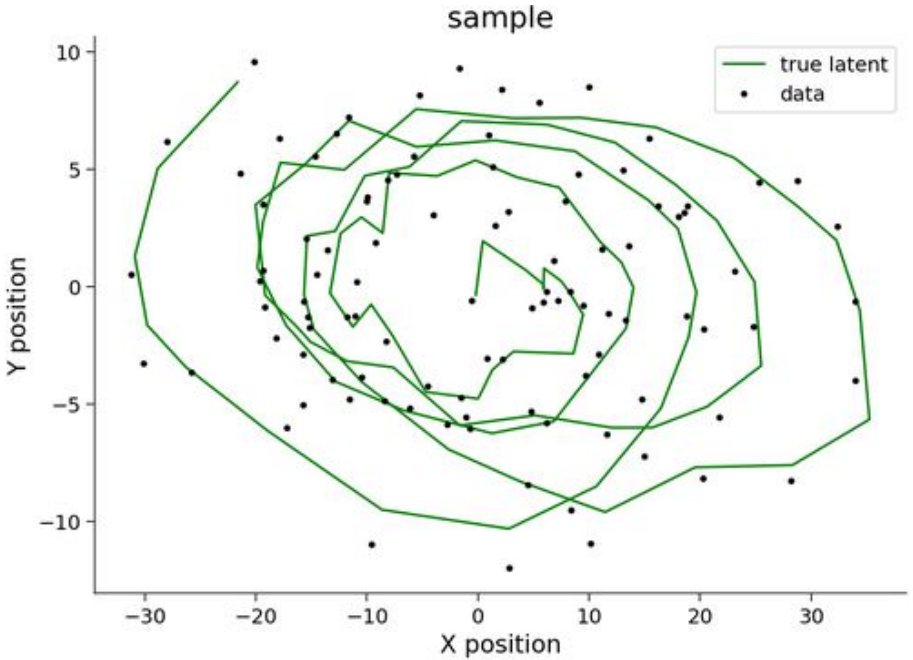
$$q_t \sim \mathcal{N}(0, Q)$$

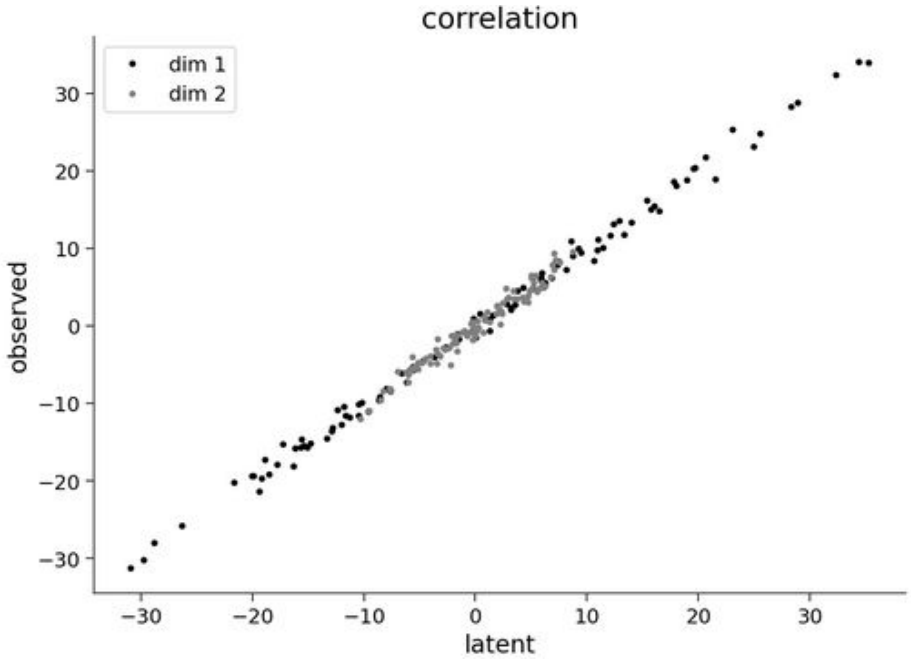
→ state noise covariance

$$r_t \sim \mathcal{N}(0, R)$$

→ observation noise covariance

$$s_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$$


sample
true latent
data
Y position
X position
10
5
0
−5
−10
−30
−20
−10
0
10
20
30


correlation
dim 1
dim 2
observed
latent
30
20
10
0
−10
−20
−30

Implement kalman filter (forward process)

Kalman gain K

filter mean μ

filter covariance Σ

@ each time step

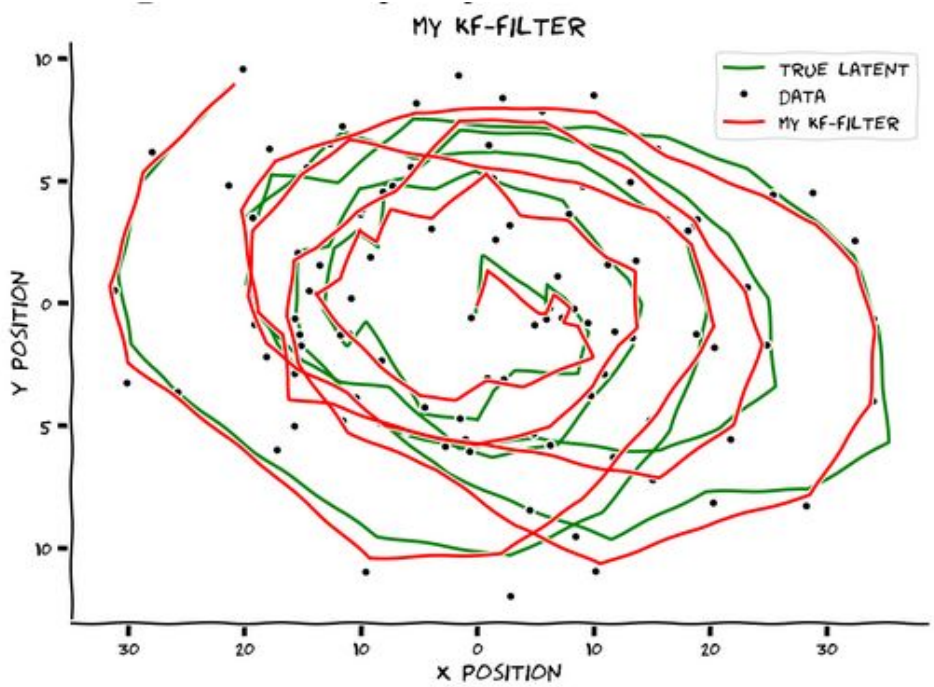
State tracking always

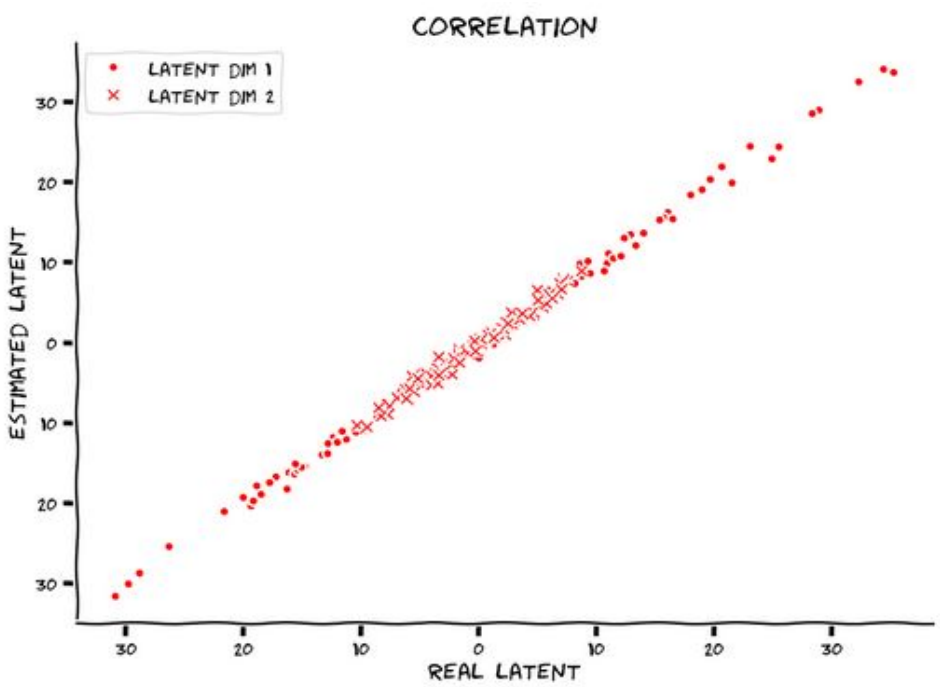
↓

filtering

↓

K, μ, Σ


MY KF-FILTER
TRUE LATENT
DATA
MY KF-FILTER
Y POSITION
X POSITION
10
5
0
5
10
30
20
10
0
10
20
30


CORRELATION
LATENT DIM 1
LATENT DIM 2
ESTIMATED LATENT
REAL LATENT
30
20
10
0
10
20
30

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).

W2D3_pod 031

subject_id -1

image_id 0

subject_id 4

image_id 2

subject_id 2

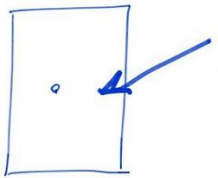
image_id 1

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)

② let EM discover the dynamics of: (x, y) coordinates
transition/observation matrices &
transition/observation covariance.

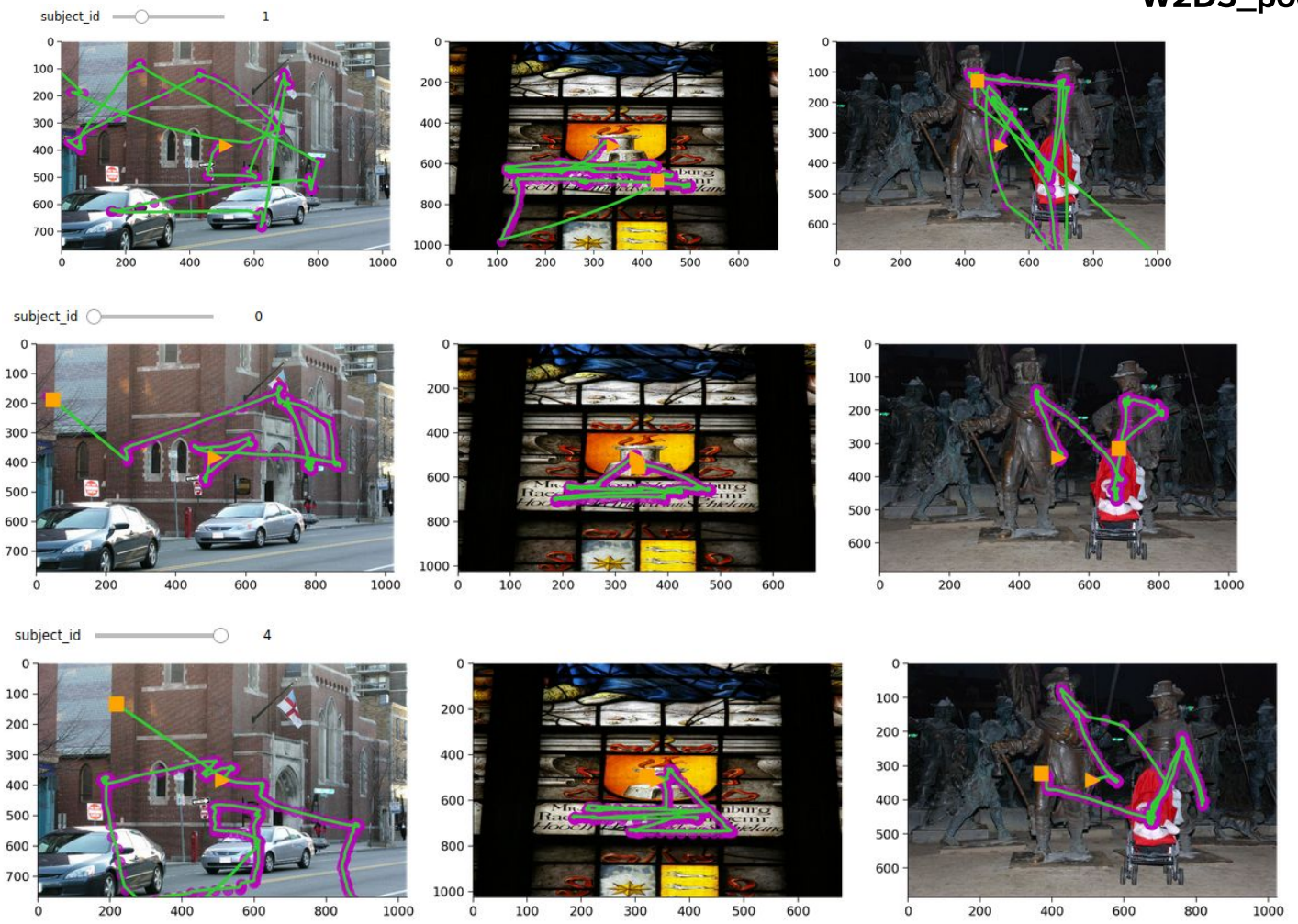


Subjects are fixated centre of screen
⇒ start state estimate μ_0
+ low initial noise covariance Σ_0

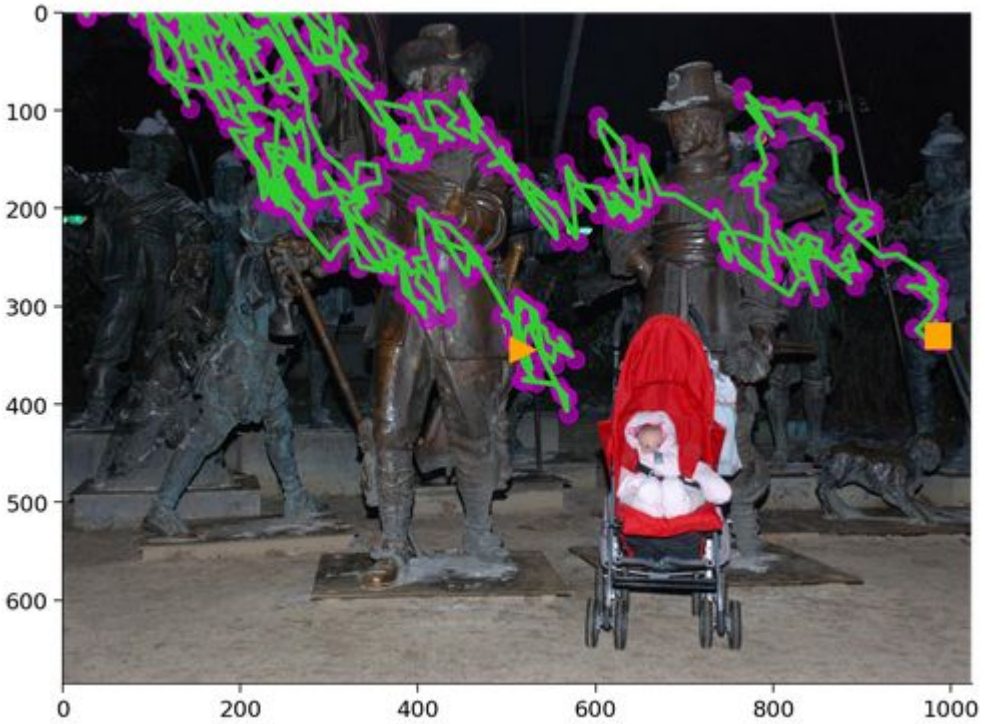
F =
[[1.004 -0.01]
 [0.005 0.989]]
Q =
[[278.016 219.292]
 [219.292 389.774]]
H =
[[0.999 0.003]
 [-0.004 1.01]]
R =
[[26.026 19.596]
 [19.596 26.745]

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: $z = [x^T y^T]^T$

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

marginal distributions: $x \sim N(a, A)$

$y \sim N(b, B)$

conditional distributions:

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

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

Kalman Smoothing

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

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

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

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

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

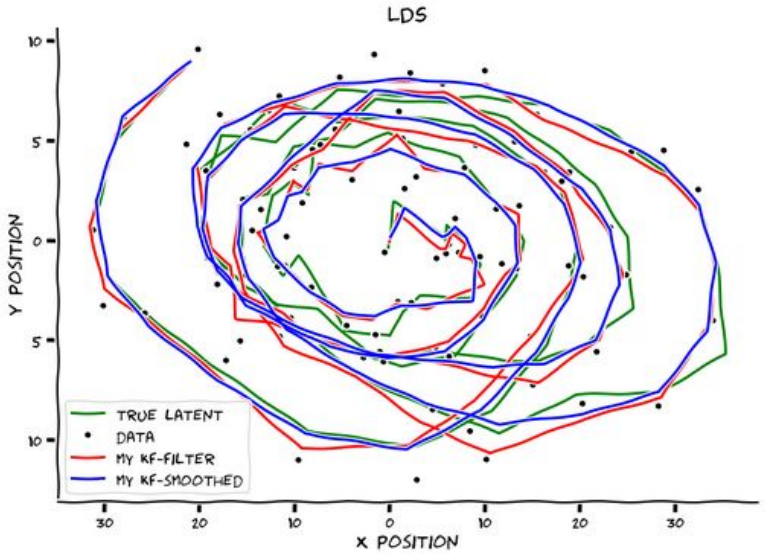
final estimate for z_t

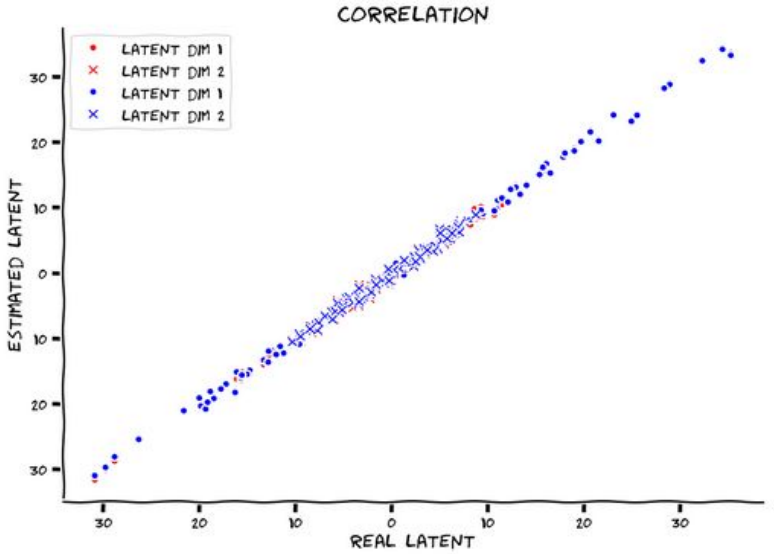
$$\hat{\mu}_t = \hat{\mu}_t^{\text{smooth}}$$

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

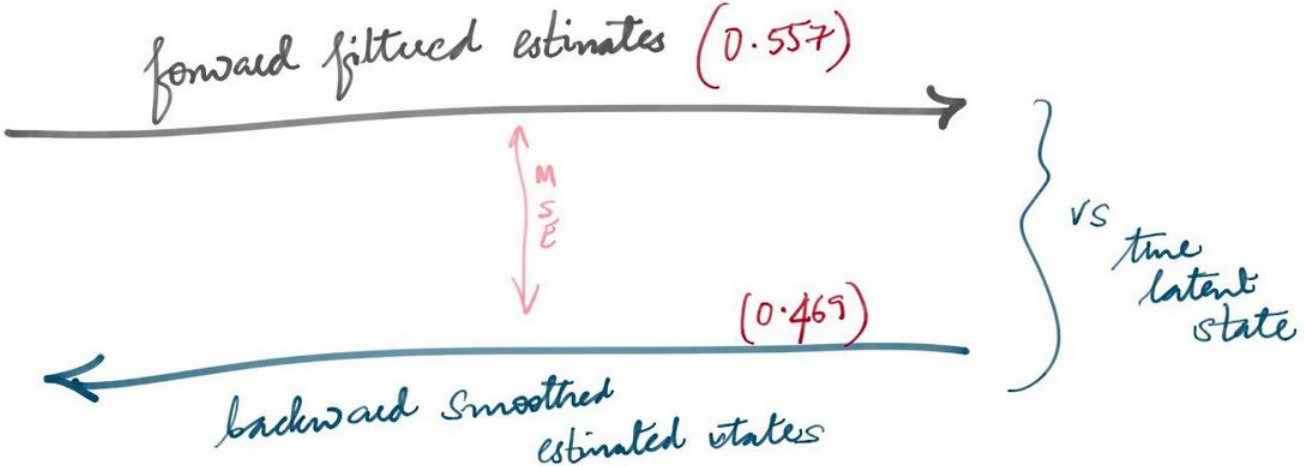
implement kalman smoothing (backward) process

- compute smoothed mean / covariance & J_t 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)

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

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

$$\log p(y|\theta) \int_s q(s)\,ds$$

joint distribution of y and s.

conditional distribution of $s|y$

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

Expectation step

(Kalman filter + smoother)

- fixed parameters
- good approximation

$q(s)$: max lower bound $L(q,\theta)$ 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 $\begin{cases} \hat{\mu}_t^{\text{smooth}} \\ \hat{\Sigma}_t^{\text{smooth}} \end{cases}$

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

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

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

parameter update

initial : $\mu_0^{new} = E(s_0)$

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

hidden / latent state

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

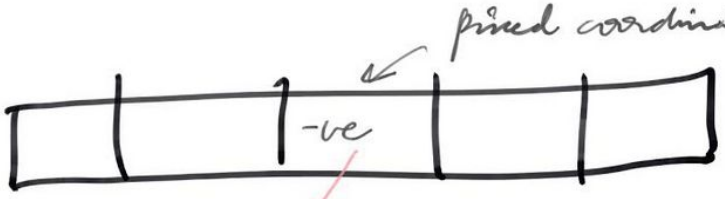
$$Q^{new} = \frac{1}{T-1} \sum_{t=2}^{N} E\left(s_t s_t^T\right) - F^{new} E\left(s_{t-1} s_t^T\right)$$

$$- E\left(s_t s_{t-1}^T\right) F^{new} + \left(F^{new} E\left(s_{t-1} s_{t-1}^T\right) \cdot \left(F^{new}\right)^T \right)$$

Observed/measured space

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

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



#data stream

issue mitigation - ① delete values.

con: integrity of consistent time step between samples: lost

② masked array

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