Linear Dynamical Systems

SWC Neuroinformatics 2024

Dr. Aniruddh Galgali

Gatsby Unit, UCL

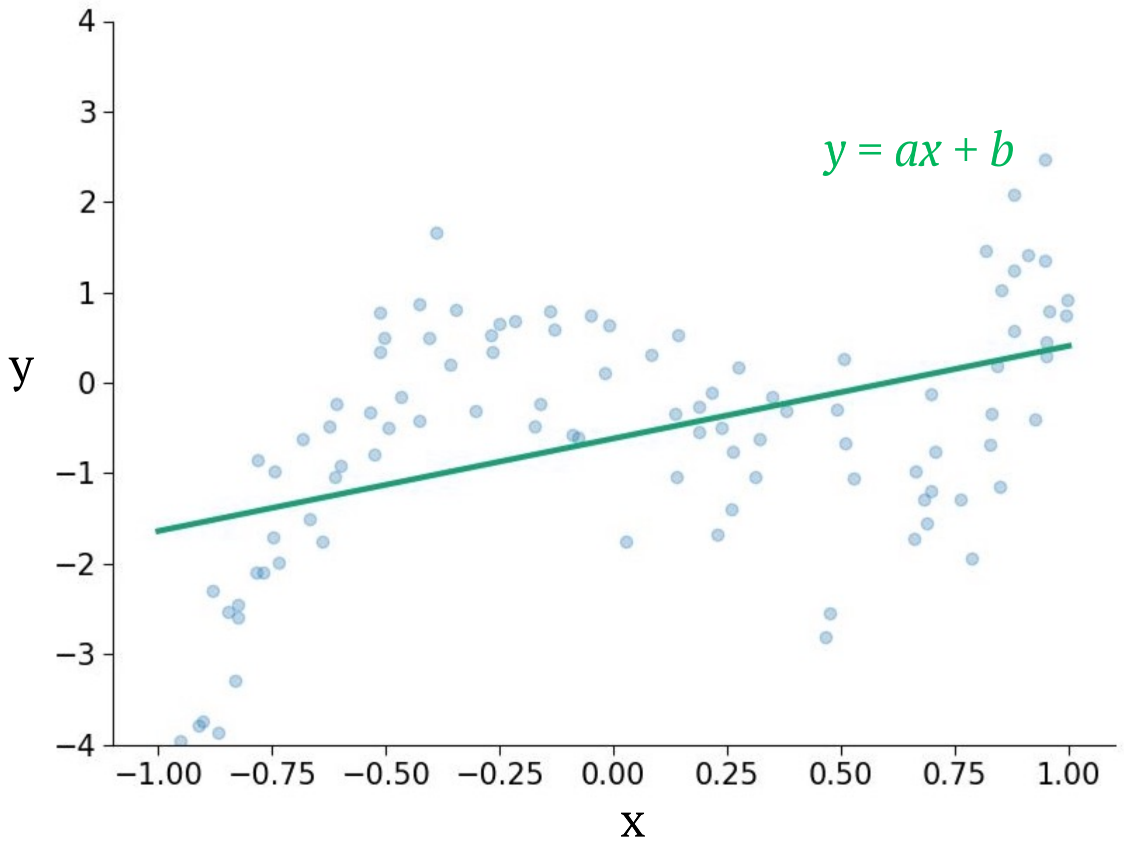
22 Feb 2024

Last week

Key assumptions

1. datapoints are independent & identically distributed (i.i.d)

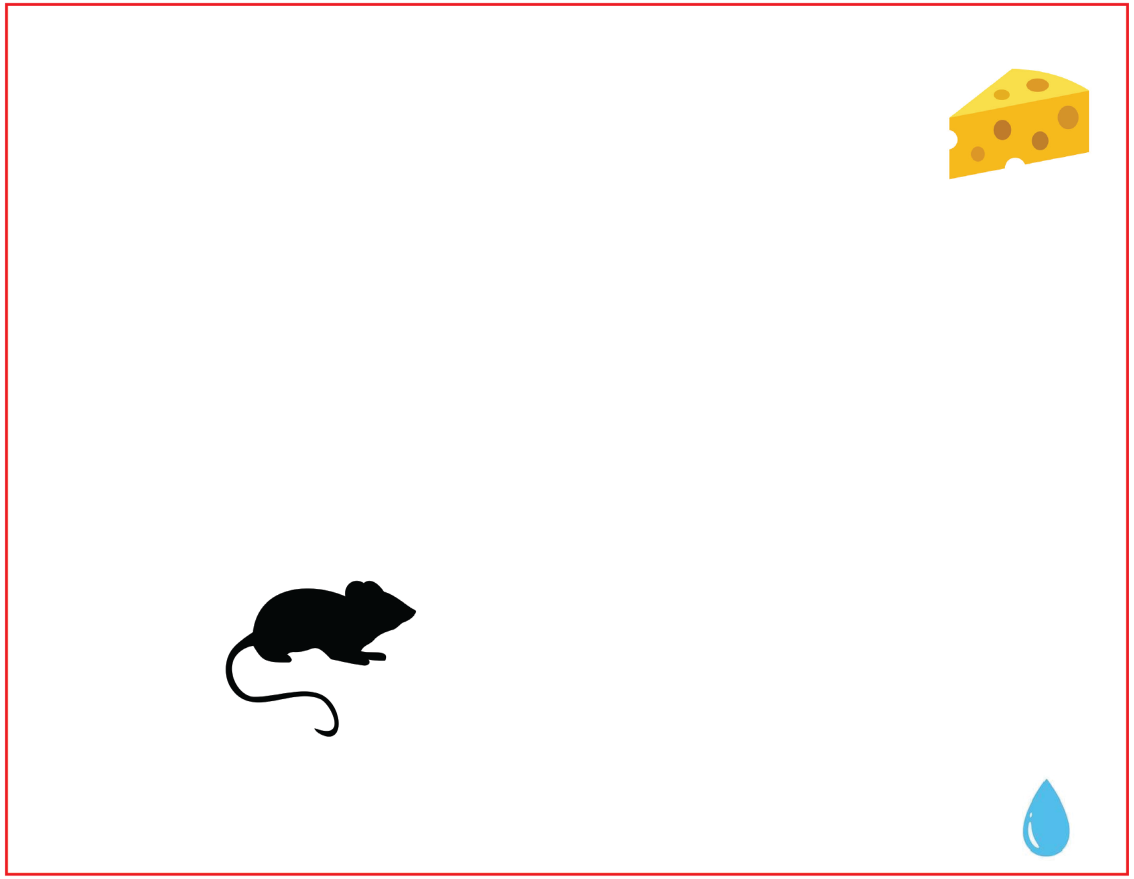
2. given x (regressor), model dependence of y on x ("discriminative" model)



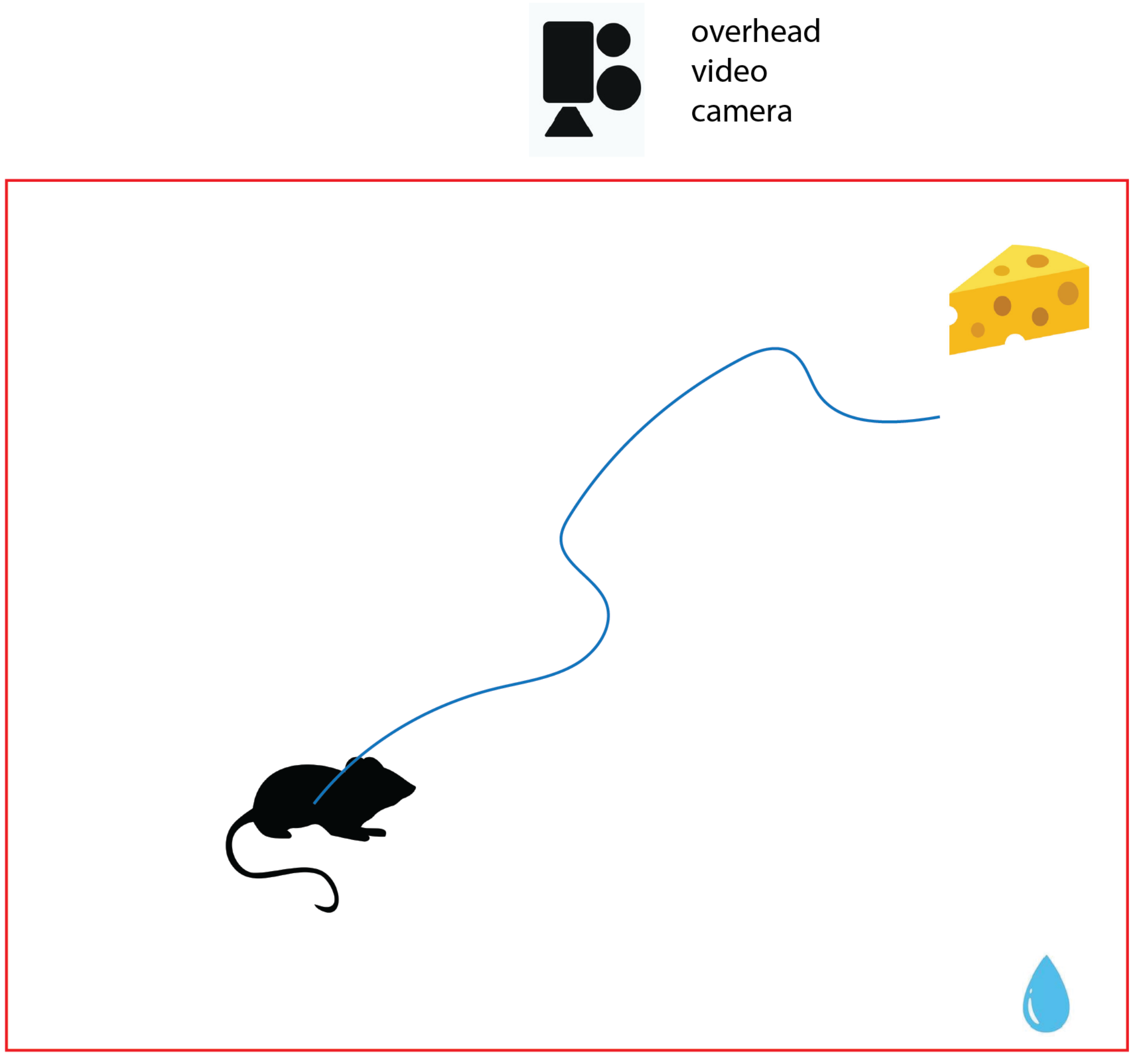
Most datasets in neuroscience typically violate assumption (1)

Today - a model for 'more' structured datasets (time-series)

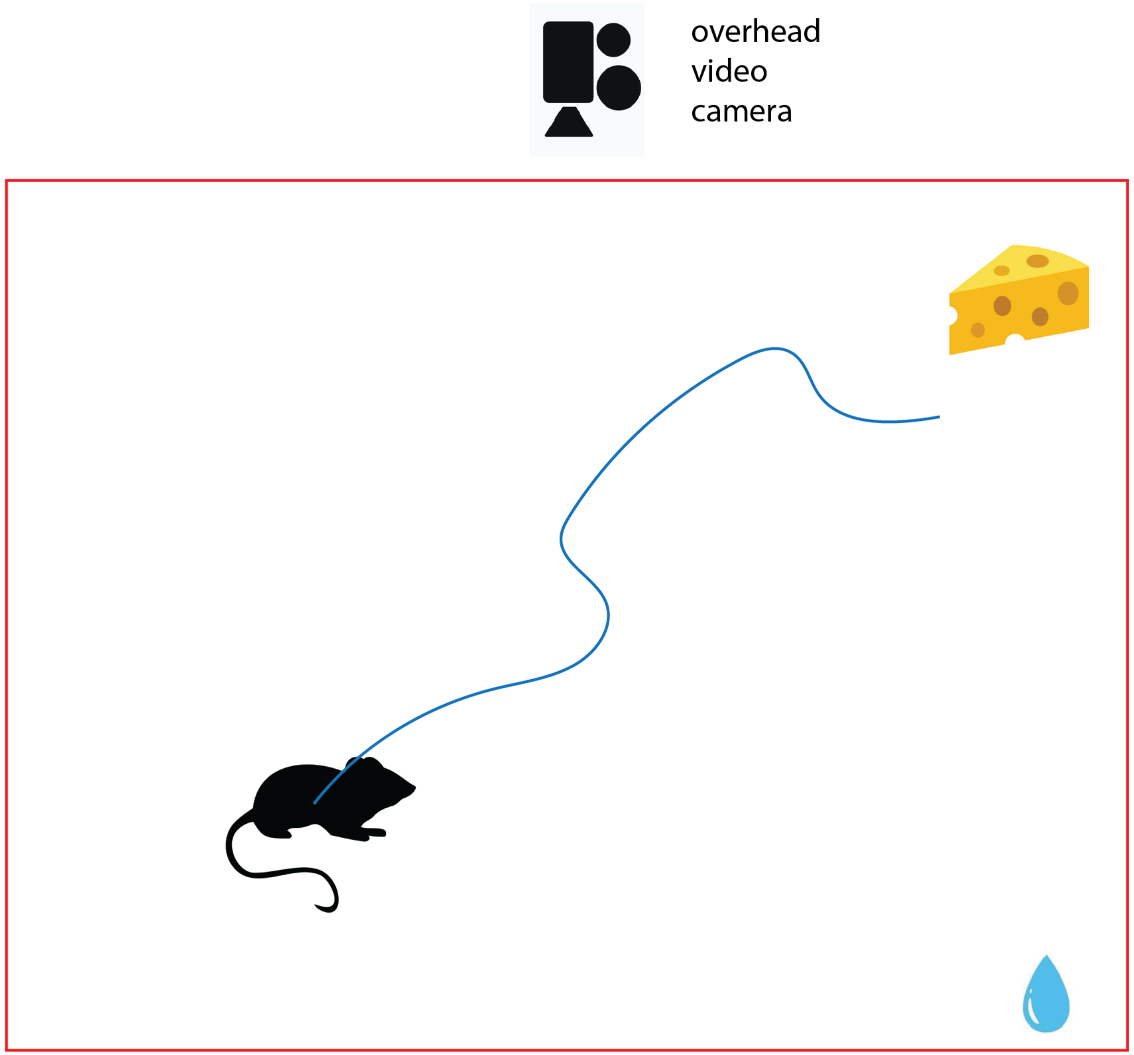
A day in the lab



A day in the lab

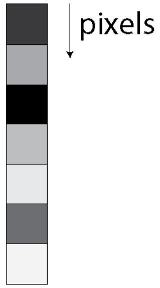


A day in the lab



Data: $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T]$

$\mathbf{y}_t =$



Adjacent $\boldsymbol{y}_t$ will be highly correlated (i.e not independent)

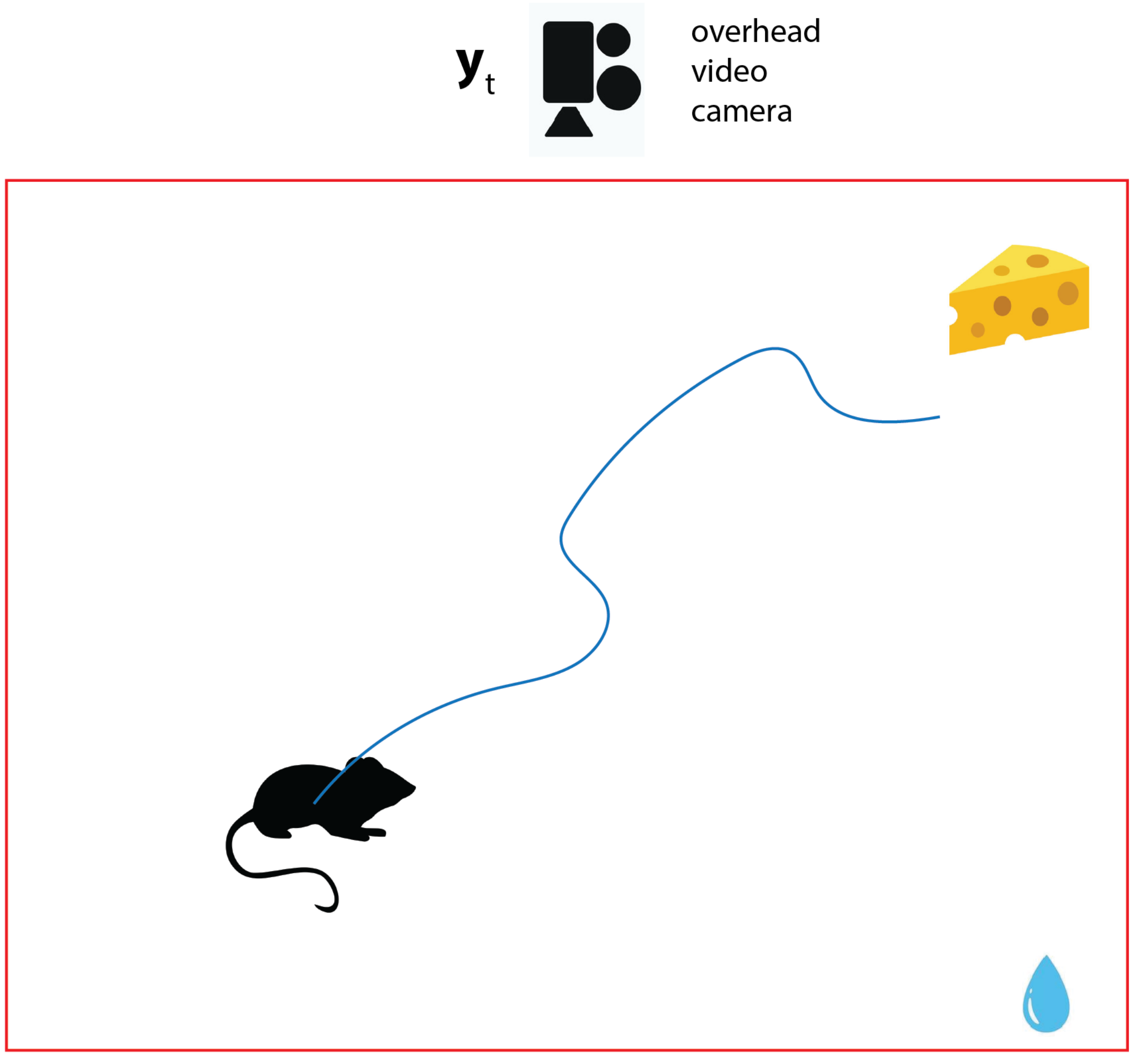
$\boldsymbol{y}_t$ is very high-dimensional with lots of redundant information

How can we leverage this to extract useful information?

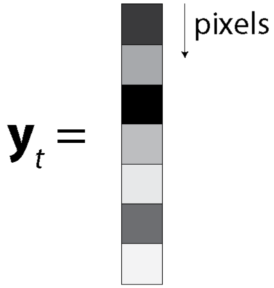
Outline

- Linear dynamical systems and state estimation
    - Applications in neuroscience
    - Model & problem definition
- Mathematical Preliminaries
    - A review of gaussian distributions
    - Bayes Rule
- Optimal state estimation - Kalman Filter
    - Prediction
    - Filtering
    - Smoothing

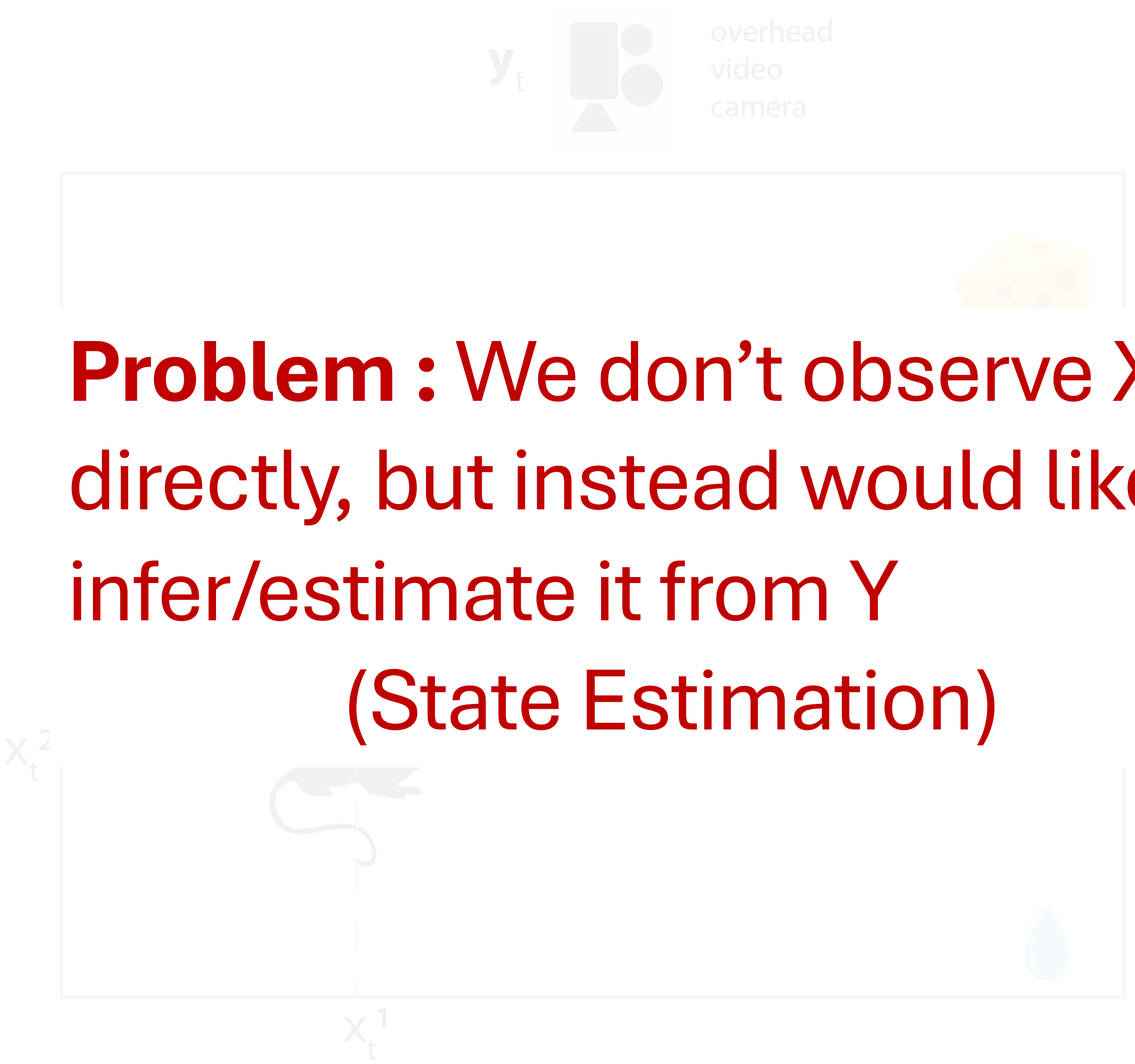
Motivation



Data: $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T]$



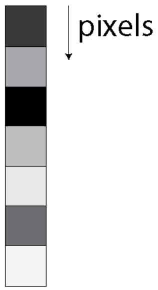
Motivation



Problem : We don't observe X directly, but instead would like to infer/estimate it from Y

(State Estimation)

Data: $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T]$

$\mathbf{y}_t =$ 

State: $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T]$

$$\boldsymbol{x}_t = \begin{bmatrix} \mathrm{x}_t^1 \\ \mathrm{x}_t^2 \end{bmatrix}$$

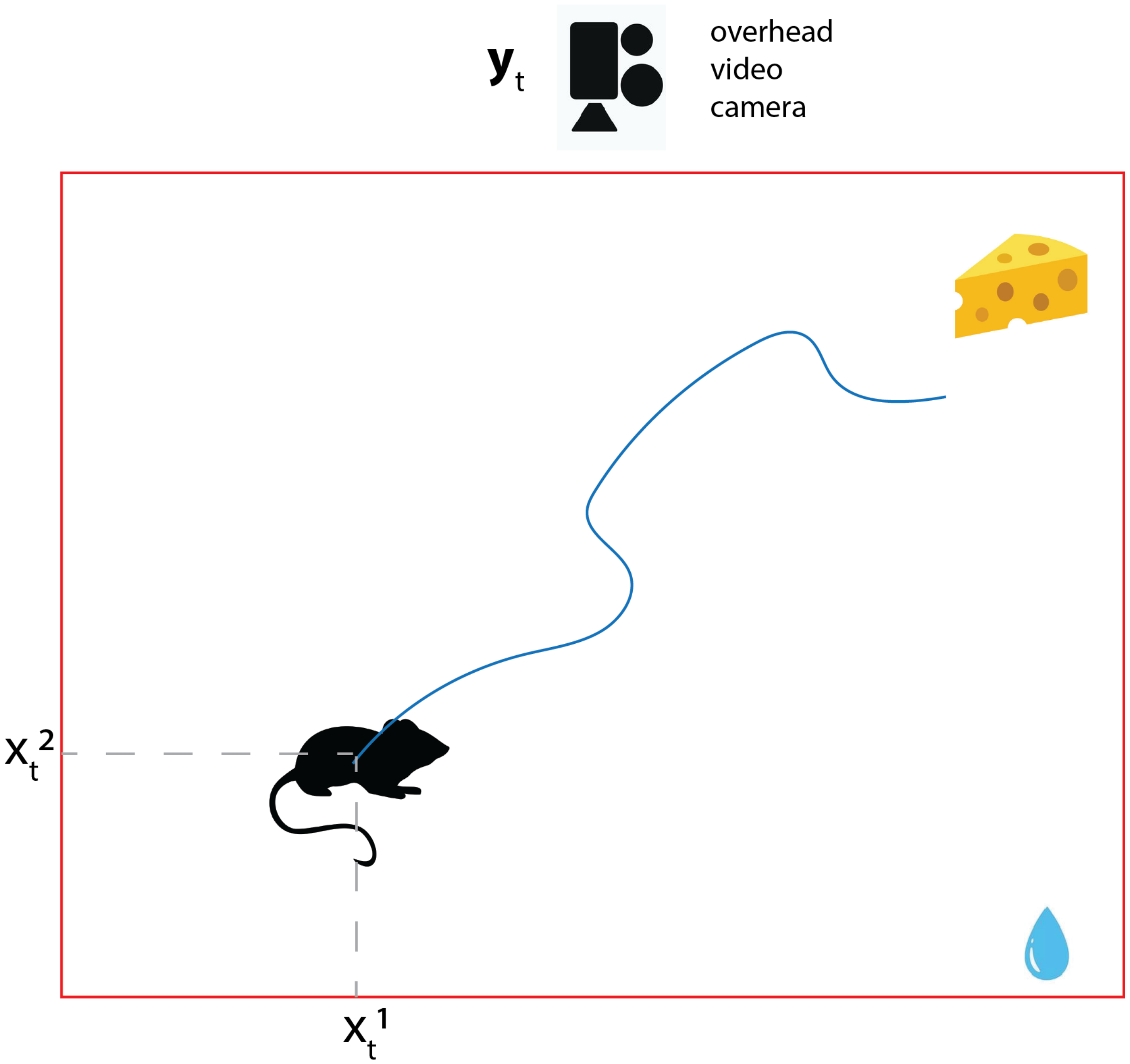
Position in 2D space

OR

$$\boldsymbol{x}_t = \begin{bmatrix} \mathrm{x}_t^1 \\ \mathrm{x}_t^2 \\ \dot{\mathrm{x}}_t^1 \\ \dot{\mathrm{x}}_t^2 \end{bmatrix}$$

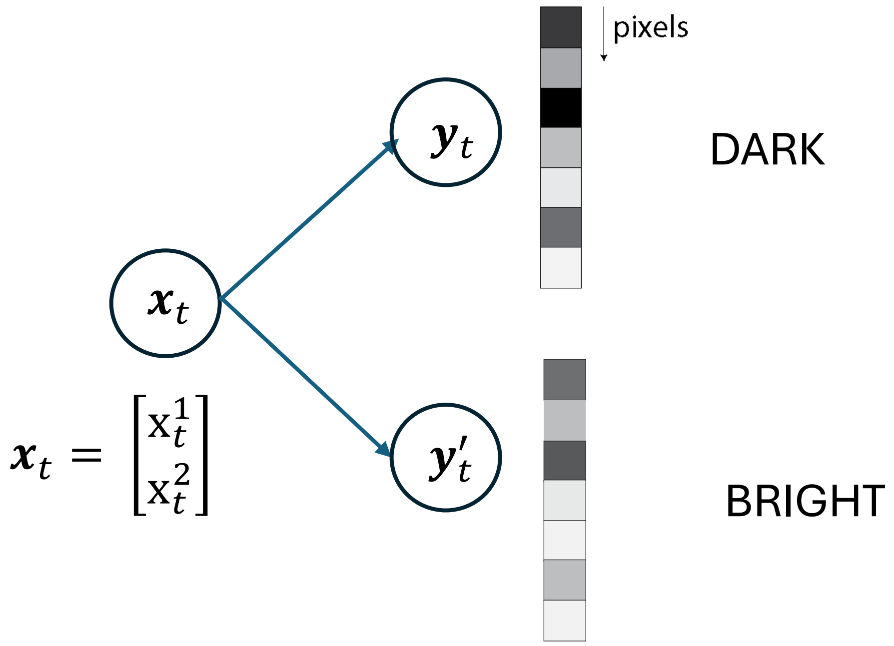
Position & Velocity in 2D space

The problem is ill-posed

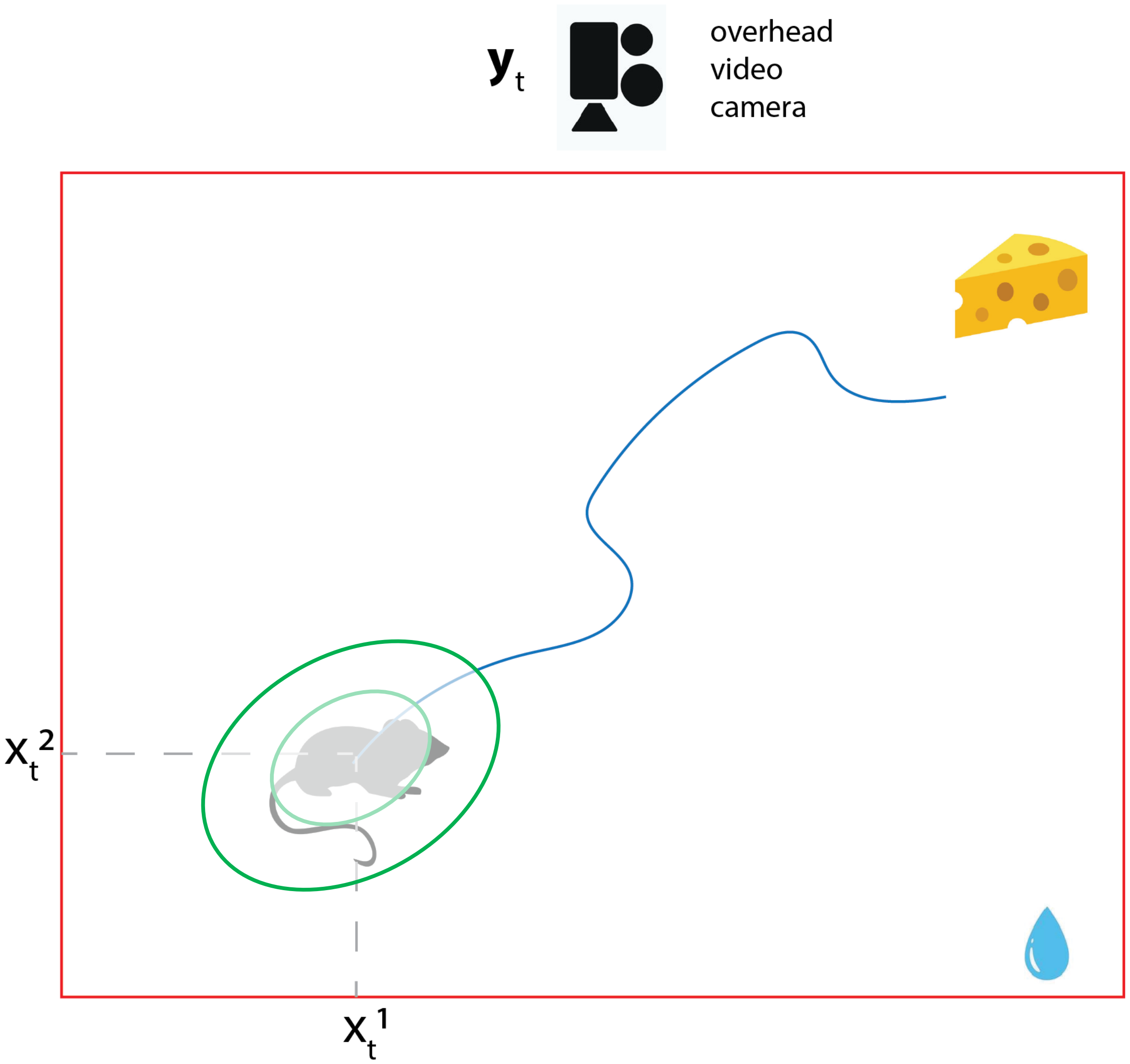


Data: $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T]$

State: $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T]$

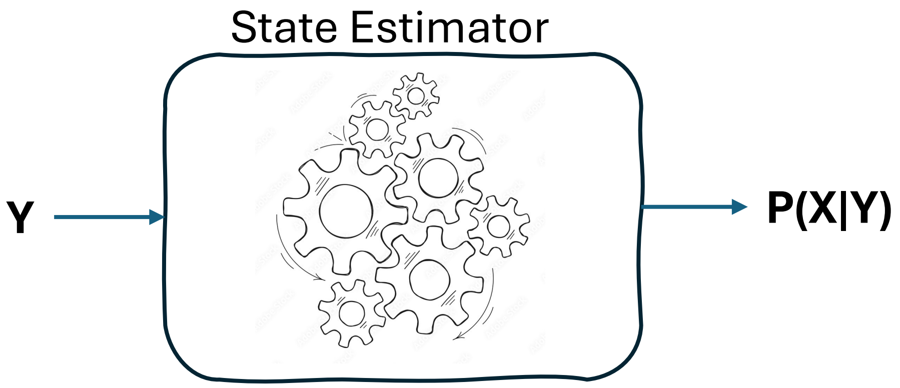


State Estimation requires probabilities



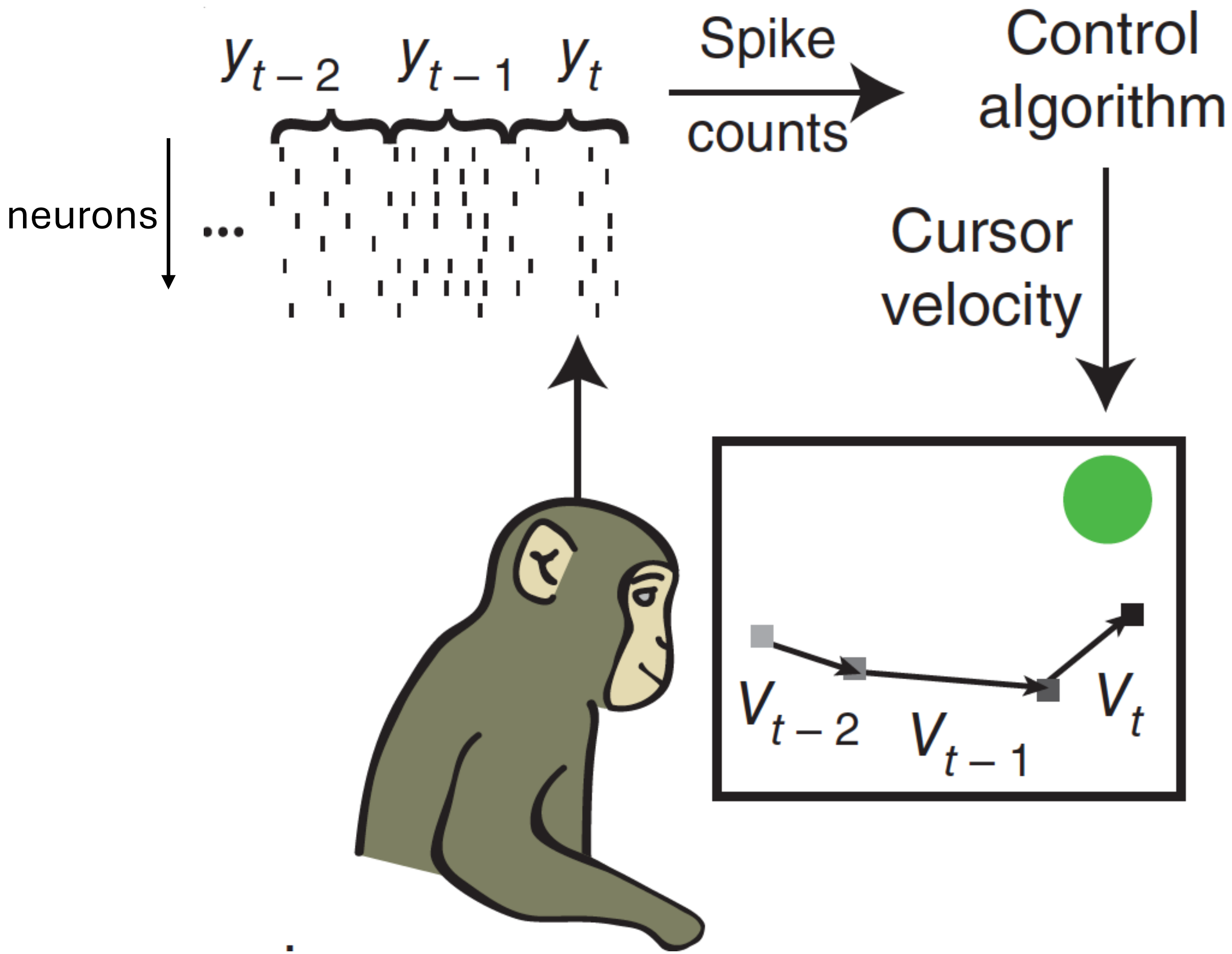
Data: $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T]$

State: $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T]$



We would like to infer a probability distribution over possible states (locations) that the animal could be in.

Application #2 – Brain Computer Interfaces



Gilja et al, 2012

Data: $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T]$

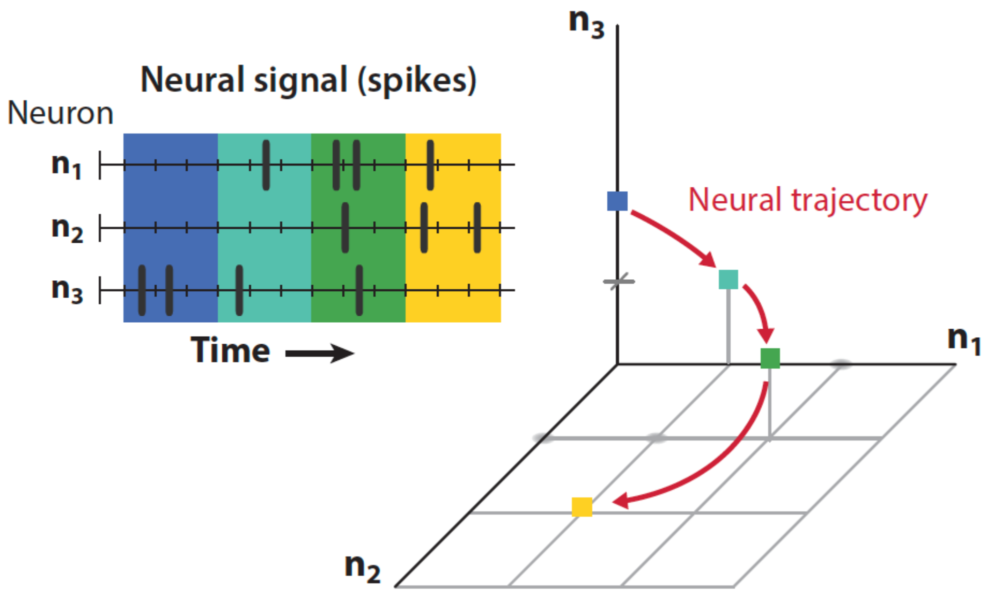
$$\boldsymbol{y}_t = \begin{bmatrix} \text{activity of neuron 1} \\ \text{activity of neuron 2} \\ \vdots \\ \text{activity of neuron 'N'} \end{bmatrix}$$

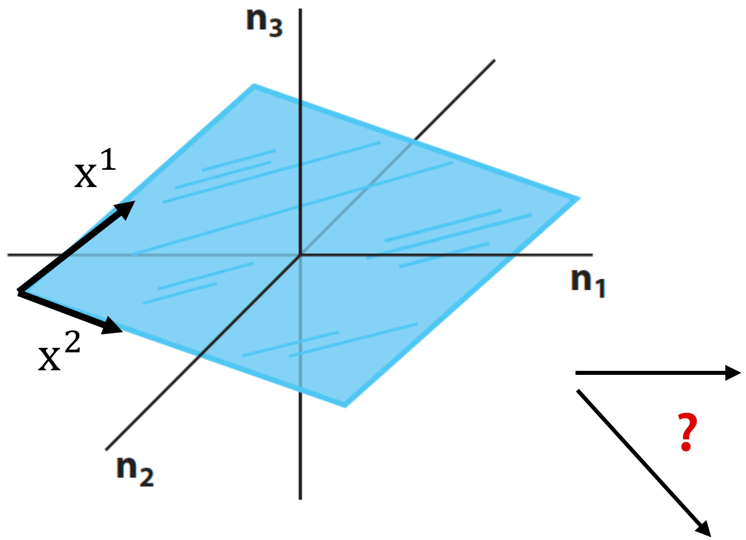
State: $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T]$

$$\boldsymbol{x}_t = \begin{bmatrix} \text{cursor vel. along } \rightarrow \\ \text{cursor vel. along } \uparrow \end{bmatrix}$$

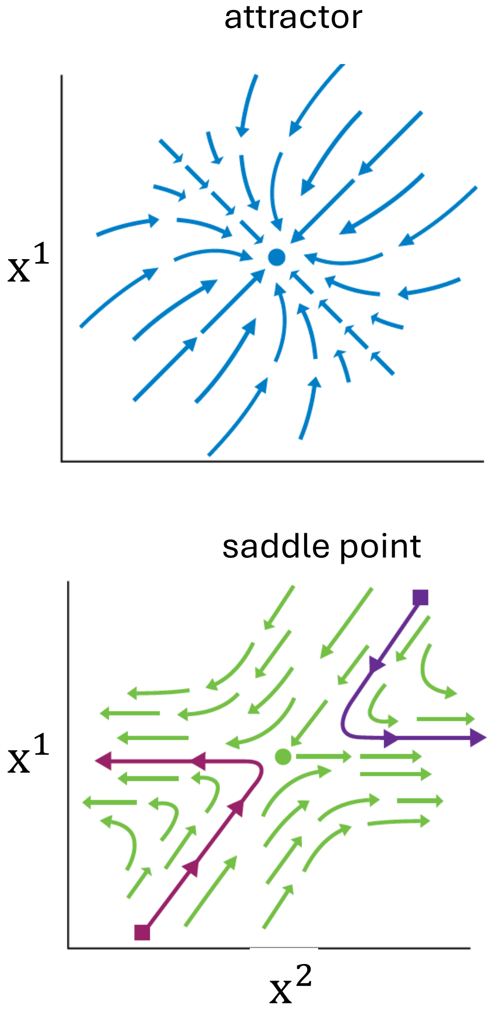
Goal: Infer/'Decode' cursor locations (in real-time) based on spiking activity.

Application #3 – Inferring population dynamics





Infer rules that govern evolution of X (dynamics)



Neural Data: $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T]$

$$\boldsymbol{y}_t = \begin{bmatrix} \text{activity of neuron 1} \\ \text{activity of neuron 2} \\ \vdots \\ \text{activity of neuron 'N'} \end{bmatrix}$$

Population State: $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T]$

$$\boldsymbol{x}_t = \begin{bmatrix} x_t^1 \\ x_t^2 \end{bmatrix}$$

A 'low-d' representation that captures activity patterns across a population

Vyas et al, 2020
Macke et al, 2013
Galgali et al, 2023

Specifying a model

pixels, neural activity, …

Data: $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T]$

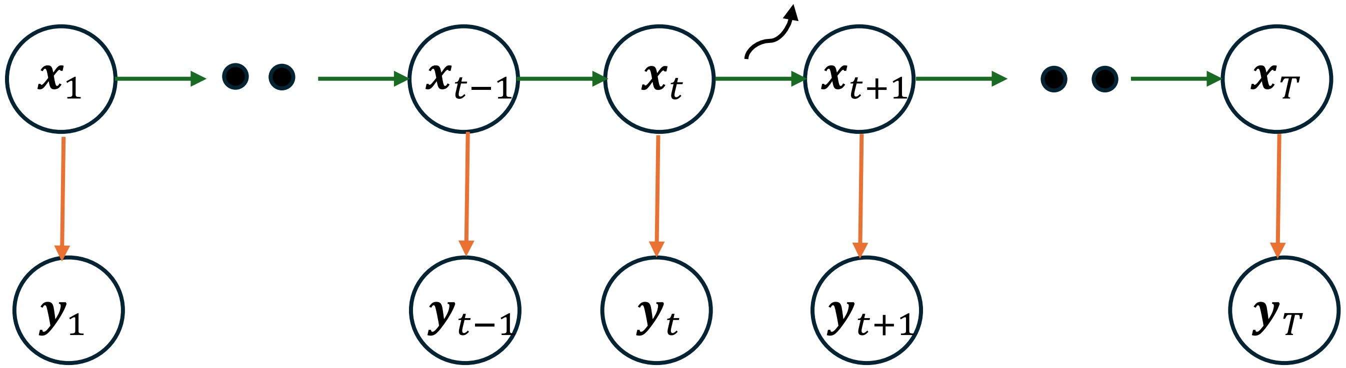
q-dimensional

location, cursor vel, neural pop. state

State: $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T]$

p-dimensional

'***Markov***' assumption



"Emission" model – specifies how $\boldsymbol{y}_t$ depends on $\boldsymbol{x}_t$ (probabilistically)

$$p(\boldsymbol{y}_t | \, \boldsymbol{x}_t)$$

"Dynamics" model – specifies how $\boldsymbol{x}_{t+1}$ depends on $\boldsymbol{x}_t$

$$p(\boldsymbol{x}_{t+1} | \, \boldsymbol{x}_t)$$

For a 'linear' dynamical system:

- Distributions are **gaussian**

random variable | mean | covariance matrix

$$p(\boldsymbol{x}_{t+1} | \, \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; \mathbf{A}\boldsymbol{x}_t \, , \mathbf{Q})$$

$$p(y_t | \, x_t) = \mathcal{N}(\boldsymbol{y}_t; \mathbf{H}\boldsymbol{x}_t \, , \mathbf{R})$$

$$p(x_1) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_0 \, , \mathbf{Q}_0)$$ "Initial Condition"

- **Linear** in parameters

$\mathbf{A}, \mathbf{Q}, \mathbf{Q}_0$ – p x p matrices

$\mathbf{H}$ – q x p matrix

$\mathbf{R}$ – q x q matrix

$\boldsymbol{\mu}_0$ – p x 1 vector

A ‘generative’ model for time series

pixels, neural activity, …

Data: $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T]$

q-dimensional

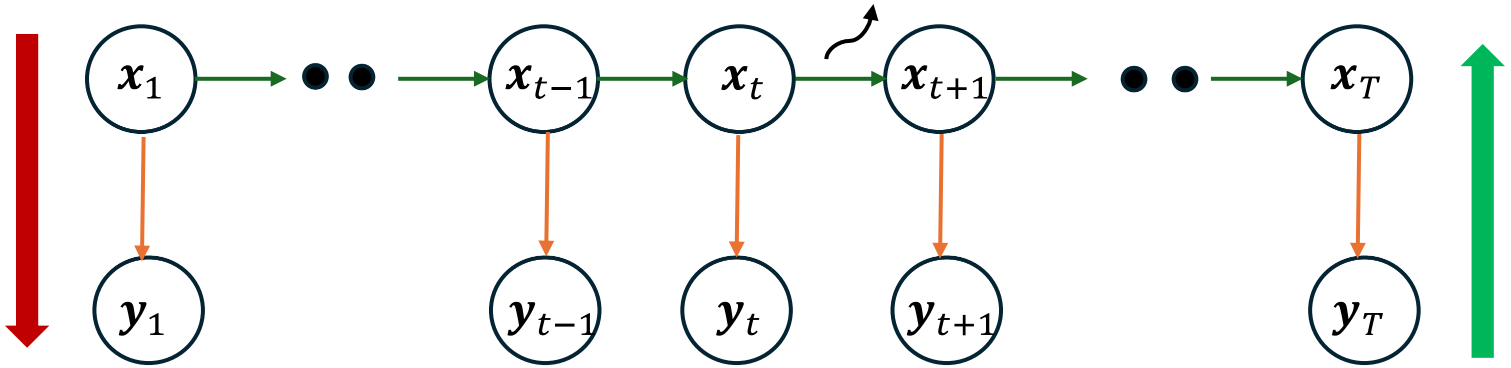
location, cursor vel, neural pop. state

State: $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T]$

p-dimensional

‘***Markov***’ assumption

Inference (“Bayes Rule”)



“Emission” model – specifies how $\boldsymbol{y}_t$ depends on $\boldsymbol{x}_t$ (probabilistically)

$$p(\boldsymbol{y}_t | \boldsymbol{x}_t)$$

“Dynamics” model – specifies how $\boldsymbol{x}_{t+1}$ depends on $\boldsymbol{x}_t$

$$p(\boldsymbol{x}_{t+1} | \boldsymbol{x}_t)$$

For a ‘linear’ dynamical system:

- Distributions are **gaussian**

random variable | mean | covariance matrix

$$p(\boldsymbol{x}_{t+1} | \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; \mathbf{A}\boldsymbol{x}_t, \mathbf{Q})$$

$$p(\boldsymbol{y}_t | x_t) = \mathcal{N}(\boldsymbol{y}_t; \mathbf{H}\boldsymbol{x}_t, \mathbf{R})$$

$$p(x_1) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_0, \mathbf{Q}_0)$$ “Initial Condition”

- **Linear** in parameters

$\mathbf{A}, \mathbf{Q}, \mathbf{Q}_0$ – p x p matrices

$\mathbf{H}$ – q x p matrix

$\mathbf{R}$ – q x q matrix

$\boldsymbol{\mu}_0$ – p x 1 vector

Outline

- Linear dynamical systems and state estimation
  - Applications in neuroscience
  - Defining the model

- Mathematical Preliminaries (*see whiteboard*)
  - Bayes Rule
  - A review of gaussian distributions

- Optimal state estimation - Kalman Filter
  - Prediction
  - Filtering
  - Smoothing

Alternative model definition

pixels, neural activity, …

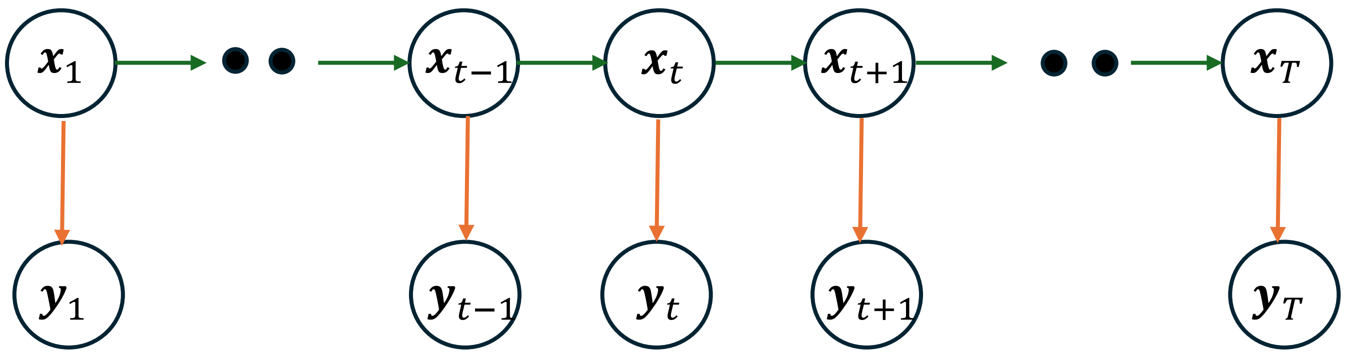
Data: $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T]$

q-dimensional

location, cursor vel, neural pop. state

State: $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T]$

p-dimensional



Follows from linearity and affine properties of gaussian distributions

For a 'linear' dynamical system:

random variable | mean | covariance matrix

$$p(\boldsymbol{x}_{t+1} | \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; \mathbf{A}\boldsymbol{x}_t, \mathbf{Q})$$

$$p(\boldsymbol{y}_t | \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{y}_t; \mathbf{H}\boldsymbol{x}_t, \mathbf{R})$$

$$p(\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_0, \mathbf{Q}_0)$$

'equivalent'

$$\boldsymbol{x}_{t+1} = \mathbf{A}\boldsymbol{x}_t + \boldsymbol{\varepsilon}_t$$
$$\boldsymbol{y}_t = \mathbf{H}\boldsymbol{x}_t + \boldsymbol{\eta}_t$$

$$p(\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_0, \mathbf{Q}_0)$$
$$p(\varepsilon_t) = \mathcal{N}(\epsilon; \mathbf{0}, \mathbf{Q})$$
$$p(\eta_t) = \mathcal{N}(\eta; \mathbf{0}, \mathbf{R})$$

Outline

- Linear dynamical systems and state estimation
  - Applications in neuroscience
  - Defining the model
- Mathematical Preliminaries (*see whiteboard*)
  - Bayes Rule
  - A review of gaussian distributions
- Optimal state estimation - Kalman Filter
  - Prediction
  - Filtering
  - Smoothing

Optimal State Estimation

pixels, neural activity, … Data: $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T]$

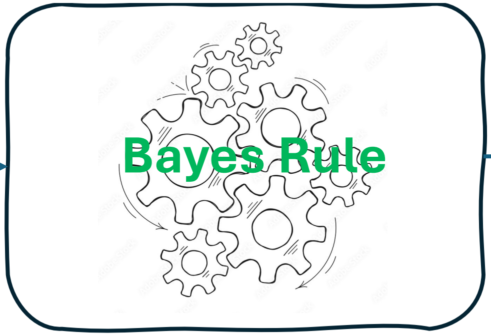
q-dimensional

location, cursor vel,… State: $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T]$

p-dimensional

Kalman Filter + Smoother

$\boldsymbol{y}_1, \dots, \boldsymbol{y}_T$



$\boldsymbol{P}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T \mid \boldsymbol{y}_1, \dots, \boldsymbol{y}_T)$

What if q and T are large?

Recursively apply Bayes rule as new data come in

Decomposes into 3 fundamental steps

1. **Prediction** – Predict what the next best state is given the data I have observed so far
2. **Filtering** – Use disparity between model prediction and new measurement to update prediction
3. **Smoothing** – Use knowledge of entire sequence to further refine estimate of state.

$$\boldsymbol{P}(\boldsymbol{x}_t \mid \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_{t-1})$$

$$\boldsymbol{P}(\boldsymbol{x}_t \mid \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_t)$$

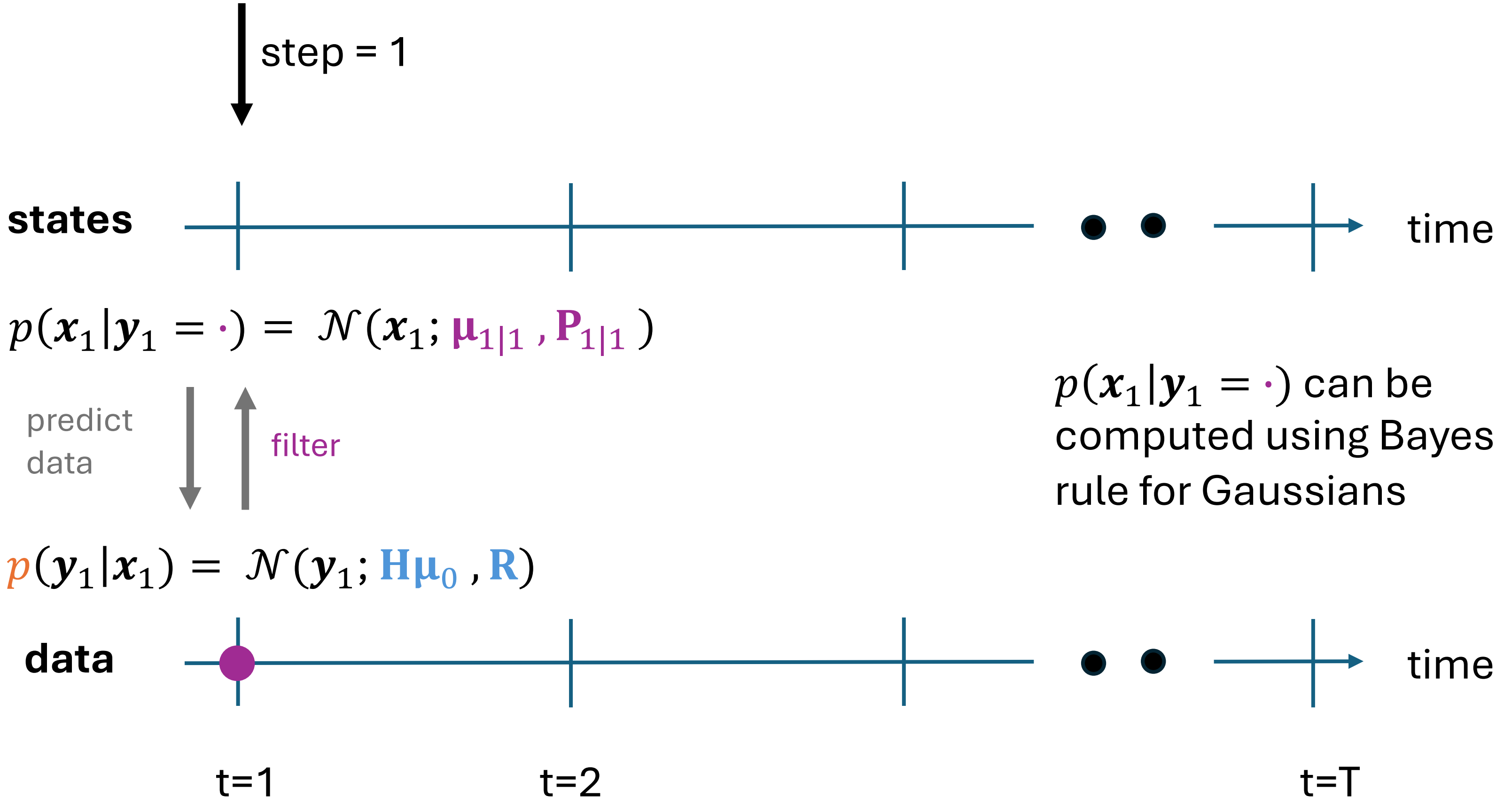
$$\boldsymbol{P}(\boldsymbol{x}_t \mid \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T)$$

Optimal in what sense ?

- maximize probability ('likelihood') of observing the data : $\boldsymbol{P}(\boldsymbol{y}_1, \dots, \boldsymbol{y}_T)$

- minimize discrepancy between LDS model prediction ($\hat{\boldsymbol{y}}_t$) and true observation ($\boldsymbol{y}_t$) but in a recursive least squares framework.

Kalman prediction & filtering



$p(\boldsymbol{x}_1|\boldsymbol{y}_1 = \cdot) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_{1|1}, \mathbf{P}_{1|1})$

$p(\boldsymbol{y}_1|\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{y}_1; \mathbf{H}\boldsymbol{\mu}_0, \mathbf{R})$

$p(\boldsymbol{x}_1|\boldsymbol{y}_1 = \cdot)$ can be computed using Bayes rule for Gaussians

Model definition

$p(\boldsymbol{x}_{t+1}| \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; \mathbf{A}\boldsymbol{x}_t, \mathbf{Q})$

$p(\boldsymbol{y}_t| \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{y}_t; \mathbf{H}\boldsymbol{x}_t, \mathbf{R})$

$p(\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_0, \mathbf{Q}_0)$

At current step:

$\boldsymbol{\mu}_{1|1} = ?$

$\mathbf{P}_{1|1} = ?$

Kalman prediction & filtering

step = 1

$p(\boldsymbol{x}_1 | \boldsymbol{y}_1 = \cdot) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_{1|1}, \mathbf{P}_{1|1})$

predict data

filter

$p(\boldsymbol{y}_1 | \boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{y}_1; \mathbf{H}\boldsymbol{\mu}_0, \mathbf{R})$

$p(\boldsymbol{x}_1 | \boldsymbol{y}_1 = \cdot)$ can be computed using Bayes rule for Gaussians

Model definition

$p(\boldsymbol{x}_{t+1} | \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; \mathbf{A}\boldsymbol{x}_t, \mathbf{Q})$

$p(\boldsymbol{y}_t | \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{y}_t; \mathbf{H}\boldsymbol{x}_t, \mathbf{R})$

$p(\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_0, \mathbf{Q}_0)$

At current step:

'residual' (data – pred)

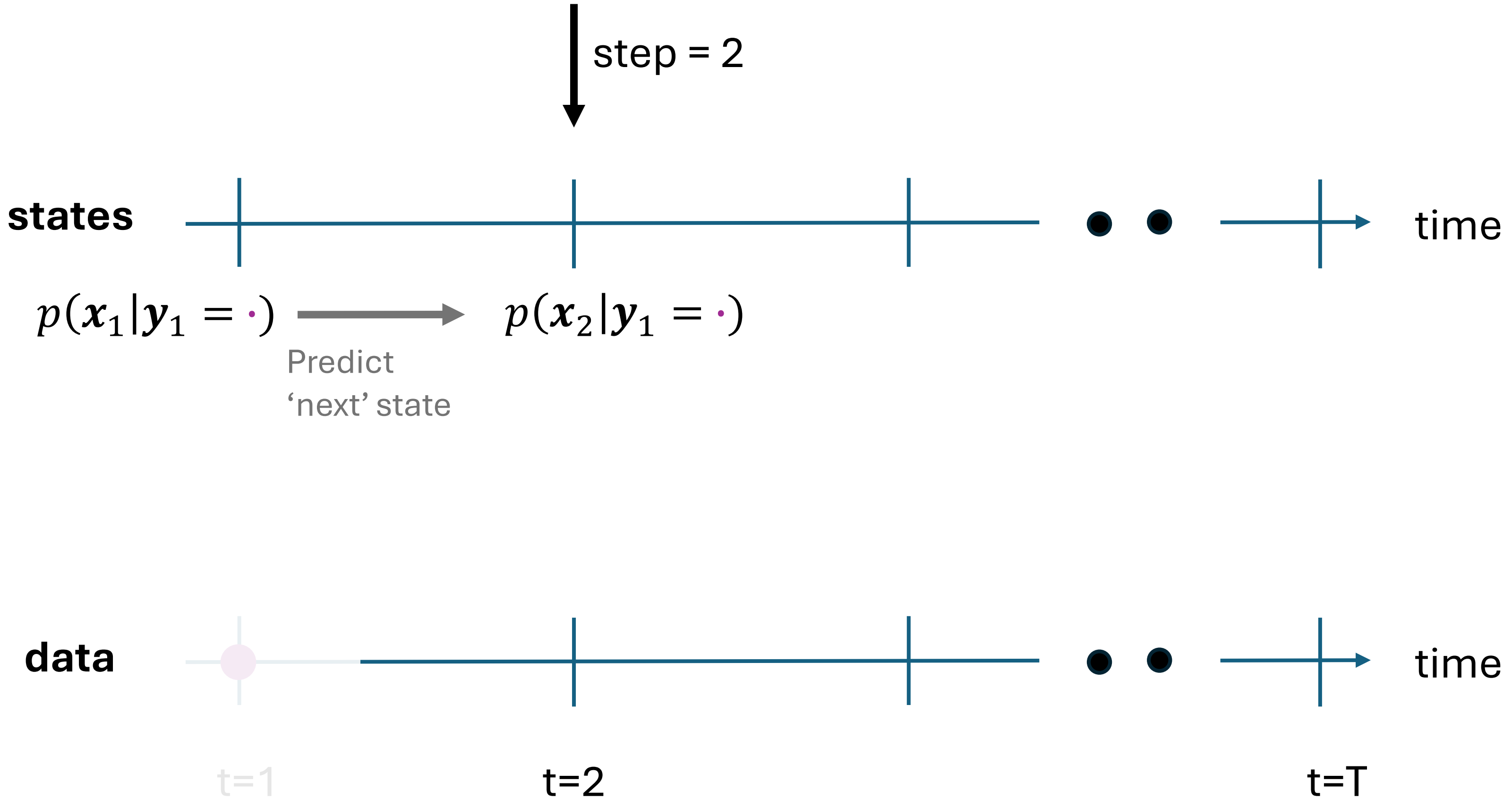
$$\boldsymbol{\mu}_{1|1} = \boldsymbol{\mu}_0 + \mathbf{K}_1(\boldsymbol{y}_1 - \mathbf{H}\boldsymbol{\mu}_0)$$

$$\mathbf{P}_{1|1} = (\mathbf{I} - \mathbf{K}_1\mathbf{H})\,\mathbf{Q}_0$$

$$\mathbf{K}_1 = \mathbf{Q}_0\mathbf{H}'(\mathbf{H}\mathbf{Q}_0\mathbf{H}' + \mathbf{R})^{-1}$$

Kalman 'gain'

Kalman prediction & filtering



Model definition

$p(\boldsymbol{x}_{t+1}|\,\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; \mathbf{A}\boldsymbol{x}_t\,, \mathbf{Q})$

$p(\boldsymbol{y}_t|\,\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{y}_t; \mathbf{H}\boldsymbol{x}_t\,, \mathbf{R})$

$p(\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_0\,, \mathbf{Q}_0)$

From previous step:

$p(\boldsymbol{x}_1|\boldsymbol{y}_1 = \cdot) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_{1|1}\,, \mathbf{P}_{1|1})$

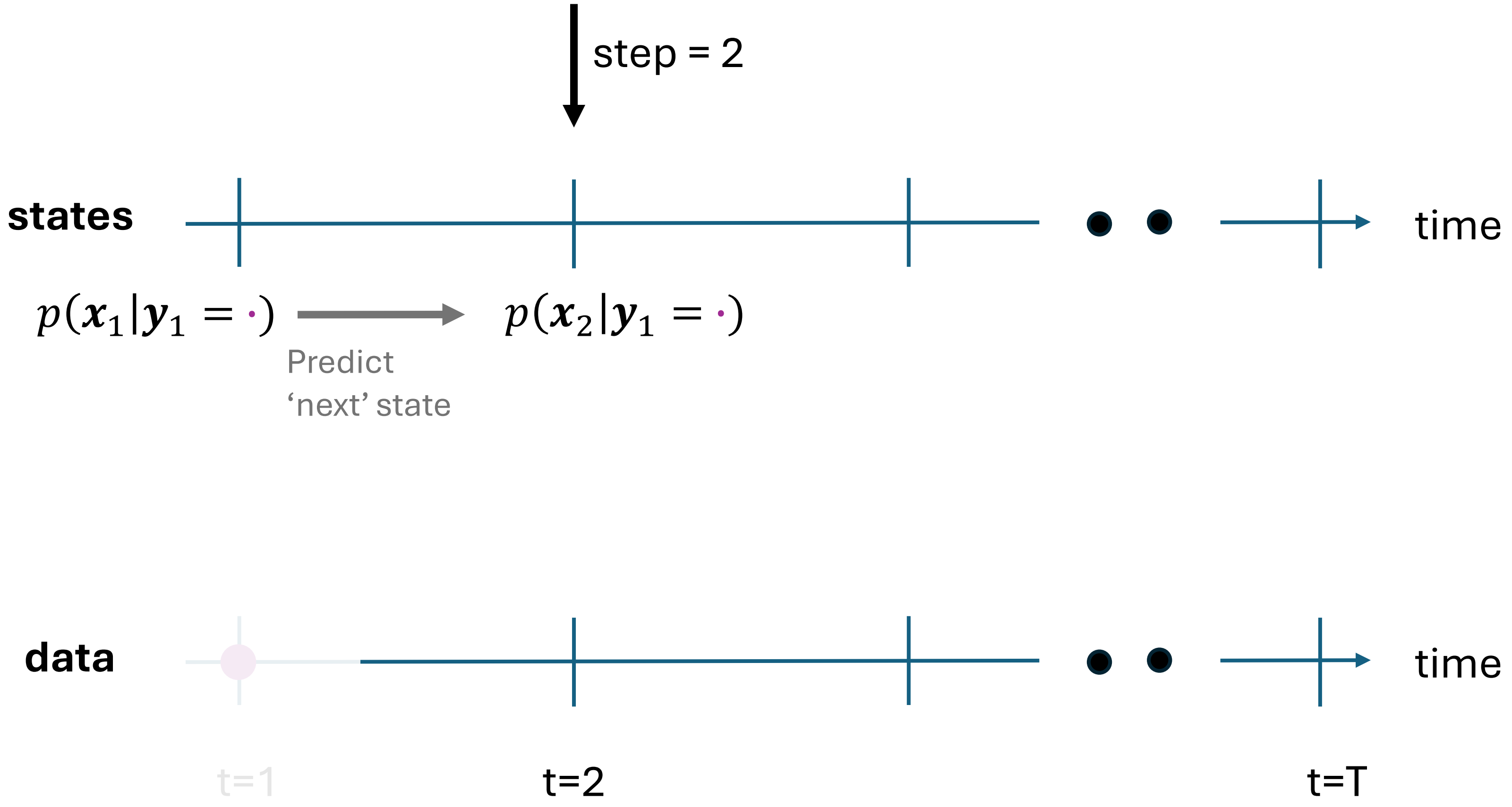
At current step:

$\boldsymbol{\mu}_{2|1} = ?$

$\boldsymbol{P}_{2|1} = ?$

$$p(\boldsymbol{x}_2|\boldsymbol{y}_1 = \cdot) = \int p(\boldsymbol{x}_2|\,\boldsymbol{x}_1)p(\boldsymbol{x}_1|\boldsymbol{y}_1 = \cdot)d\boldsymbol{x}_1 = \mathcal{N}(\boldsymbol{x}_2; \boldsymbol{\mu}_{2|1}\,, \mathbf{P}_{2|1})$$

Kalman prediction & filtering



Model definition

$$p(\boldsymbol{x}_{t+1}|\,\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; \mathbf{A}\boldsymbol{x}_t\,, \mathbf{Q})$$

$$p(\boldsymbol{y}_t|\,\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{y}_t; \mathbf{H}\boldsymbol{x}_t\,, \mathbf{R})$$

$$p(\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_0\,, \mathbf{Q}_0)$$

From previous step:

$$p(\boldsymbol{x}_1|\boldsymbol{y}_1 = \cdot) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_{1|1}\,, \mathbf{P}_{1|1})$$

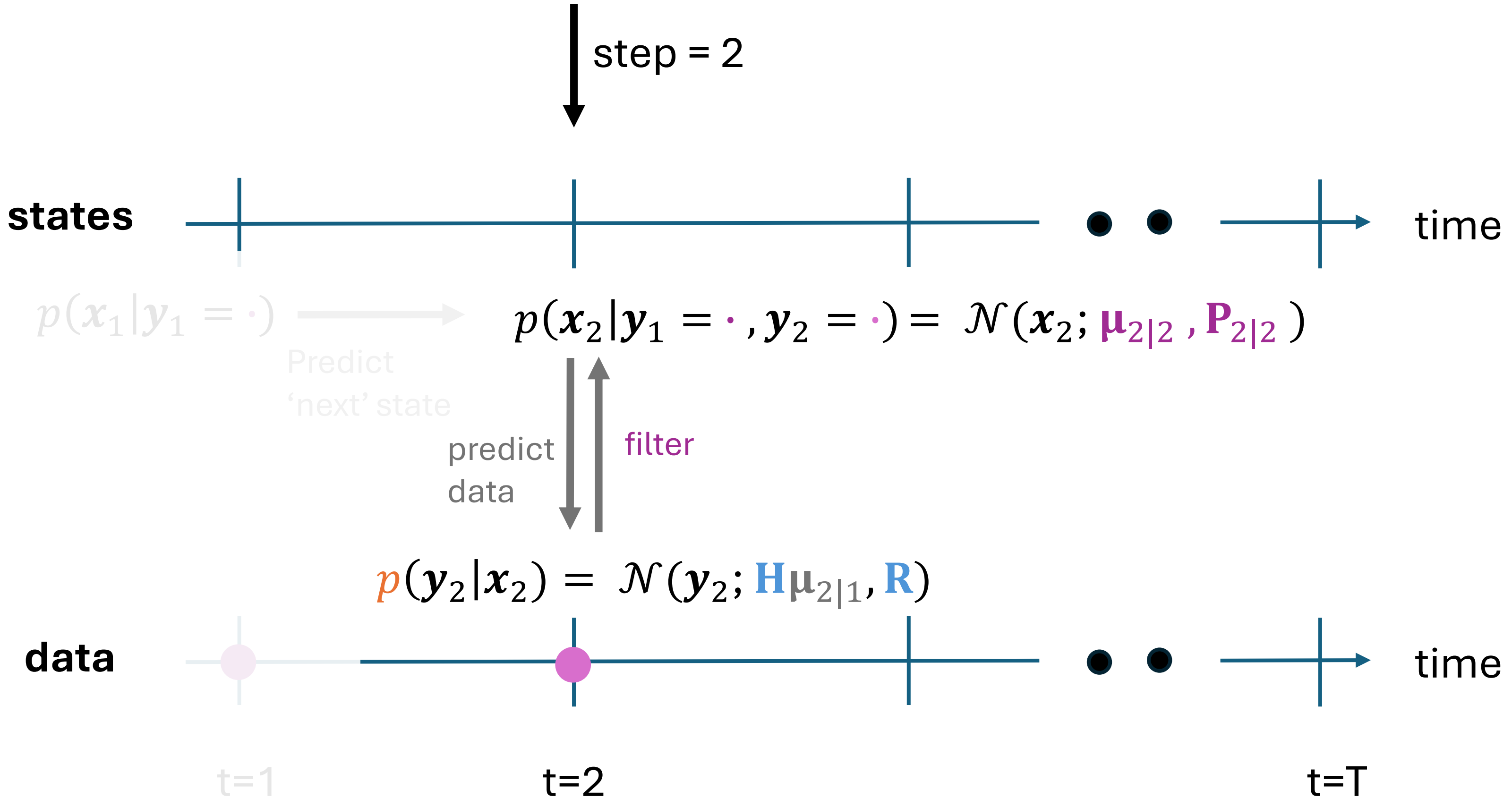
At current step:

$$\boldsymbol{\mu}_{2|1} = \mathbf{A}\boldsymbol{\mu}_{1|1}$$

$$\boldsymbol{P}_{2|1} = \mathbf{A}\mathbf{P}_{1|1}\mathbf{A}' + \mathbf{Q}$$

$$p(\boldsymbol{x}_2|\boldsymbol{y}_1 = \cdot) = \int p(\boldsymbol{x}_2|\,\boldsymbol{x}_1)p(\boldsymbol{x}_1|\boldsymbol{y}_1 = \cdot)d\boldsymbol{x}_1 = \mathcal{N}(\boldsymbol{x}_2; \boldsymbol{\mu}_{2|1}\,, \mathbf{P}_{2|1})$$

Kalman prediction & filtering



Model definition

$p(x_{t+1}| x_t) = \mathcal{N}(x_{t+1}; Ax_t, Q)$

$p(y_t| x_t) = \mathcal{N}(y_t; Hx_t, R)$

$p(x_1) = \mathcal{N}(x_1; \mu_0, Q_0)$

From previous step:

$p(x_1|y_1 = \cdot) = \mathcal{N}(x_1; \mu_{1|1}, P_{1|1})$

At current step:

$\mu_{2|1} = A\mu_{1|1}$

$P_{2|1} = AP_{1|1}A' + Q$

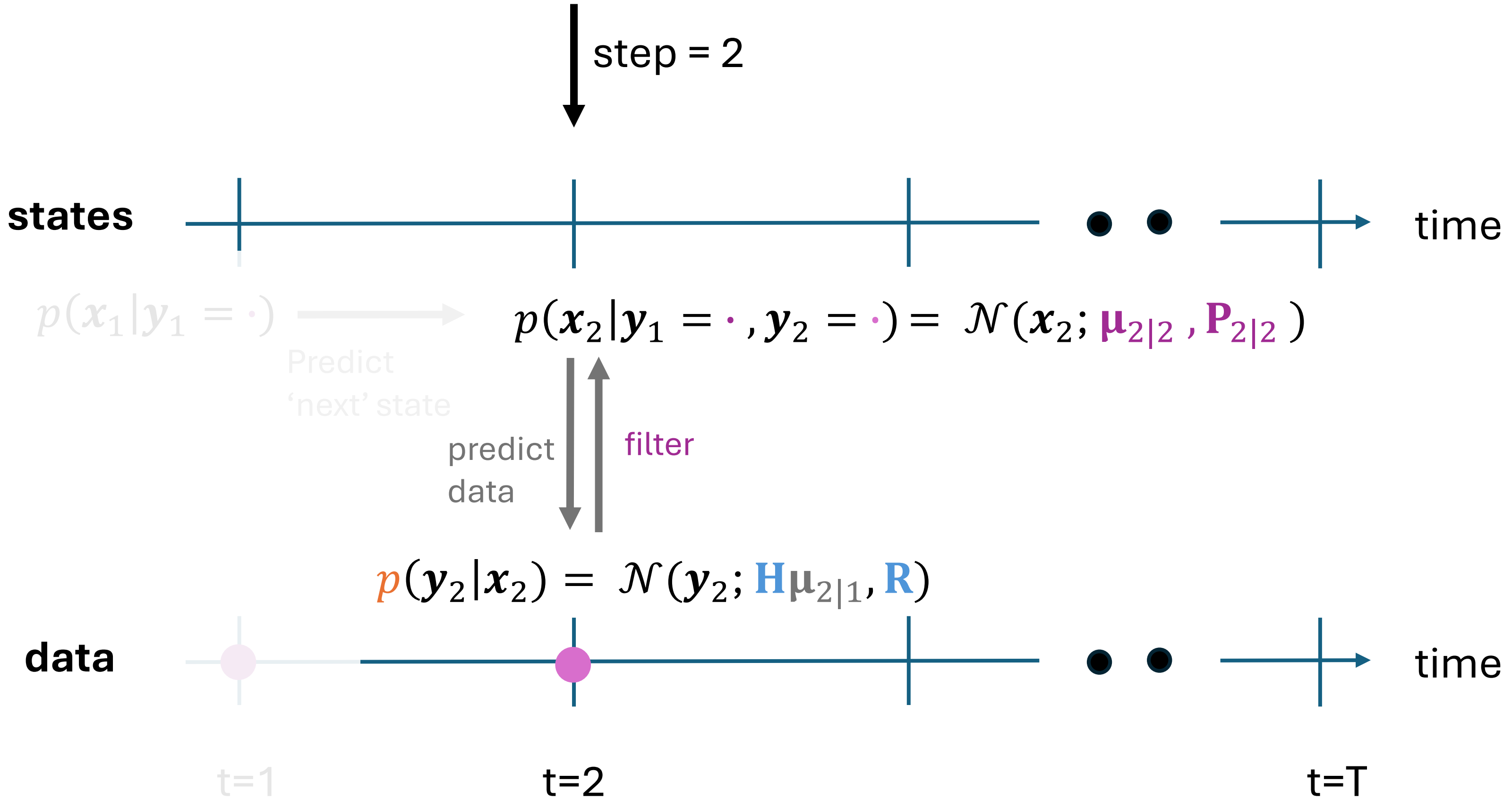
$\mu_{2|2} = ?$

$P_{2|2} = ?$

But 'predict data + filter' is analogous to what we did in Step 1!

'Initial' belief at step 2 is $\mathcal{N}(x_2; \mu_{2|1}, P_{2|1})$ instead of $\mathcal{N}(x_1; \mu_0, Q_0)$

Kalman prediction & filtering



But 'predict data + filter' is analogous to what we did in Step 1!

'Initial' belief at step 2 is $\mathcal{N}(\boldsymbol{x}_2; \boldsymbol{\mu}_{2|1}, \mathbf{P}_{2|1})$ instead of $\mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_0, \mathbf{Q}_0)$

Model definition

$$p(\boldsymbol{x}_{t+1}|\,\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; \mathbf{A}\boldsymbol{x}_t, \mathbf{Q})$$

$$p(\boldsymbol{y}_t|\,\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{y}_t; \mathbf{H}\boldsymbol{x}_t, \mathbf{R})$$

$$p(\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_0, \mathbf{Q}_0)$$

From previous step:

$$p(\boldsymbol{x}_1|\boldsymbol{y}_1 = \cdot) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_{1|1}, \mathbf{P}_{1|1})$$

At current step:

$$\boldsymbol{\mu}_{2|1} = \mathbf{A}\boldsymbol{\mu}_{1|1}$$

$$P_{2|1} = \mathbf{A}\mathbf{P}_{1|1}\mathbf{A}' + \mathbf{Q}$$

$$\boldsymbol{\mu}_{2|2} = \boldsymbol{\mu}_{2|1} + \mathbf{K}_2(\boldsymbol{y}_2 - \mathbf{H}\boldsymbol{\mu}_{2|1})$$

$$P_{2|2} = (\mathbf{I} - \mathbf{K}_2\mathbf{H})P_{2|1}$$

$$\mathbf{K}_2 = P_{2|1}\mathbf{H}'\left(\mathbf{H}P_{2|1}\mathbf{H}' + \mathbf{R}\right)^{-1}$$

Kalman prediction & filtering

Let, $\boldsymbol{P}(\boldsymbol{x}_{t-1}\,|\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_{t-1}) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_{t-1|t-1}, \mathbf{P}_{t-1|t-1})$

1. Predict next state using 'dynamics' model:

$$\boldsymbol{\mu}_{t|t-1} = \mathbf{A}\boldsymbol{\mu}_{t-1|t-1} \qquad (\boldsymbol{\mu}_{1|0} = \boldsymbol{\mu}_0)$$

$$\boldsymbol{P}_{t|t-1} = \mathbf{A}\mathbf{P}_{t-1|t-1}\mathbf{A}' + \mathbf{Q} \qquad (\boldsymbol{P}_{1|0} = \mathbf{Q}_0)$$

2. Compute Kalman gain at 't'

$$\mathbf{K}_t = \boldsymbol{P}_{t|t-1}\mathbf{H}'\left(\mathbf{H}\boldsymbol{P}_{t|t-1}\mathbf{H}' + \mathbf{R}\right)^{-1}$$

3. Update using observed data & 'emission' model

$$\boldsymbol{\mu}_{t|t} = \boldsymbol{\mu}_{t|t-1} + \mathbf{K}_t(\boldsymbol{y}_t - \mathbf{H}\boldsymbol{\mu}_{t|t-1})$$

$$\boldsymbol{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\boldsymbol{P}_{t|t-1}$$

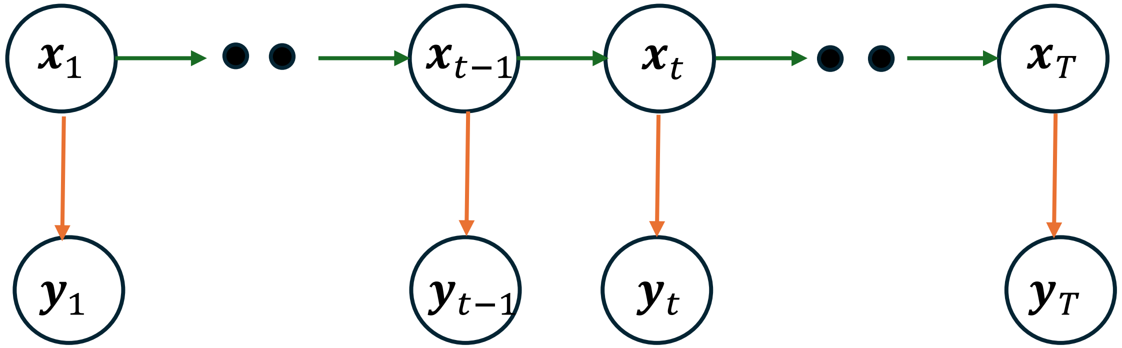
Online algorithm. Really useful for real-time application (e.g BCIs, tracking)

Model definition

$$p(\boldsymbol{x}_{t+1}|\,\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; \mathbf{A}\boldsymbol{x}_t, \mathbf{Q})$$

$$p(\boldsymbol{y}_t|\,\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{y}_t; \mathbf{H}\boldsymbol{x}_t, \mathbf{R})$$

$$p(\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_0, \mathbf{Q}_0)$$



Intuition

If model is very uncertain about measurements (i.e large $\mathbf{R}$ relative to $\mathbf{Q}$), then ignore data and use output of prediction as best state

Kalman Smoothing

Let, $\boldsymbol{P}(\boldsymbol{x}_{t+1} \mid \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T) = \mathcal{N}(\boldsymbol{x}_{t+1}; \boldsymbol{\mu}_{t+1|T}, \mathbf{P}_{t+1|T})$

1. Start at t = T and **work backwards**

2. Compute reverse Kalman gain at 't'

$$\boldsymbol{J}_t = \boldsymbol{P}_{t|t} \mathbf{A}' \left(\boldsymbol{P}_{t|t-1}\right)^{-1}$$

3. Update

$$\boldsymbol{\mu}_{t|T} = \boldsymbol{\mu}_{t|t} + \boldsymbol{J}_t(\boldsymbol{\mu}_{t+1|T} - \boldsymbol{\mu}_{t+1|t})$$

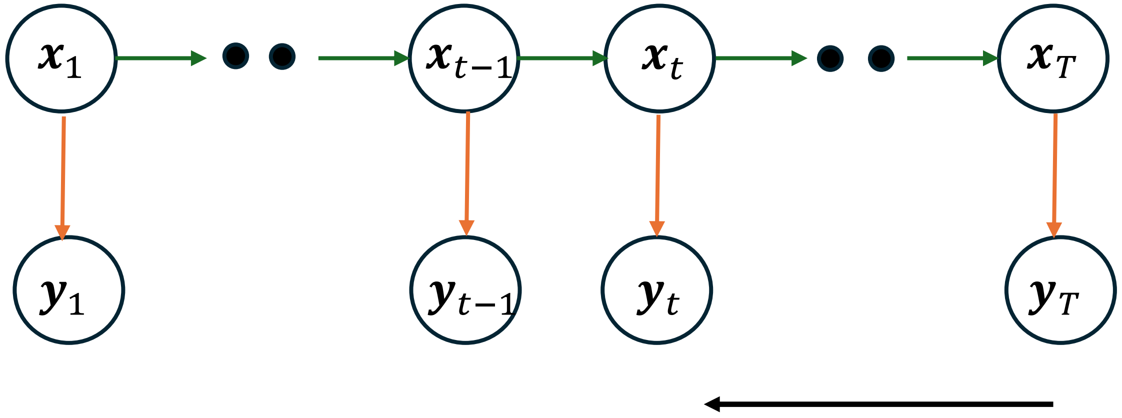
$$\mathbf{P}_{t|T} = \boldsymbol{P}_{t|t} + \boldsymbol{J}_t\left(\mathbf{P}_{t+1|T} - \boldsymbol{P}_{t+1|t}\right)\boldsymbol{J}_t{}'$$

Model definition

$p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; \mathbf{A}\boldsymbol{x}_t, \mathbf{Q})$

$p(\boldsymbol{y}_t \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{y}_t; \mathbf{H}\boldsymbol{x}_t, \mathbf{R})$

$p(\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_0, \mathbf{Q}_0)$



Offline algorithm. Can be computed only once entire data has been using. Useful for refinement.

Just need the quantities that were computed in the forward direction (prediction + filtering)

Summary

- Time-series data are ubiquitous in neuroscience.

- An important problem – state estimation
  - Many applications – behavior tracking, neural/BCI decoding, inferring internal states, and many more..
  - Can be formulated as a recursive Bayesian estimation problem

- Bayesian state estimation is a **general** algorithm
  - For fixed model parameters, estimate distribution over states given data.
  - We looked at a specific case – linear, gaussian model – gives us an analytical solution (Kalman filtering + smoothing)
  - Dynamics/Emission models can be more general (think neural networks) – this requires sophisticated algorithms that approximate Bayes Rule.

Further Reading

Concepts

- Pattern Recognition & Machine Learning (Chris Bishop)
  - Chapter 2 – for gaussian distributions
  - Chapter 13 – for linear dynamical systems//hidden markov models
- A nice visualization of Gaussians and associated concepts
  - https://distill.pub/2019/visual-exploration-gaussian-processes/

Applications:

- https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python (code notebook)
- Macke et al, NeurIPS 2013, Empirical models of spiking in neural populations
- Gilja et al, Nat. Neuro 2012, A high performance neural prosthesis enabled by control algorithm design.