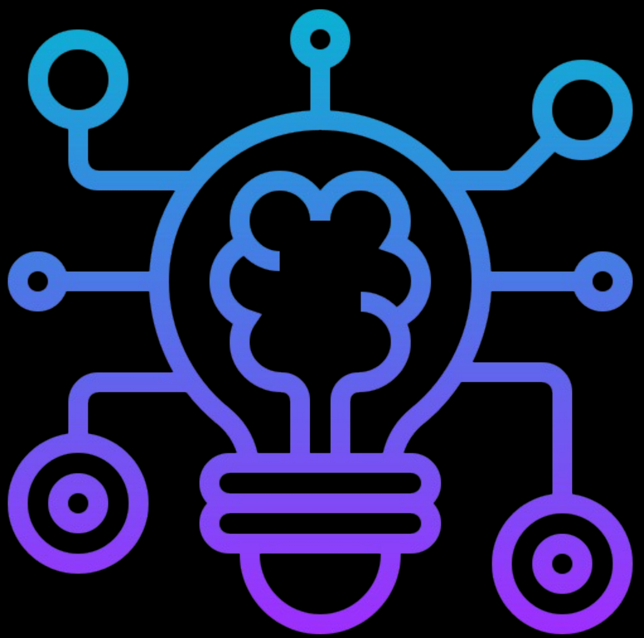


Reasoning (V)

Mingsheng Long

Tsinghua University

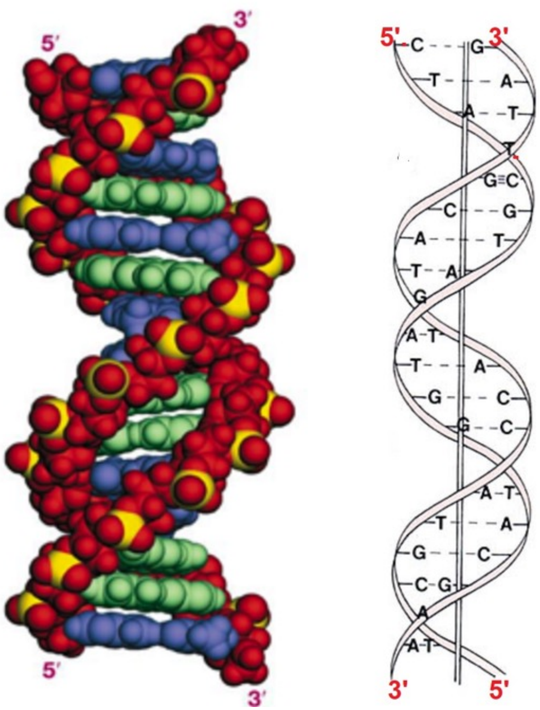
Spring 2023

Outline

- Hidden Markov Model
  - **Markov Model**
  - Hidden Markov Model
  - Inference for Hidden Markov Model
    - Viterbi Algorithm
    - Forward-Backward Algorithm
- Linear Dynamical System
  - Kalman Filter
  - Particle Filter

Sequential Data

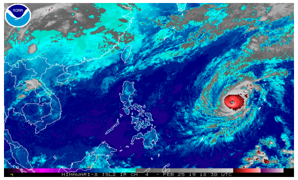
- Sequential data have certain order and dependency.



DNA sequence

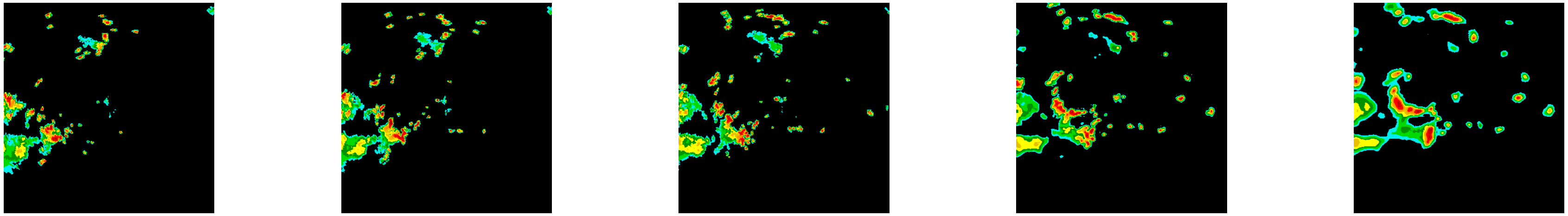
Title: Star's Tux Promise Draws Megyn Kelly's Sarcasm
Subtitle: Joaquin Phoenix pledged to not change for each awards event
Article: **A year ago, Joaquin Phoenix made headlines when he appeared on the red carpet at the Golden Globes wearing a tuxedo with a paper bag over his head that read, "I am a shape-shifter. I can't change the world. I can only change myself." It was a promise to not change to fit into the Hollywood mold: "I think that's a really special thing, to not change yourself. I think it's a really special thing to say, 'This is what's inside of me, I'm proud of it, and I'm not going to be ashamed because of the way that someone else thinks I should be.'" Now, it's the Oscars, and Phoenix is at it again. But this time, his publicist is saying he'll be wearing a tux no matter what.**
Megyn Kelly was not impressed, and she let him have it on The Tonight Show. "You know, I feel like, I feel like you could have worn the tux," she says. "But you're saying you're a shape-shifter. I don't know if you can change your tux, but you can change your mind. You can change your mind. You can change your mind." Phoenix says he did, but it didn't stick. "I was like, 'Okay, I'm going to wear a tuxedo to this thing.' And then I thought, 'I don't want to wear a tuxedo to this thing.'" Kelly goes on to encourage him to change his mind again, but Phoenix says it's too late: "I'm committed to wearing this."

Nature language



Weather

Discrete Time Stochastic Process



x_1 x_2 x_3 x_4 x_5

$t = 1$ $t = 2$ $t = 3$ $t = 4$ $t = 5$

- After sampling in time, we can model sequential data with a discrete time stochastic process.
- We call the random variable $\boldsymbol{x}_t$ the state of the process at time t.
- Stationarity assumption: the data evolves in time, but the distribution from which it is generated remains the same.

Markov Model

- Markov assumption: The probability of the next state depends only on the current state:

$$P(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \dots, \boldsymbol{x}_1) = P(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)$$

- The Markov assumption implies that:

$$P(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T) = P(\boldsymbol{x}_1) \prod_t P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}, \dots, \boldsymbol{x}_1)$$

Any process: $\mathcal{O}(K^T)$

$$\Big\Downarrow \text{ Markov assumption}$$

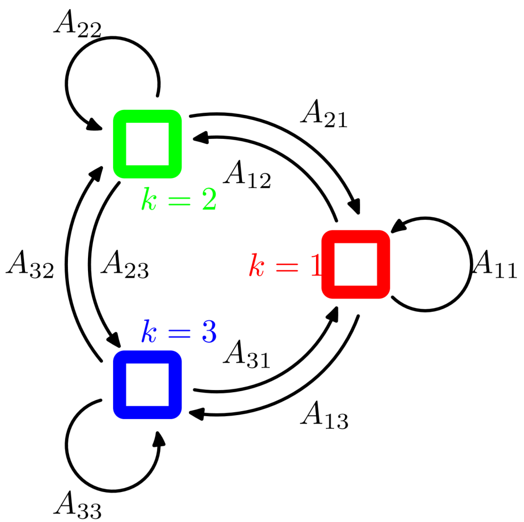
$$= P(\boldsymbol{x}_1) \prod_t P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$$

Markov process: $\mathcal{O}(K^2)$

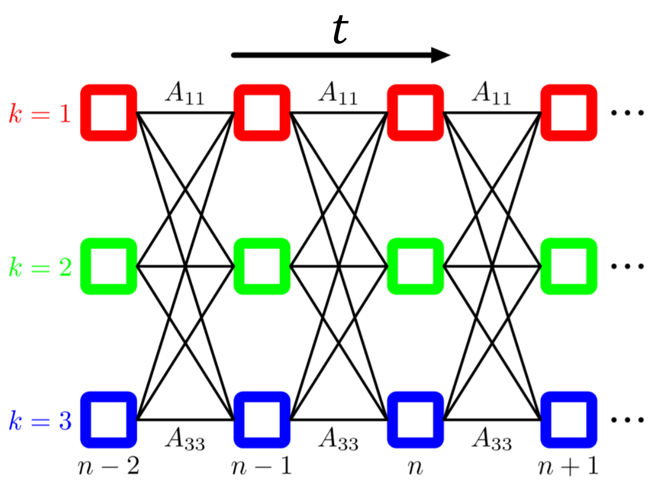
State Transition Matrix

- Markov chain is defined via a state transition matrix:
  - Assume the state is discrete with K possible outcomes:

$$A_{ij} = P(x_{t+1,j} = 1 | x_{t,i} = 1) \quad i, j \in \{1, 2, \cdots, K\}$$



State transition diagram:
nodes → states
edges → state transition matrix



If we unfold the state transition diagram over time, we obtain a lattice.

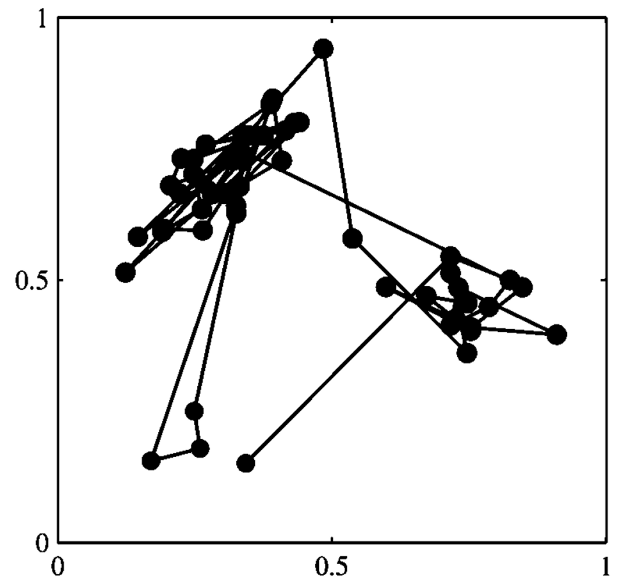
Outline

- Hidden Markov Model
  - Markov Model
  - **Hidden Markov Model**
  - Inference for Hidden Markov Model
    - Viterbi Algorithm
    - Forward-Backward Algorithm
- Linear Dynamical System
  - Kalman Filter
  - Particle Filter

Observation without Markov Property

The data tend to gather in a group

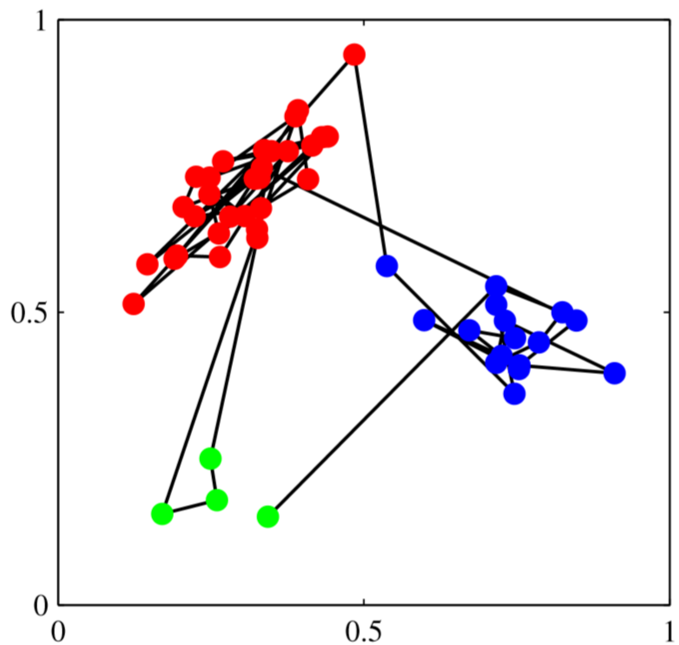
No Markov property



X	0.35	0.75	0.81	0.76	0.87	0.8	0.55	0.75	0.51	0.5	0.25	0.14	0.23	0.36
Y	0.1	0.5	0.49	0.47	0.4	0.4	0.5	0.3	0.51	0.95	0.75	0.55	0.66	0.71

- How to extend the expressiveness of Markov models in this case?

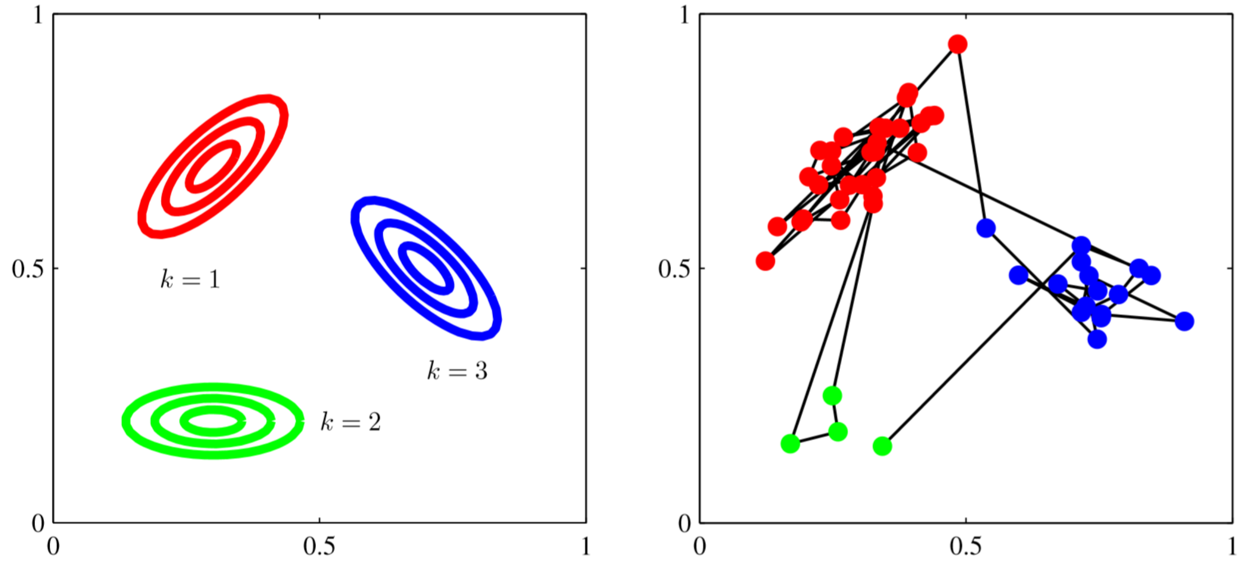
Observation without Markov Property



X	0.35	0.75	0.81	0.76	0.87	0.8	0.55	0.75	0.51	0.5	0.25	0.14	0.23	0.36
Y	0.1	0.5	0.49	0.47	0.4	0.4	0.5	0.3	0.51	0.95	0.75	0.55	0.66	0.71

- If we give them a label, we can find that the hidden state is tractable.
- We can use a Markov model to describe the hidden state transitions.

Hidden Markov Model (HMM)



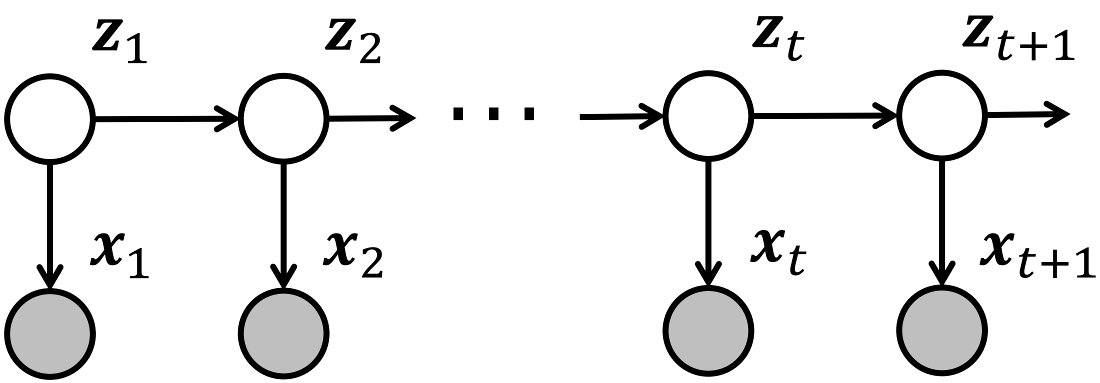
- **Hidden Markov Model**: the hidden state is a Markov model, and the observation is generated depending on the choice of hidden label.
- The hidden state variable of HMM is similar to the latent variable in Gaussian Mixture Model, but with order and Markov dependency.

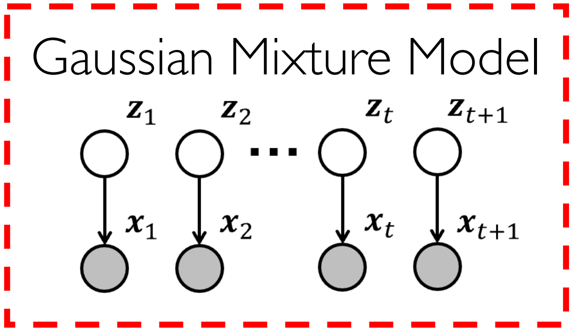
Hidden Markov Model (HMM)

- HMM is defined by a sequence of **hidden states** and **observations**:
  - The hidden state $\boldsymbol{z}_t$ is a Markov chain.

    Transition distribution: $P(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:t}) = P(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$
  - The observation $\boldsymbol{x}_t$ is dependent only on the current hidden state.

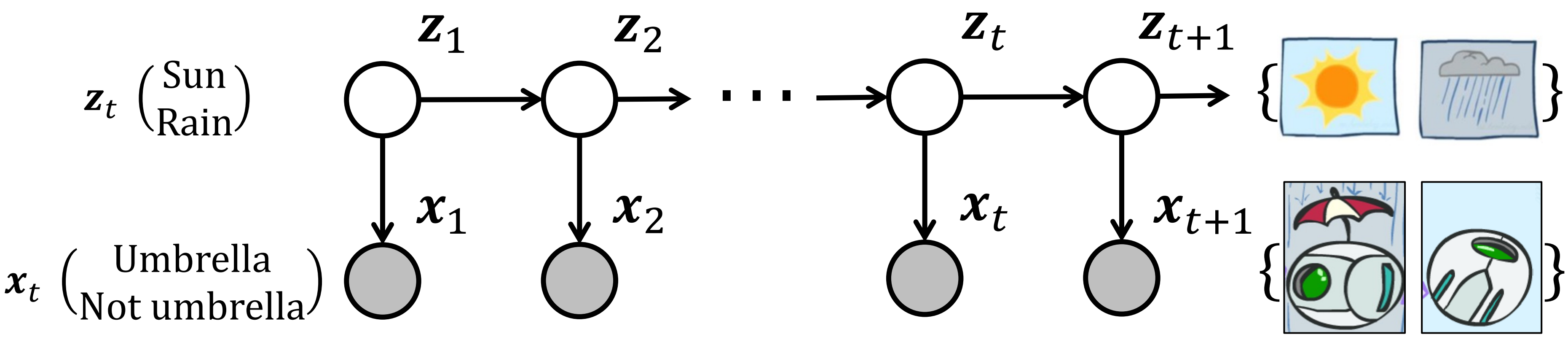
    Emission distribution: $P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{1:t}) = P(\boldsymbol{x}_t|\boldsymbol{z}_t)$





- The observed variables, however, do not satisfy the Markov property.

Example: Weather Forecasting



Transitions: $P(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$

Z_{t-1} \ Z_t	Rain	Sun
Rain	0.7	0.3
Sun	0.1	0.9

Emissions: $P(\boldsymbol{x}_t|\boldsymbol{z}_t)$

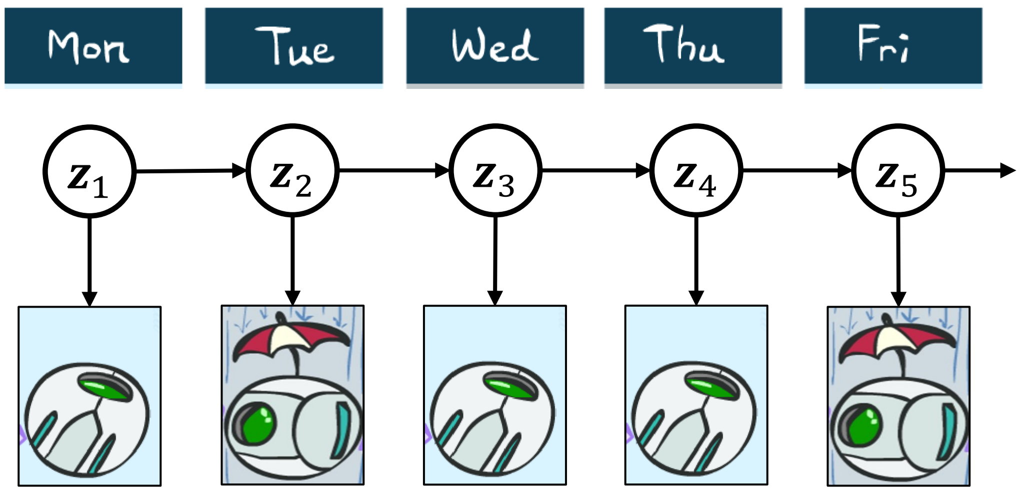
Z_t \ X_t	Umbrella	Not umbrella
Rain	0.9	0.1
Sun	0.2	0.8

Outline

- Hidden Markov Model
  - Markov Model
  - Hidden Markov Model
  - **Inference for Hidden Markov Model**
    - Viterbi Algorithm
    - Forward-Backward Algorithm
- Linear Dynamical System
  - Kalman Filter
  - Particle Filter

Inference of HMM

- **Question:** Given the observation sequence $\widehat{\boldsymbol{X}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$, how to estimate the hidden states $\widehat{\boldsymbol{Z}} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T\}$?



State: Rain or Sun ?

⇧

Observation: Umbrella

- How about Learning of HMM? Please work it out by yourself!!

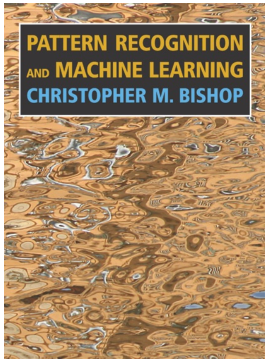
EM Algorithm

The inference problem is the evaluation of the E step.
We have Markov property

- Choose initial θ^{old}.
- Expectation Step
  - Let $q^*(z) = p(z|x, \theta^{\text{old}})$. [$q^*$ gives best lower bound at θ^{old}]
  - Let $J(\theta) \coloneqq \mathcal{L}(q^*, \theta) = \underbrace{\sum_z q^*(z) \log\left(\frac{p(x,z|\theta)}{q^*(z)}\right)}_{\text{Expectation w.r.t. } z \sim q^*(z)}$
- Maximization Step

$$\theta^{\text{new}} = \underset{\theta}{\operatorname{argmax}}\, J(\theta)$$

  - The M step of HMM is similar to GMM (omitted here)
- Go to the Expectation Step, until converged.



PRML
Chapter 13

Two Methods

- Minimize the sequence error rate:

$$L(\boldsymbol{Z}, \widehat{\boldsymbol{Z}}) = \mathbf{1}(\widehat{\boldsymbol{Z}} \neq \boldsymbol{Z})$$

$$\widehat{\boldsymbol{Z}} = \arg\max_{\boldsymbol{Z}} p(\boldsymbol{Z}|\boldsymbol{X}) \propto \arg\max_{\boldsymbol{Z}} p(\boldsymbol{Z}, \boldsymbol{X})$$

Viterbi algorithm

- Minimize the state error rate:

$$L(\boldsymbol{Z}, \widehat{\boldsymbol{Z}}) = \sum_t \mathbf{1}(\boldsymbol{z}_t \neq \hat{\boldsymbol{z}}_t)$$

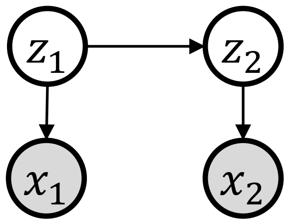
$$\hat{\boldsymbol{z}}_t = \arg\max_{\boldsymbol{z}_t} p(\boldsymbol{z}_t|\boldsymbol{X})$$

Forward-Backward algorithm

Example: Binary HMM

- Suppose after observation, the posterior $P(\boldsymbol{Z}|\boldsymbol{X})$ is:

$z_t \in \{1,2\}$



$\boldsymbol{z}_2$ \ $\boldsymbol{z}_1$	1	2
1	0.04	0.3
2	0.36	0.3

- Minimize sequence error rate:

$$E\left(L\left(\boldsymbol{Z},\widehat{\boldsymbol{Z}}\right)|\boldsymbol{X}\right) = 1 - P\left(\boldsymbol{Z} = \widehat{\boldsymbol{Z}}\middle|\boldsymbol{X}\right) \Rightarrow \widehat{\boldsymbol{Z}} = \{1,2\}$$

- Minimize state error rate:

$$E\left(L\left(\boldsymbol{Z},\widehat{\boldsymbol{Z}}\right)|\boldsymbol{X}\right) = 1 - P(\boldsymbol{z}_1 = \hat{\boldsymbol{z}}_1|\boldsymbol{X}) + 1 - P(\boldsymbol{z}_2 = \hat{\boldsymbol{z}}_2|\boldsymbol{X})$$

$$\Rightarrow \widehat{\boldsymbol{Z}} = \{2,2\}$$

Outline

- Hidden Markov Model
  - Markov Model
  - Hidden Markov Model
  - Inference for Hidden Markov Model
    - **Viterbi Algorithm**
    - Forward-Backward Algorithm
- Linear Dynamical System
  - Kalman Filter
  - Particle Filter

Viterbi Algorithm

Andrew Viterbi
Shannon Award (1991)

- In HMMs, Viterbi algorithm is used for minimizing sequence error rate.

$$\hat{\boldsymbol{Z}} = \arg\max_{\boldsymbol{Z}} P(\boldsymbol{Z}|\boldsymbol{X}) \propto \arg\max_{\boldsymbol{Z}} P(\boldsymbol{Z},\boldsymbol{X})$$

$$= \arg\max_{\boldsymbol{Z}} \left[P(\boldsymbol{z}_1)\prod_t P(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})\right]\cdot\left[\prod_t P(\boldsymbol{x}_t|\boldsymbol{z}_t)\right]$$

Transition

Emission

$$= \arg\min_{\boldsymbol{Z}} \left[\phi_1(\boldsymbol{z}_1) + \sum_{t=2}^{T}\left(\phi_t(\boldsymbol{z}_t) + \psi_t(\boldsymbol{z}_t,\boldsymbol{z}_{t-1})\right)\right]$$

minimizing negative log-probability.

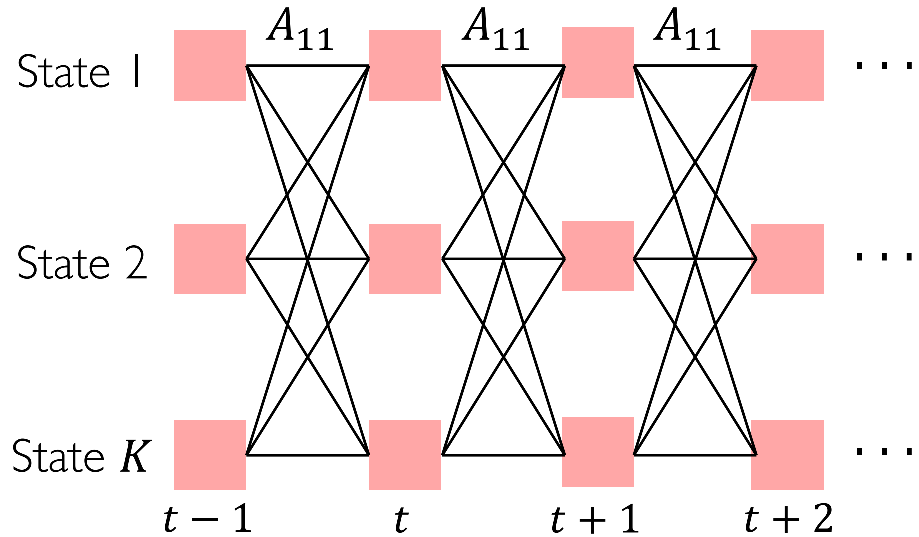
$$\phi_t(\boldsymbol{z}_t) = \begin{cases} -\log P(\boldsymbol{x}_1|\boldsymbol{z}_1) - \log P(\boldsymbol{z}_1) & t = 1 \\ -\log P(\boldsymbol{x}_t|\boldsymbol{z}_t) & t \geq 2 \end{cases}$$

$$\psi_t(\boldsymbol{z}_t,\boldsymbol{z}_{t-1}) = -\log P(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$$

Lattice Diagram

$$\hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}} \left[\phi_1(\boldsymbol{z}_1) + \sum_{t=2}^{T} \left(\phi_t(\boldsymbol{z}_t) + \psi_t(\boldsymbol{z}_t, \boldsymbol{z}_{t-1}) \right) \right]$$

- Node weight: $\phi_t(\boldsymbol{z}_t)$, edge weight: $\psi_t(\boldsymbol{z}_t, \boldsymbol{z}_{t-1})$
- Row for each possible state, column for each time step.
- State transition diagram determines allowable paths (K^T in total).
- MAP estimate is the **shortest path** through weighted edges in the graph.



Viterbi Algorithm

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}} \left[\phi_1(\mathbf{z}_1) + \sum_{t=2}^{T} \left(\phi_t(\mathbf{z}_t) + \psi_t(\mathbf{z}_t, \mathbf{z}_{t-1}) \right) \right]$$

- Define cost of shortest path ending with a specified state $\mathbf{z}_t$:

$$f_t(\mathbf{z}_t) = \min_{\mathbf{z}} \left[\phi_1(\mathbf{z}_1) + \sum_{t'=2}^{t} \left(\phi_{t'}(\mathbf{z}_{t'}) + \psi_{t'}(\mathbf{z}_{t'}, \mathbf{z}_{t'-1}) \right) \right]$$

- We can recursively write this minimization as follows:

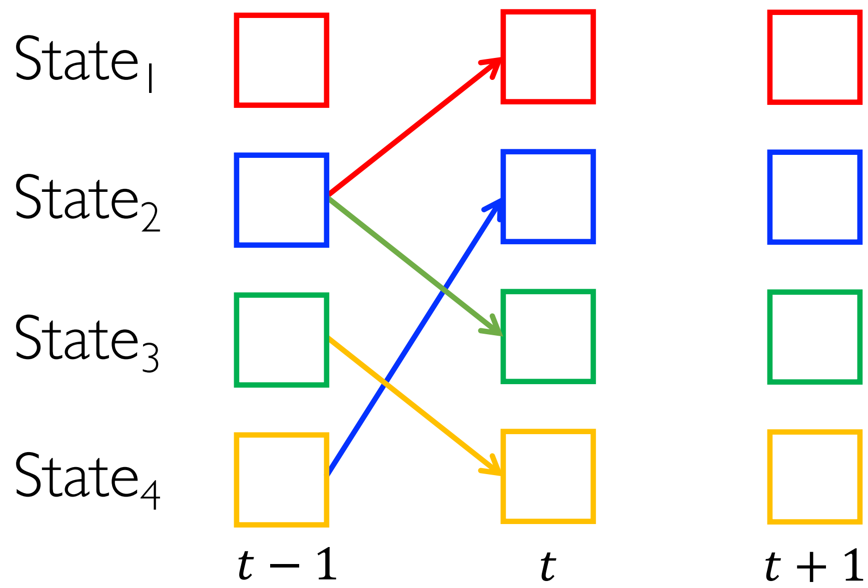
$$f_t(\mathbf{z}_t) = \phi_t(\mathbf{z}_t) + \min_{\mathbf{z}_{t-1}} [\psi_t(\mathbf{z}_t, \mathbf{z}_{t-1}) + f_{t-1}(\mathbf{z}_{t-1})]$$

- **Interpretation:** every path up to time t has the following form:
  1. A path to some state at time $t-1$.
  2. A cost for the transition from time $t-1$ to time t.

Dynamical programming

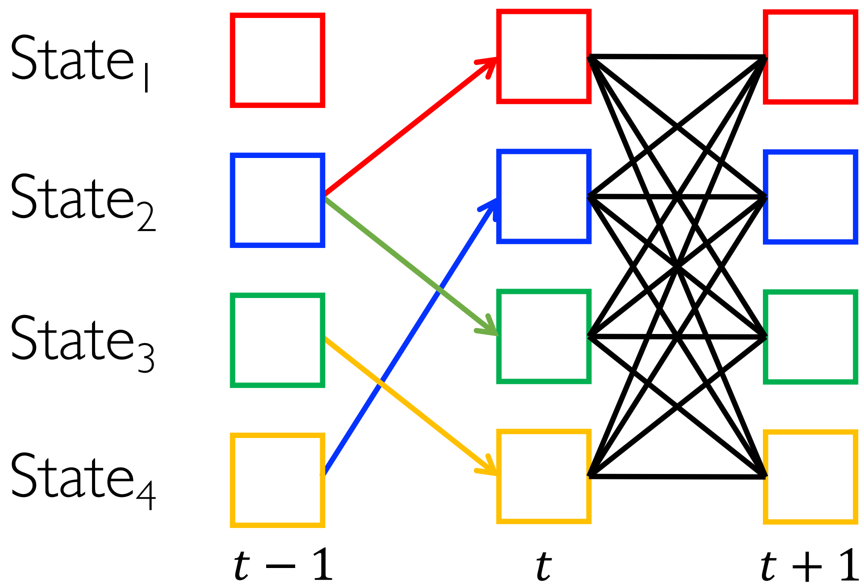
Viterbi Algorithm

- Initiation: $f_1(\boldsymbol{z}_1) = \phi_1(\boldsymbol{z}_1)$
- At time step $t-1$, keep the shortest path and corresponding cost $f_{t-1}(\boldsymbol{z}_{t-1})$ ending with each state $\boldsymbol{z}_{t-1}$:



Viterbi Algorithm

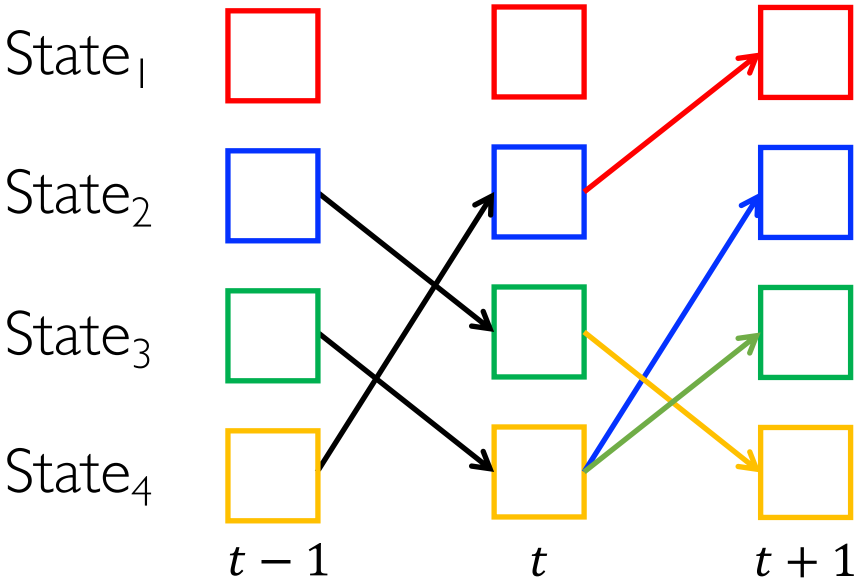
- From time t to $t+1$:
  - Evaluate the weights $\psi_t(\boldsymbol{z}_t, \boldsymbol{z}_{t-1})$ of every edge from t to $t+1$.
  - K paths arriving at each node k, $\mathcal{O}(K^2)$ computation cost in total.



Viterbi Algorithm

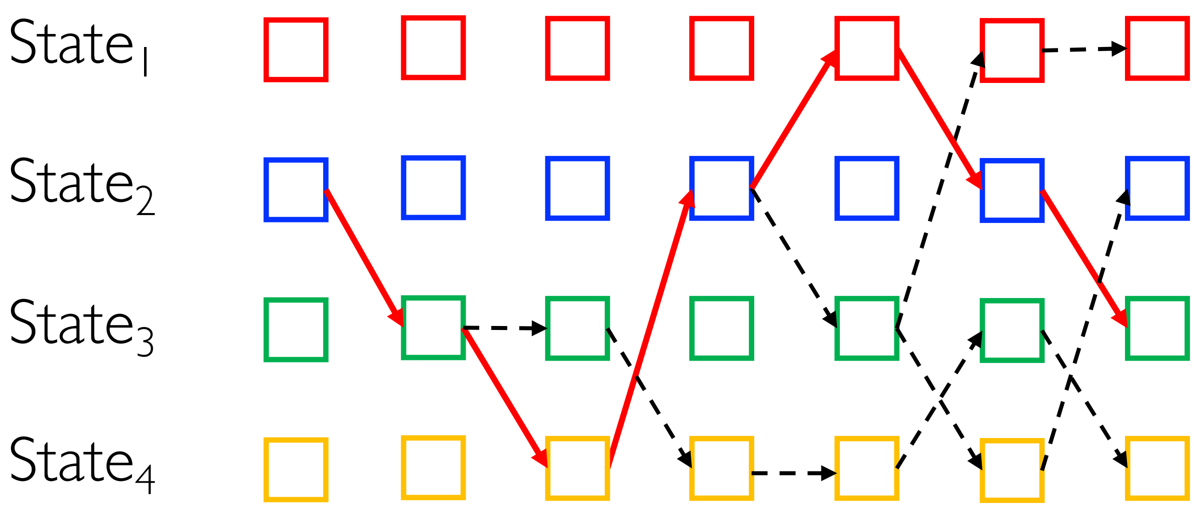
- From time t to $t+1$:
  - Update the best path ending with each state $\boldsymbol{z}_t$.

$$f_t(\boldsymbol{z}_t) = \phi_t(\boldsymbol{z}_t) + \min_{\boldsymbol{z}_{t-1}}[\psi_t(\boldsymbol{z}_t, \boldsymbol{z}_{t-1}) + f_{t-1}(\boldsymbol{z}_{t-1})]$$



Viterbi Algorithm

- Algorithm stop:
  - When reaching the end T, we have K paths through this lattice.
  - Select the best one as the final choice.
- Overall computation cost: $\mathcal{O}(TK^2)$



State estimation:	2	3	4	2	1	2	3

Outline

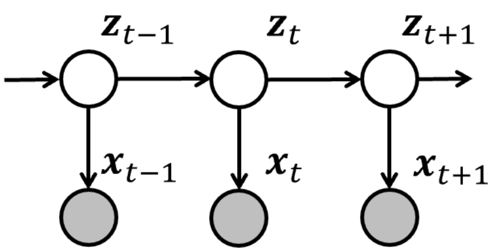
- Hidden Markov Model
  - Markov Model
  - Hidden Markov Model
  - Inference for Hidden Markov Model
    - Viterbi Algorithm
    - **Forward-Backward Algorithm**
- Linear Dynamical System
  - Kalman Filter
  - Particle Filter

Optimization of State Error Rate

- Viterbi algorithm is used to minimize the sequence error rate. Now, we focus on state error rate:

$$
\begin{aligned}
P(\boldsymbol{z}_t|\boldsymbol{x}_{1:T}) &= P(\boldsymbol{x}_{1:T}, \boldsymbol{z}_t)/P(\boldsymbol{x}_{1:T}) \\
&= P(\boldsymbol{x}_{1:T}|\boldsymbol{z}_t)P(\boldsymbol{z}_t)/P(\boldsymbol{x}_{1:T}) \\
&= P(\boldsymbol{x}_{1:t}|\boldsymbol{z}_t)P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_t)P(\boldsymbol{z}_t)/P(\boldsymbol{x}_{1:T}) \\
&= P(\boldsymbol{x}_{1:t}, \boldsymbol{z}_t)P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_t)/P(\boldsymbol{x}_{1:T}) \\
&= P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})P(\boldsymbol{x}_{1:t})P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_t)/P(\boldsymbol{x}_{1:T}) \\
&\propto P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_t)
\end{aligned}
$$

Conditional independency of HMM



$\equiv \alpha_t\ (\boldsymbol{z}_t)$, we evaluate it with forward algorithm

$\equiv \beta_t(\boldsymbol{z}_t)$, we evaluate it with backward algorithm

Forward Algorithm

- Physics meaning: estimate each hidden state online.

Define $$\alpha_t(\mathbf{z}_t) = P(\mathbf{z}_t|\mathbf{x}_{1:t})$$

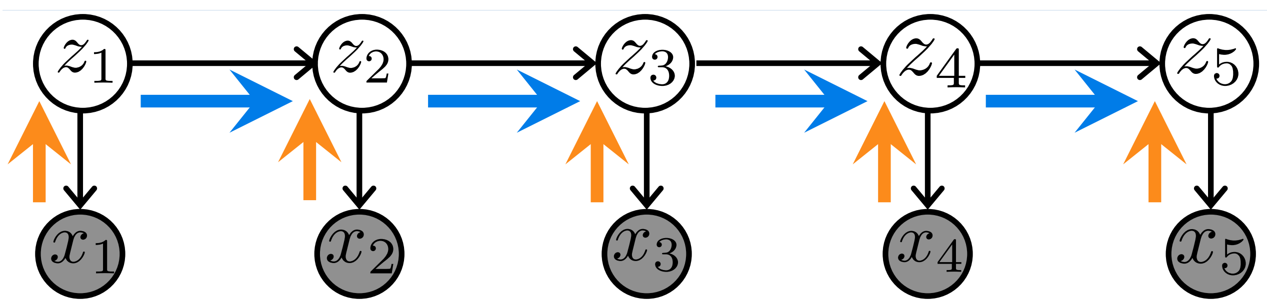
Prior distribution $$P(\mathbf{z}_t|\mathbf{x}_{1:t-1}) = \sum_{\mathbf{z}_{t-1}} P(\mathbf{z}_t|\mathbf{z}_{t-1})\alpha_{t-1}(\mathbf{z}_{t-1})$$

Posterior distribution $$\alpha_t(\mathbf{z}_t) = c_t P(\mathbf{x}_t|\mathbf{z}_t) P(\mathbf{z}_t|\mathbf{x}_{1:t-1})$$

Normalizer

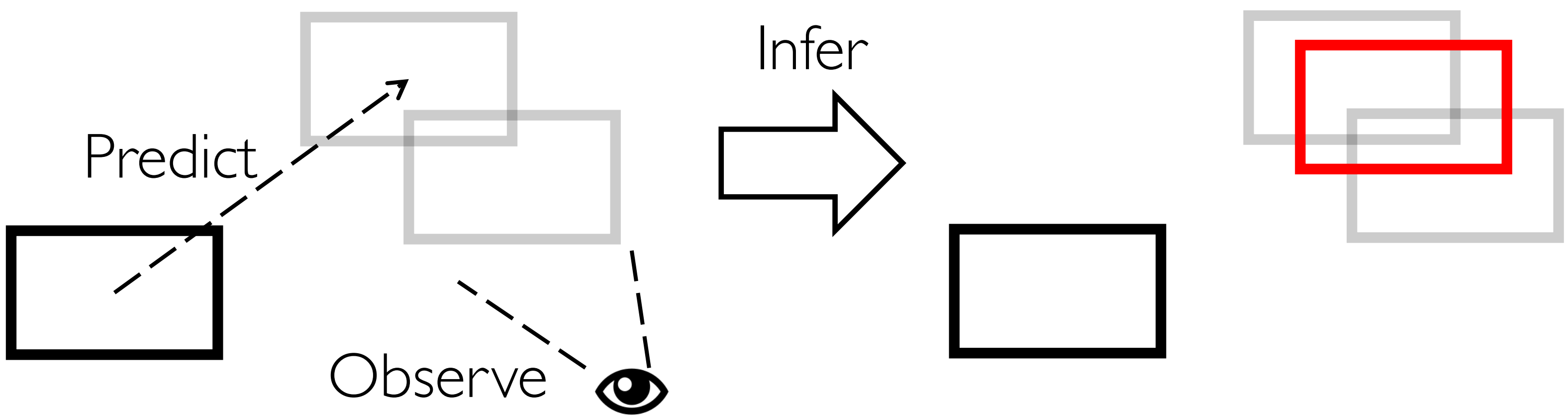
- Recursively proceed forward through the HMM network:

$$P(\mathbf{z}_1|\mathbf{x}_1) \longrightarrow P(\mathbf{z}_2|\mathbf{x}_1) \longrightarrow P(\mathbf{z}_2|\mathbf{x}_{1:2}) \longrightarrow P(\mathbf{z}_3|\mathbf{x}_{1:2})$$



Forward Algorithm

- We have two inaccurate methods to estimate the hidden state of HMM:
  - From the new observation, using emission equation.
  - From current estimation, using transition equation.
- The forward algorithm "combines" the results of two methods!



Forward Algorithm: Prior Distribution

- **Prior Distribution:** Predict the next state before observation

$$P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}) = \sum_{z_{t-1}} P(\boldsymbol{z}_t, \boldsymbol{z}_{t-1}|\boldsymbol{x}_{1:t-1}) \qquad \text{(Total probability)}$$

$$= \sum_{z_{t-1}} P(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_{1:t-1}) P(\boldsymbol{z}_{t-1}|\boldsymbol{x}_{1:t-1})$$

$$= \sum_{z_{t-1}} P(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) \alpha_{t-1}(\boldsymbol{z}_{t-1})$$

- **Basic idea:** The beliefs get pushed through the **transition matrix.**

Forward Algorithm: Posterior Distribution

- Posterior Distribution: After an observation, the belief is updated

$$\alpha_t(\boldsymbol{z}_t) = P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t}) = \frac{P(\boldsymbol{z}_t, \boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})}{P(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})}$$

$$\propto P(\boldsymbol{z}_t, \boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}) \qquad \text{(Normalize of } \boldsymbol{x}_t\text{)}$$

$$= P(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_t)P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1})$$

$$= P(\boldsymbol{x}_t|\boldsymbol{z}_t)P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}) \qquad \text{(Markov dependency)}$$

- **Basic idea:** The beliefs reweighted by **emission matrix.**
- Due to the ∝, we need to renormalize after the above calculation.

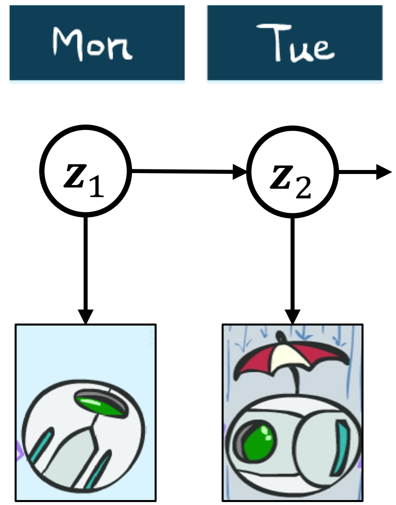
Example: Weather Forecasting

Initial distribution:

$$P(\mathbf{z}_1|\mathbf{x}_1) = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

Observation:

{not umbrella, umbrella}



Question:

What is the weather on Tuesday?

Transition Matrix

$\mathbf{z}_{t-1}$ \ $\mathbf{z}_t$	Rain	Sun
Rain	0.7	0.3
Sun	0.1	0.9

Emission Matrix

$\mathbf{z}_t$ \ $\mathbf{x}_t$	Umbrella	Not umbrella
Rain	0.9	0.1
Sun	0.2	0.8

Example: Weather Forecasting

z_{t-1} \ z_t	Rain	Sun
Rain	0.7	0.3
Sun	0.1	0.9

z_t \ x_t	Umbrella	Not umbrella
Rain	0.9	0.1
Sun	0.2	0.8

$$P(\boldsymbol{z}_2|\boldsymbol{x}_1) = P(\boldsymbol{z}_2|z_{1,\text{rain}})\alpha_1(z_{1,\text{rain}}) + P(\boldsymbol{z}_2|z_{1,\text{sun}})\alpha_1(z_{1,\text{sun}})$$

$$= \begin{pmatrix}0.3\\0.7\end{pmatrix}\times 0.5 + \begin{pmatrix}0.9\\0.1\end{pmatrix}\times 0.5 = \begin{pmatrix}0.6\\0.4\end{pmatrix}$$

$$\alpha_2(\boldsymbol{z}_2) \propto P(\boldsymbol{x}_2|\boldsymbol{z}_2)P(\boldsymbol{z}_2|\boldsymbol{x}_1) = \begin{pmatrix}0.2\\0.9\end{pmatrix}\times\begin{pmatrix}0.6\\0.4\end{pmatrix} = \begin{pmatrix}0.12\\0.36\end{pmatrix}$$

P(sun)=0.25
P(rain)=0.75

Backward Algorithm

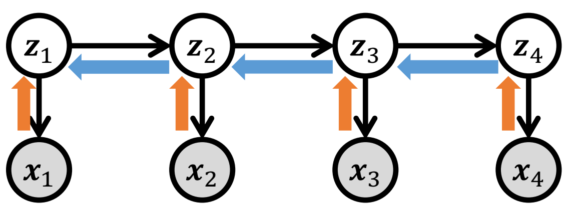
- Similar to forward algorithm, we can deduce **backward algorithm**:

$$\beta_t(\boldsymbol{z}_t) = P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_t) = \sum_{\boldsymbol{z}_{t+1}} P(\boldsymbol{x}_{t+1:T}, \boldsymbol{z}_{t+1}|\boldsymbol{z}_t)$$

$$= \sum_{\boldsymbol{z}_{t+1}} P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_{t+1}, \boldsymbol{z}_t)P(\boldsymbol{z}_{t+1}|\boldsymbol{z}_t)$$

$$= \sum_{\boldsymbol{z}_{t+1}} P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_{t+1})P(\boldsymbol{z}_{t+1}|\boldsymbol{z}_t)$$

$$= \sum_{\boldsymbol{z}_{t+1}} \underbrace{P(\boldsymbol{x}_{t+2:T}|\boldsymbol{z}_{t+1})}_{\beta_{t+1}(z_{t+1})} \underbrace{P(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{t+1})}_{\text{Emission}} \underbrace{P(\boldsymbol{z}_{t+1}|\boldsymbol{z}_t)}_{\text{Transition}}$$



Recursively run this algorithm backward

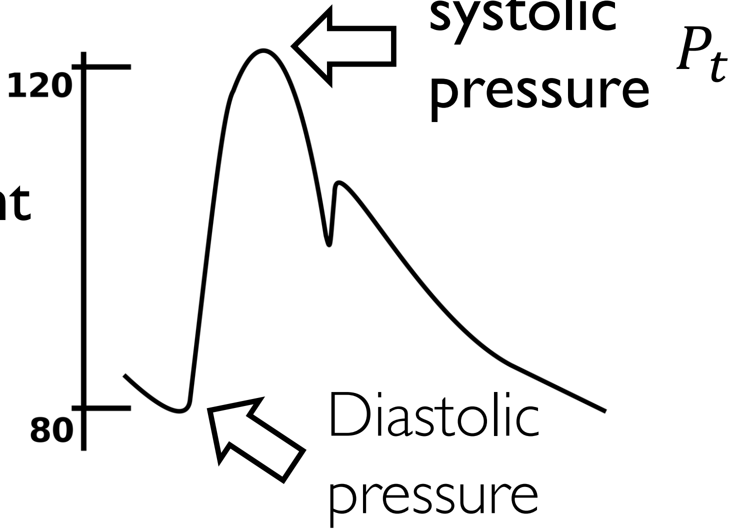
Outline

- Hidden Markov Model
  - Markov Model
  - Hidden Markov Model
  - Inference for Hidden Markov Model
    - Viterbi Algorithm
    - Forward-Backward Algorithm
- **Linear Dynamical System**
  - Kalman Filter
  - Particle Filter

Track of Blood Pressure

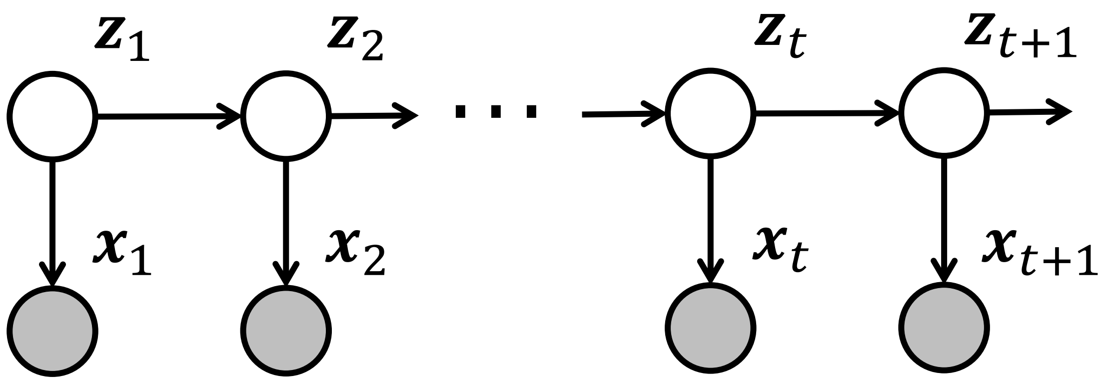
- Suppose we are tracking the systolic pressure P_t of a person, the measurement sequence is denoted as M_t:
  - P_t is not stable, and the measurement is not accurate.
- HMM no more applies to this case, since the value of hidden state P_t is continuous.

Measurement M_t



Linear Dynamical System (LDS)

- We can use Linear Dynamical System (LDS) to model this case.
- The structure of LDS is almost the same as HMM.
  - But the hidden state can be continuous in LDS.



Transition $\quad P(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:t}) = P(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}), \qquad \boldsymbol{z}_i \in \mathbb{R}$

Emission $\quad P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{1:t}) = P(\boldsymbol{x}_t|\boldsymbol{z}_t), \qquad \boldsymbol{x}_i \in \mathbb{R}$

Inference of LDS

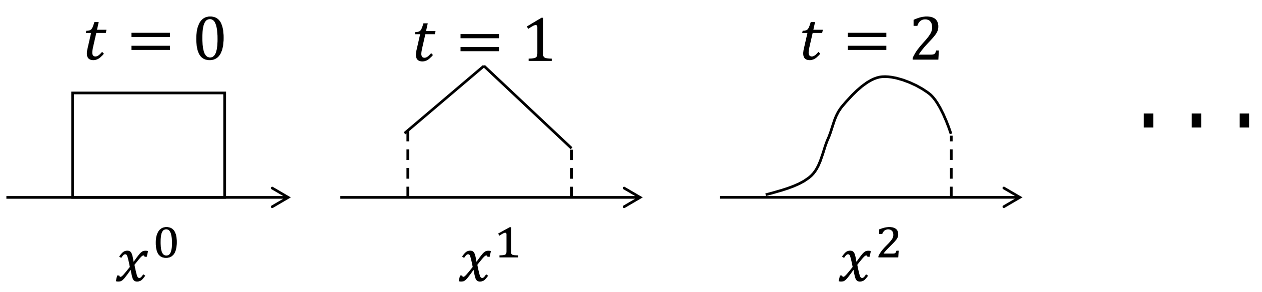
- Recall the forward algorithm of HMM:

$$P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t}) = c_t P(\boldsymbol{x}_t|\boldsymbol{z}_t) \sum_{z_{t-1}} \mathrm{P}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) P(\boldsymbol{z}_{t-1}|\boldsymbol{x}_{1:t-1})$$

- For LDS, we change the discrete sum in HMM to continuous integral:

$$P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t}) = c_t P(\boldsymbol{x}_t|\boldsymbol{z}_t) \int \mathrm{P}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) P(\boldsymbol{z}_{t-1}|\boldsymbol{x}_{1:t-1})\, \mathrm{d}\boldsymbol{z}_{t-1}$$

- However, if we recursively calculate the distribution, the expression will become extremely complex.



Kalman Filter vs. Particle Filter

- This problem can be addressed from two directions:
  - We can choose certain form of the distribution of hidden state, so that this form will not change at each stage.

    This direction leads to **Kalman Filter**.

  - We can use Monte Calo method to estimate the complex distribution, instead of trying to calculating it.

    This direction leads to **Particle Filter**.

  - Both were developed in different fields but unified by the PGM.

Outline

- Hidden Markov Model
  - Markov Model
  - Hidden Markov Model
  - Inference for Hidden Markov Model
    - Viterbi Algorithm
    - Forward-Backward Algorithm
- Linear Dynamical System
  - **Kalman Filter**
  - Particle Filter

Gaussian Distribution

- In order to get a simple expression, we aim at finding a certain form of distribution which will not change at each time step.

Parameter change $F(\theta_0)$ $F(\theta_1)$ $\bullet\bullet\bullet$ $F(\theta_t)$ ✓

Form change $F(\theta_0)$ $F^2(\theta_1)$ $\bullet\bullet\bullet$ $F^t(\theta_t)$ ✗

- Fortunately, **Gaussian distribution** meets this demand. We can write the transition and emission distributions in the general form:

$$P(\boldsymbol{z}_{n+1}|\boldsymbol{z}_n) = \mathcal{N}(\boldsymbol{z}_n|\boldsymbol{A}\boldsymbol{z}_n, \boldsymbol{\Gamma})$$

$$P(\boldsymbol{x}_n|\boldsymbol{z}_n) = \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{C}\boldsymbol{z}_n, \boldsymbol{\Sigma})$$

$$P(\boldsymbol{z}_1) = \mathcal{N}(\boldsymbol{z}_1|\boldsymbol{\mu}_0, \boldsymbol{V}_0)$$

Gaussian Distribution

- Traditionally, these distributions are more commonly expressed in an equivalent form in terms of noisy linear equations:

Transition	$\boldsymbol{z}_{t+1} = \boldsymbol{A}\boldsymbol{z}_t + \boldsymbol{w}_n,$	$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{w} \mid 0, \boldsymbol{\Gamma})$
Emission	$\boldsymbol{x}_t = \boldsymbol{C}\boldsymbol{z}_t + \boldsymbol{v}_t,$	$\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{v} \mid 0, \boldsymbol{\Sigma})$
Initialization	$\boldsymbol{z}_1 = \boldsymbol{\mu}_0 + \boldsymbol{u},$	$\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{u} \mid 0, \boldsymbol{V}_0)$

- Now we try to track the **expectation $\boldsymbol{\mu}_t$** and **variance $\boldsymbol{V}_t$** of state $\boldsymbol{z}_t$:

$$\boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{z}_t \mid \boldsymbol{\mu}_t, \boldsymbol{V}_t)$$

Kalman Filter

- Given the expectation $\boldsymbol{\mu}_{t-1}$ variance $\boldsymbol{V}_{t-1}$, with new observation $\boldsymbol{x}_t$, we can evaluate the posterior distribution of $\boldsymbol{z}_t$ with Kalman Filter:

Variance

$\boldsymbol{V}_{t-1}$

Expectation

$\boldsymbol{\mu}_{t-1}$

Prior variance

$$\boldsymbol{P}_{t-1} = \boldsymbol{\Gamma} + \boldsymbol{A}\boldsymbol{V}_{t-1}\boldsymbol{A}^T$$

Kalman gain matrix

$$\boldsymbol{K}_t = \boldsymbol{P}_{t-1}\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}_{t-1}\boldsymbol{C}^T + \boldsymbol{\Sigma})^{-1}$$

Update variance

$$\boldsymbol{V}_t = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{C})\boldsymbol{P}_{t-1}$$

Update expectation

$$\boldsymbol{\mu}_t = \boldsymbol{A}\boldsymbol{\mu}_{t-1} + \boldsymbol{K}_t(\boldsymbol{x}_t - \boldsymbol{C}\boldsymbol{A}\boldsymbol{\mu}_{t-1})$$

Interpretation of Expectation Update

$$\boldsymbol{\mu}_t = \boldsymbol{A}\boldsymbol{\mu}_{t-1} + \boldsymbol{K}_t(\boldsymbol{x}_t - \boldsymbol{C}\boldsymbol{A}\boldsymbol{\mu}_{t-1})$$

- We first predict the expectation $\boldsymbol{\mu}$ using the transition matrix $\boldsymbol{A}$:

$$\widehat{\boldsymbol{\mu}}_t = \boldsymbol{A}\boldsymbol{\mu}_{t-1}$$

- Then, we compare the observation of $\widehat{\boldsymbol{\mu}}_t$ and real observation:

$$\boldsymbol{x}_t - \boldsymbol{C}\widehat{\boldsymbol{\mu}}_t$$

- Finally, we calculate a correction proportional to observation error:

$$\boldsymbol{\mu}_t = \widehat{\boldsymbol{\mu}}_t + \boldsymbol{K}_t(\boldsymbol{x}_t - \boldsymbol{C}\widehat{\boldsymbol{\mu}}_t)$$

- The coefficient $\boldsymbol{K}_t$ is proportional to the prior variance $\boldsymbol{P}_{t-1}$: if the prior prediction is accurate (variance is small), the $\boldsymbol{K}_t$ will be small ("trust" the prediction), and vice versa.

Outline

- Hidden Markov Model
  - Markov Model
  - Hidden Markov Model
  - Inference for Hidden Markov Model
    - Viterbi Algorithm
    - Forward-Backward Algorithm
- Linear Dynamical System
  - Kalman Filter
  - **Particle Filter**

Motivation

- With the assumption of Gaussian distribution, we educe the method of Kalman Filter.
  - In reality, Gaussian distribution is not available in many cases.
- For example, for Hidden Markov Models, if the number of hidden state is large, we can use neither forward algorithm nor Kalman Filter.

HMM with large hidden state number

Inference of HMM: Forward algorithm

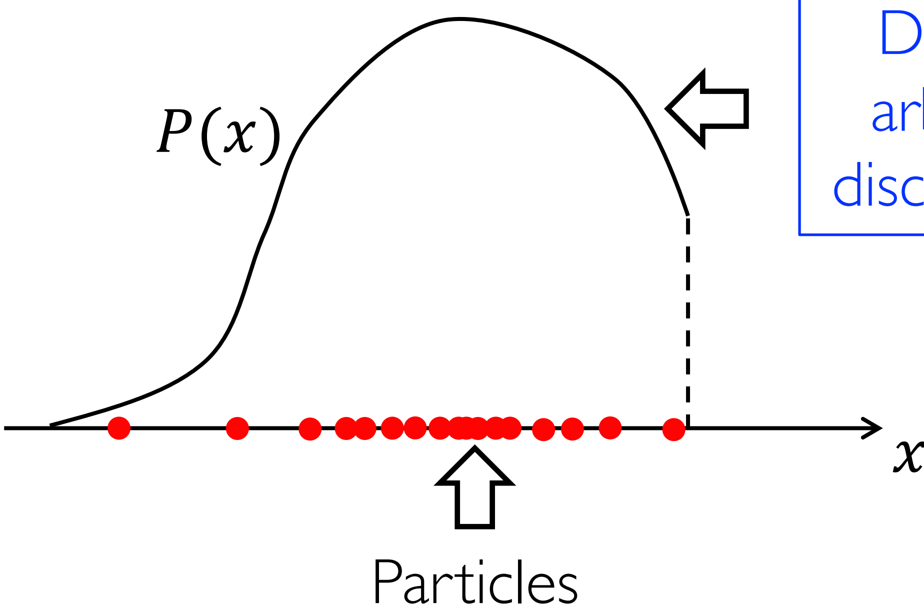
Large complexity

Inference of LDS: Kalman Filter

Discrete hidden state, does not apply

Monte Calo Method

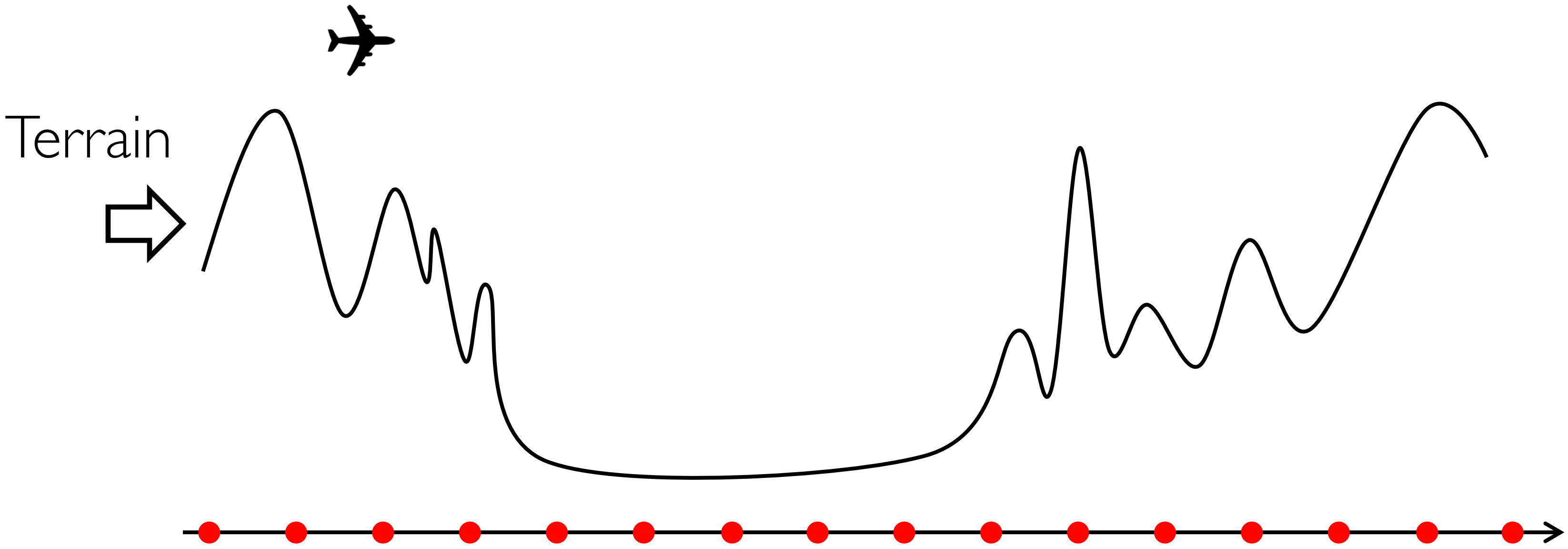
- Particle Filter uses Monte Calo method to solve the problem.
- We use the samples from a distribution to express it, instead of analytical expression:
  - The samples are called particles.



Distribution of x is arbitrary, no matter discrete or continuous

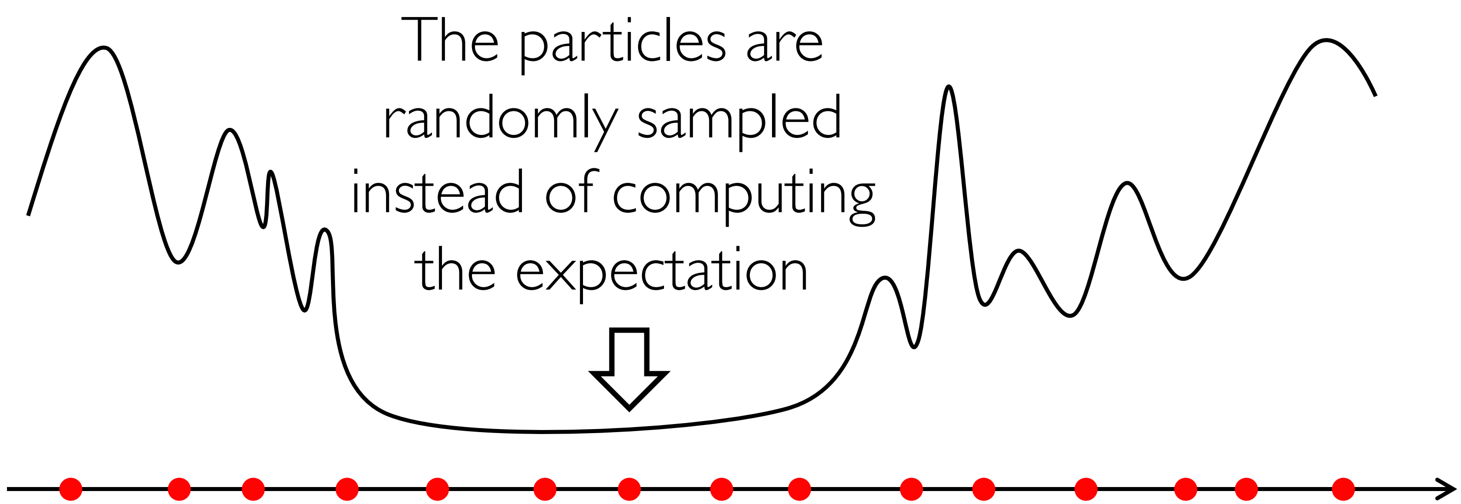
Particle Filter

- Now we explain Particle Filter with an example:
  - Position estimation of the plane.
- **Step 1:** We have particles sampled from $P(\boldsymbol{z}_{t-1}|\boldsymbol{x}_{1:t-1})$.



Particle Filter

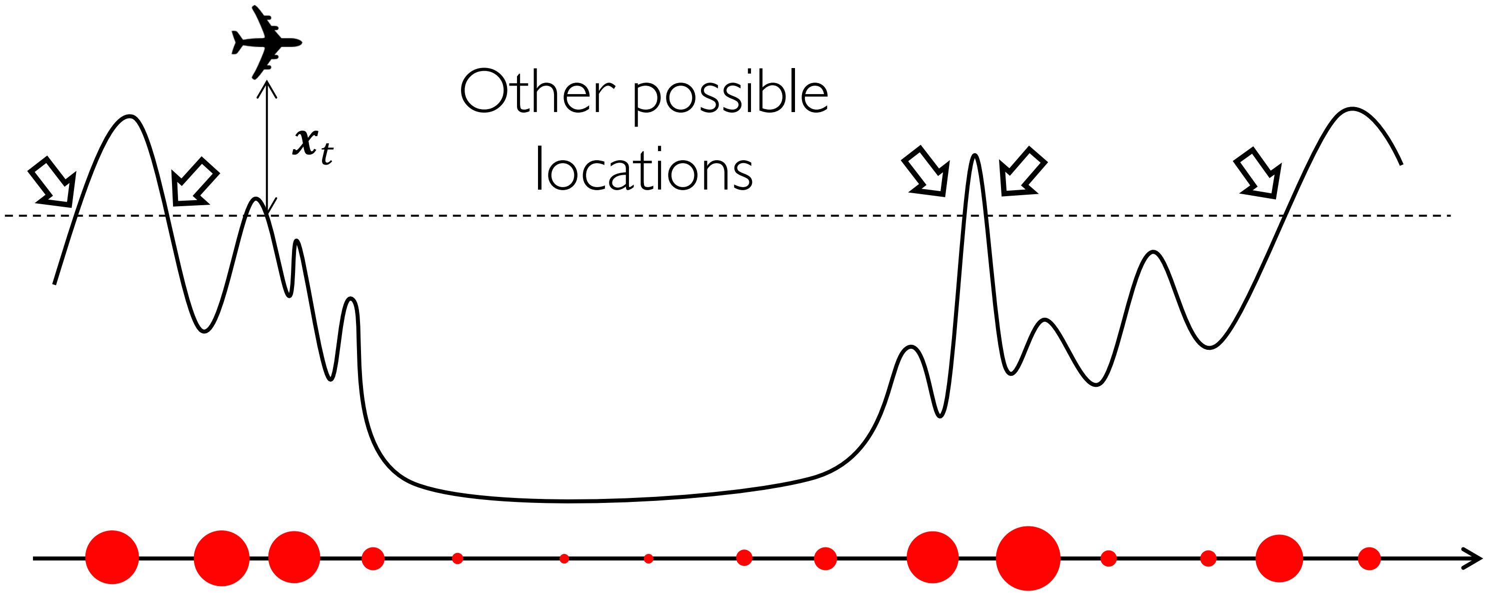
- **Step 2:** Propagate forward.
- A particle in state $\boldsymbol{z}_t$ is moved by sampling its next position directly from the transition model: $\boldsymbol{z}_t \sim P(\boldsymbol{z}_t | \boldsymbol{z}_{t-1})$



Particle Filter

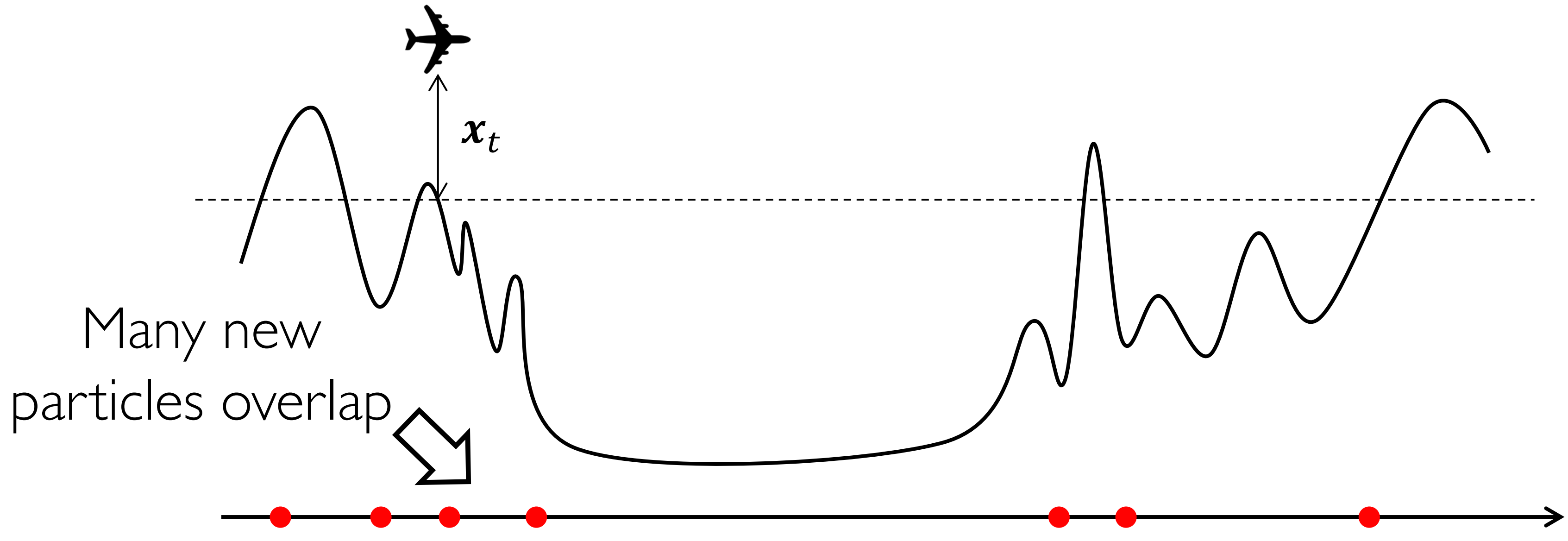
- **Step 3:** Observe.
- Now we have observation $\boldsymbol{x}_t$, we weight samples based on the evidence:

$$\textcolor{blue}{\text{weight}} \propto P(\boldsymbol{x}_t|\boldsymbol{z}_t)$$



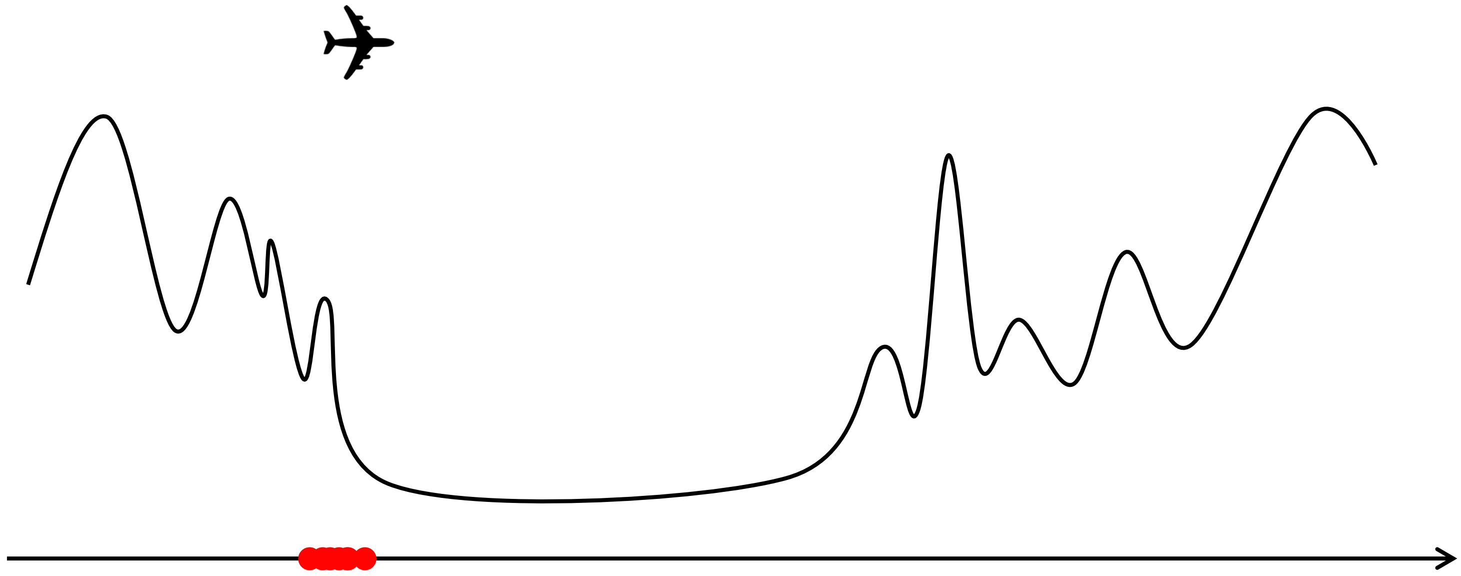
Particle Filter

- **Step 4:** Resample.
- Now sample new particles based on the weights.
- And we get the posterior distribution: $P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$



Particle Filter

- **Iterate** over Step 1 to Step 4, and the particles will finally converge around the real position of the object.



Particle Filter: Pseudocode

```
function PARTICLE-FILTER(transition, emission, initial, observe):
  n = 1000      # number of particles
  for i in range(n):
    x[i] ~ initial                                        (1). Initialize
  ----------------------------------------------------------------------
  for t in range(1, T):
    for i in range(n):
      x_prior[i] ~ transition(x[i])                       (2). Propagate forward
      weight[x_prior[i]] ← emission(observe[t], x_prior[i])
                                      (3). Giving weight based on observation
  ----------------------------------------------------------------------
      for i in range(n):
        x[i] ~ weight                                     (4). Reweight
  return x
```

Thank You

Questions?

Mingsheng Long
mingsheng@tsinghua.edu.cn
http://ise.thss.tsinghua.edu.cn/~mlong
答疑：东主楼11区413室