



Reasoning (V)

Mingsheng Long

Tsinghua University

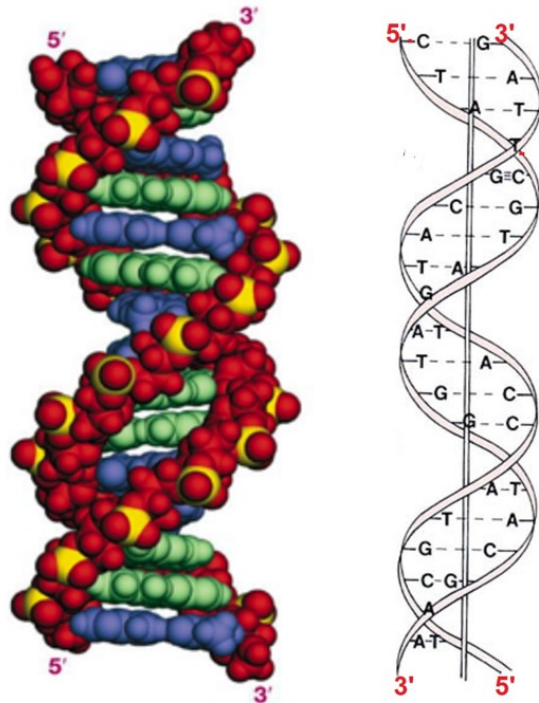
Spring 2023

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

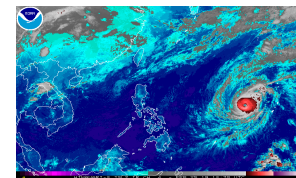
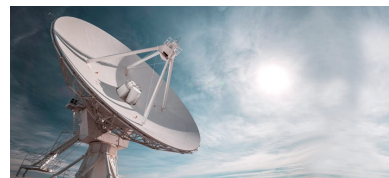
- Sequential data have certain **order** and **dependency**.



DNA sequence

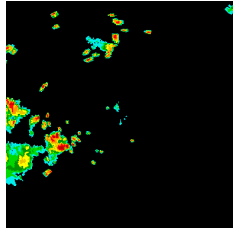
Title: Star's Tux Promise Draws Megyn Kelly's Sarcasm
Subtitle: Joaquin Phoenix pledged to not change for each awards event
Article: A year ago, Joaquin Phoenix made headlines when he appeared on the red carpet at the Golden Globes wearing a tuxedo with a paper bag over his head that read, "I am a shape-shifter. 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



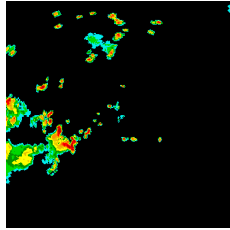
Weather

Discrete Time Stochastic Process



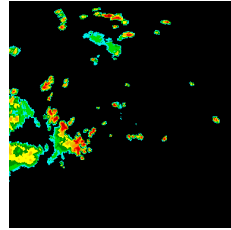
$$\mathbf{x}_1$$

$$t = 1$$



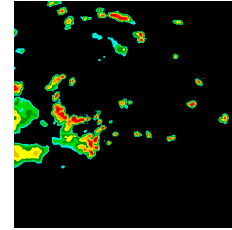
$$\mathbf{x}_2$$

$$t = 2$$



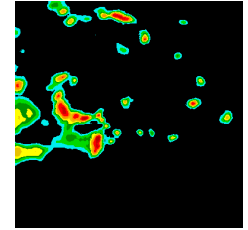
$$\mathbf{x}_3$$

$$t = 3$$



$$\mathbf{x}_4$$

$$t = 4$$



$$\mathbf{x}_5$$

$$t = 5$$

- After sampling in time, we can model sequential data with a **discrete time stochastic process**.
- We call the random variable \mathbf{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:

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

- The Markov assumption implies that:

$$P(\mathbf{x}_1, \dots, \mathbf{x}_T) = P(\mathbf{x}_1) \prod_t P(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \dots, \mathbf{x}_1) \quad \text{Any process: } \mathcal{O}(K^T)$$



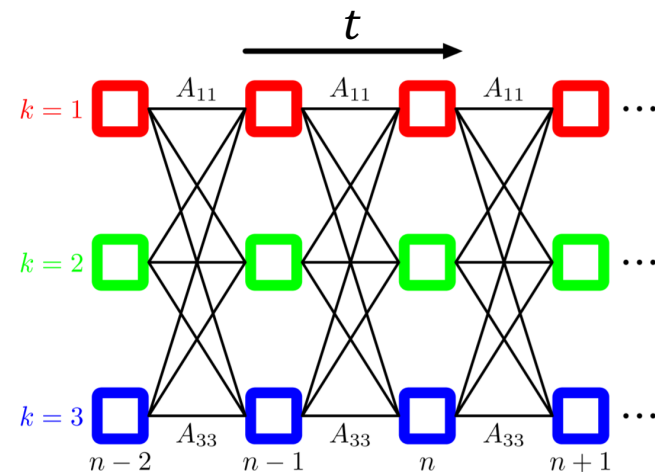
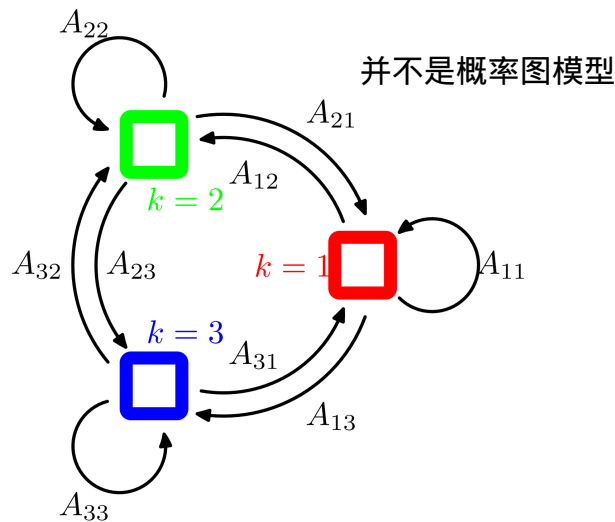
Markov assumption

$$= P(\mathbf{x}_1) \prod_t P(\mathbf{x}_t | \mathbf{x}_{t-1}) \quad \text{Markov process: } \mathcal{O}(K^2)$$

State Transition Matrix

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

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



State transition diagram:
 nodes \rightarrow states
 edges \rightarrow state transition matrix

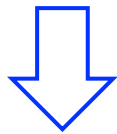
If we unfold the state transition diagram over time, we obtain a lattice.

Outline

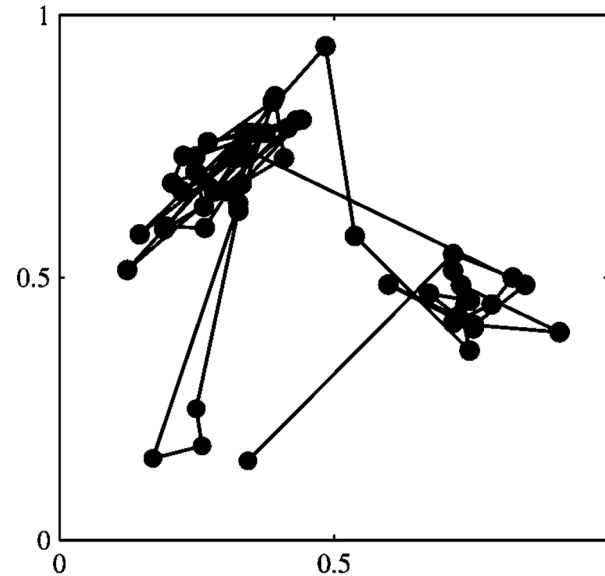
- Hidden Markov Model
 - Markov Model
 - **Hidden Markov Model**
 - Inference for Hidden Markov Model
 - Viterbi Algorithm
 - Forward-Backward Algorithm
- Linear Dynamical System
 - Kalman Filter
 - Particle Filter

Observation without Markov Property

The data tend to
gather in a group



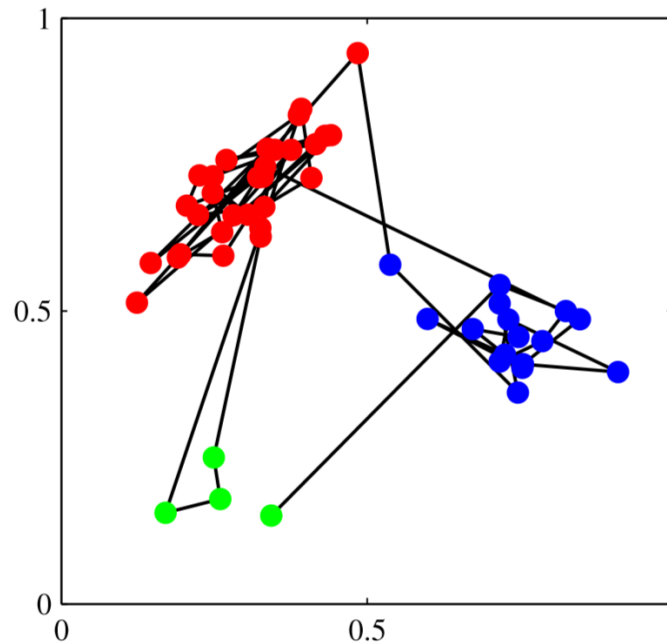
No Markov property



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?

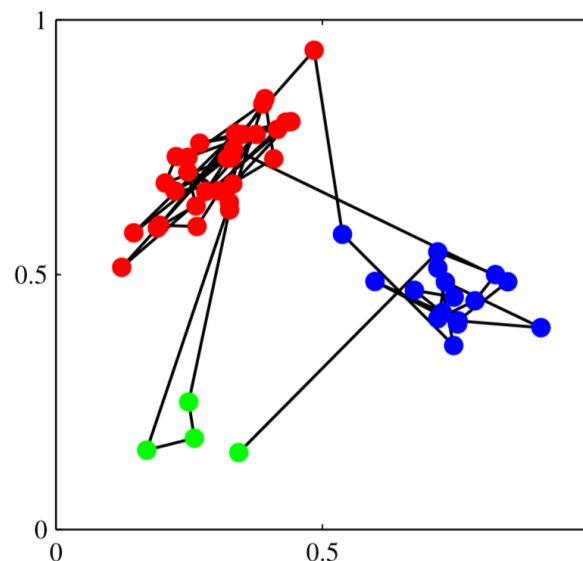
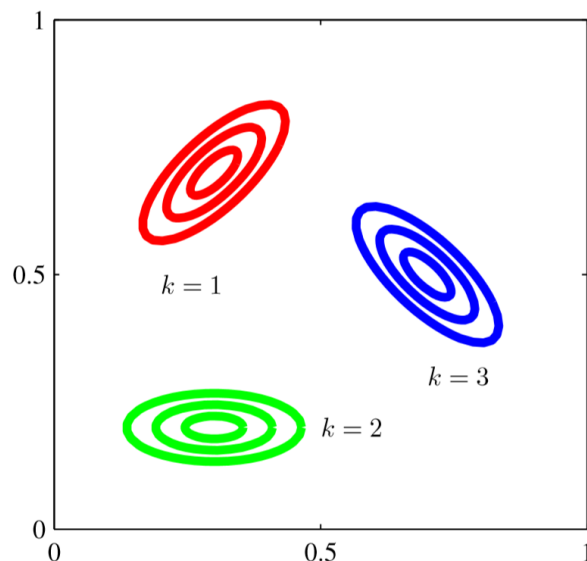
Observation without Markov Property



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)



假设某数据来自红色区域，那可以假设该点满足红色部分的高斯分布

但是数据属于某类簇使用markov转移矩阵描述

- **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**:

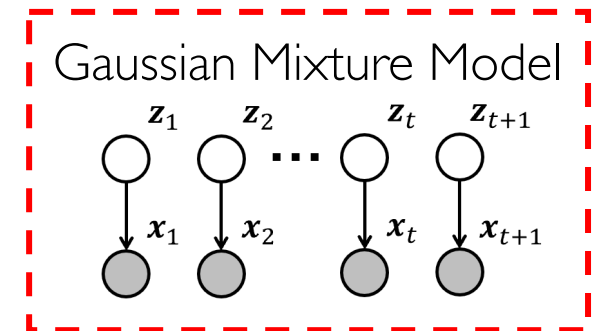
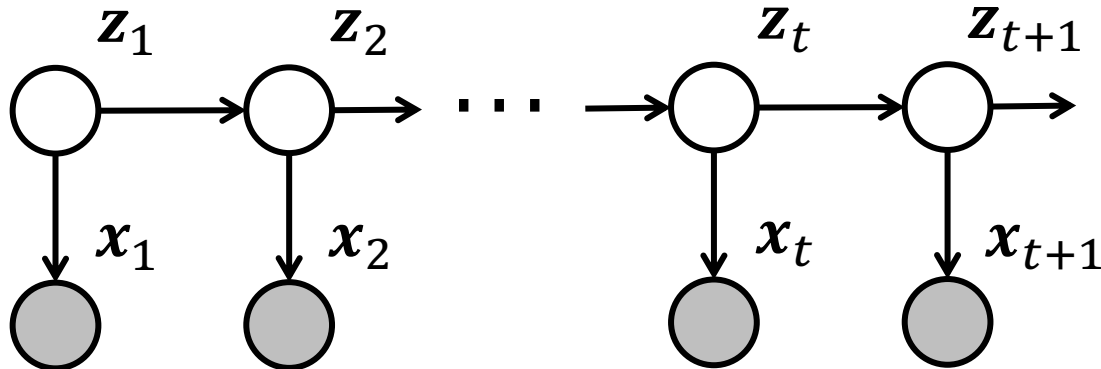
- The hidden state \mathbf{z}_t is a Markov chain.

Transition distribution: $P(\mathbf{z}_t | \mathbf{z}_{1:t-1}, \mathbf{x}_{1:t}) = P(\mathbf{z}_t | \mathbf{z}_{t-1})$ 条件独立性

- The observation \mathbf{x}_t is dependent only on the current hidden state.

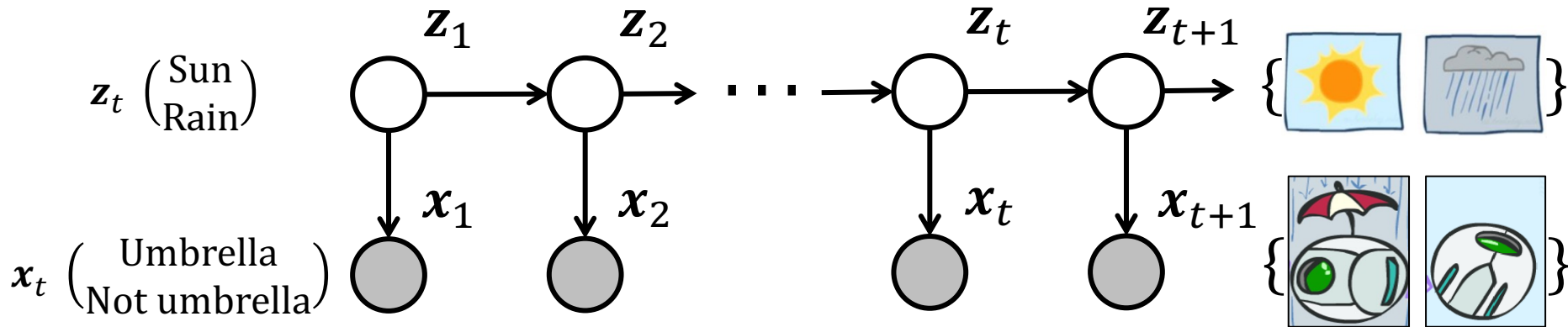
Emission distribution: $P(\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{x}_{1:t}) = P(\mathbf{x}_t | \mathbf{z}_t)$

假设只给概率图，那么 x_i, x_j 不独立（有通路）；若中间的一个 z_k 给定，则独立



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

Example: Weather Forecasting



Transitions: $P(z_t | z_{t-1})$

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

Emissions: $P(x_t | z_t)$

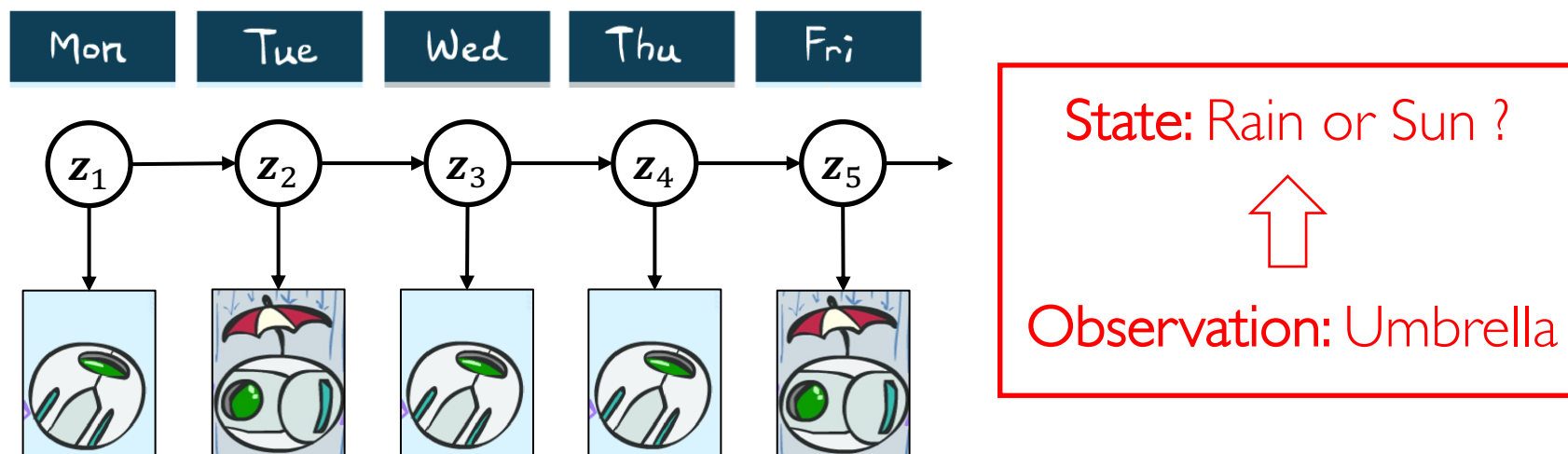
$z_t \backslash x_t$	Umbrella	Not umbrella
Rain	0.9	0.1
Sun	0.2	0.8

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 $\hat{\mathbf{X}} = \{\mathbf{x}_1, \dots, \mathbf{x}_T\}$, how to estimate the hidden states $\hat{\mathbf{Z}} = \{\mathbf{z}_1, \dots, \mathbf{z}_T\}$?



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

EM Algorithm

- Choose initial θ^{old} .

- Expectation Step



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

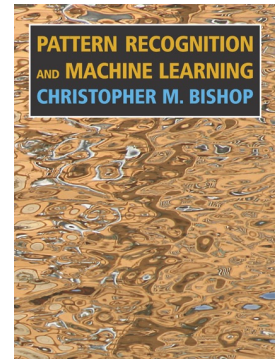
- Let $q^*(z) = p(z|x, \theta^{\text{old}})$. [q^* gives best lower bound at θ^{old}]

- Let $J(\theta) := \mathcal{L}(q^*, \theta) = \underbrace{\sum_z q^*(z) \log \left(\frac{p(x, z|\theta)}{q^*(z)} \right)}_{\text{Expectation w.r.t. } z \sim q^*(z)}$

- Maximization Step

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

- 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:

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

$$\hat{\mathbf{Z}} = \arg \max_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}) \propto \arg \max_{\mathbf{Z}} p(\mathbf{Z}, \mathbf{X})$$



Viterbi algorithm

- Minimize the state error rate:

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

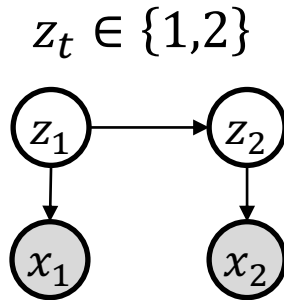
$$\hat{\mathbf{z}}_t = \arg \max_{\mathbf{z}_t} p(\mathbf{z}_t|\mathbf{X})$$



Forward-Backward algorithm

Example: Binary HMM

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



$\begin{matrix} z_2 \\ z_1 \end{matrix}$	1	2
1	0.04	0.3
2	0.36	0.3

- Minimize **sequence** error rate:

$$E(L(\mathbf{Z}, \hat{\mathbf{Z}})|\mathbf{X}) = 1 - P(\mathbf{Z} = \hat{\mathbf{Z}}|\mathbf{X}) \Rightarrow \hat{\mathbf{Z}} = \{\mathbf{1}, 2\}$$

- Minimize **state** error rate:

$$E(L(\mathbf{Z}, \hat{\mathbf{Z}})|\mathbf{X}) = 1 - P(z_1 = \hat{z}_1|\mathbf{X}) + 1 - P(z_2 = \hat{z}_2|\mathbf{X})$$

$$\Rightarrow \hat{\mathbf{Z}} = \{\mathbf{2}, 2\}$$

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.

利用贝叶斯决策准则把误差最小转化为最大化后验

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

图模型定义

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

↑ Transition ↑ Emission

取负对数似然

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

minimizing
negative log-
probability.

高斯分布

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

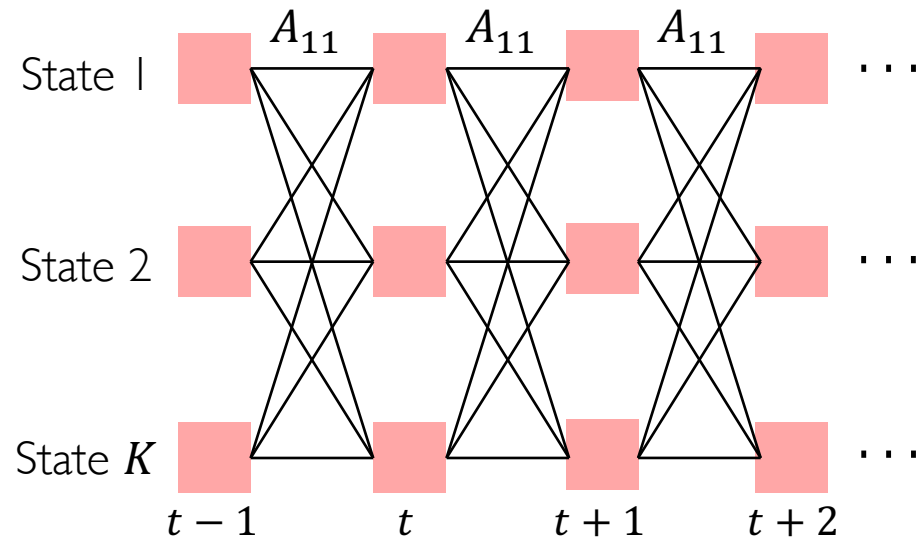
多项分布 (由于离散)

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

Lattice Diagram

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

- Node weight: $\phi_t(\mathbf{z}_t)$, edge weight: $\psi_t(\mathbf{z}_t, \mathbf{z}_{t-1})$
- 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

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

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

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

- We can **recursively** write this minimization as follows:

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

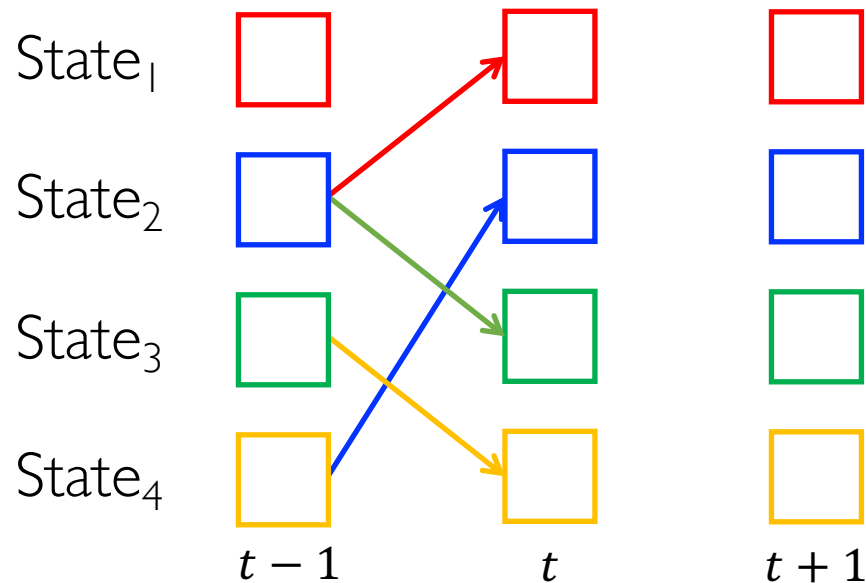
- **Interpretation:** every path up to time t has the following form:

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

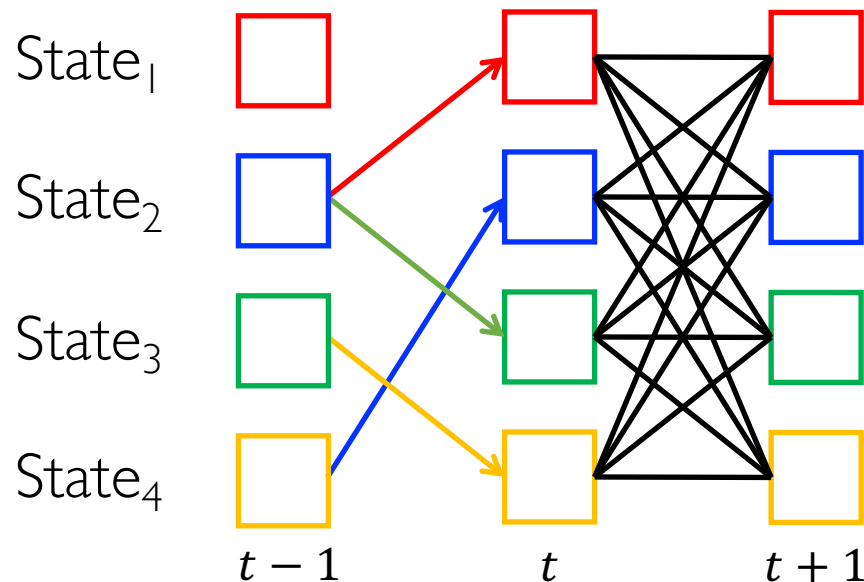
Viterbi Algorithm

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



Viterbi Algorithm

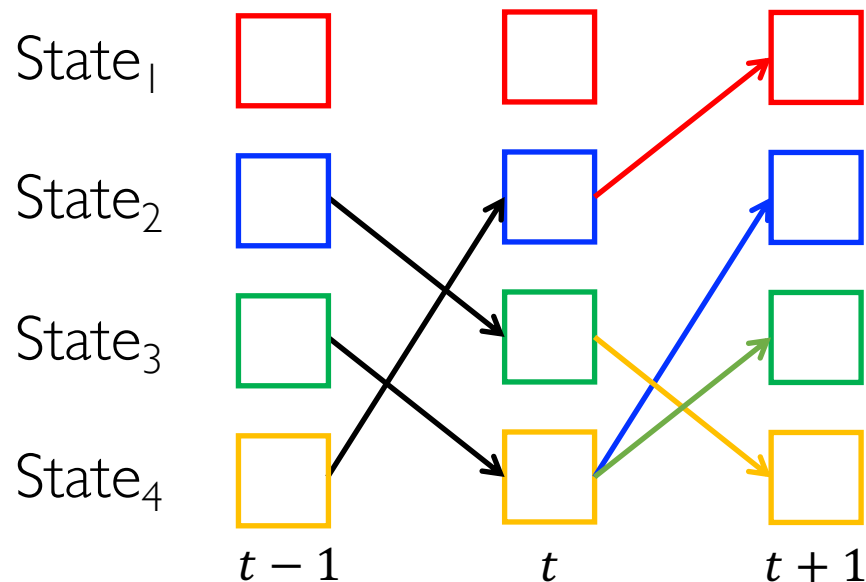
- From time t to $t + 1$:
 - Evaluate the weights $\psi_t(\mathbf{z}_t, \mathbf{z}_{t-1})$ of every edge from t to $t + 1$.
 - K paths arriving at each node k , $\mathcal{O}(K^2)$ computation cost in total.



Viterbi Algorithm

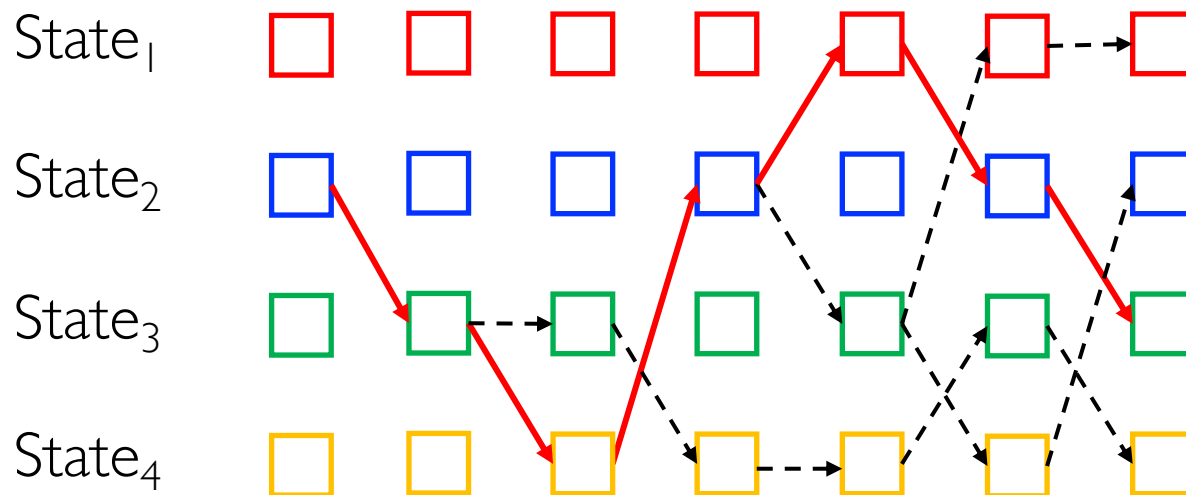
- From time t to $t + 1$:
 - Update the best path ending with each state \mathbf{z}_t .

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



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**:

$$P(\mathbf{z}_t | \mathbf{x}_{1:T}) = P(\mathbf{x}_{1:T}, \mathbf{z}_t) / P(\mathbf{x}_{1:T})$$

$$= P(\mathbf{x}_{1:T} | \mathbf{z}_t) P(\mathbf{z}_t) / P(\mathbf{x}_{1:T})$$

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

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

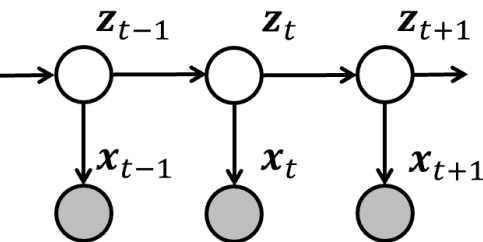
$$= P(\mathbf{z}_t | \mathbf{x}_{1:t}) P(\mathbf{x}_{1:t}) P(\mathbf{x}_{t+1:T} | \mathbf{z}_t) / P(\mathbf{x}_{1:T})$$

$$\propto P(\mathbf{z}_t | \mathbf{x}_{1:t}) P(\mathbf{x}_{t+1:T} | \mathbf{z}_t)$$

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

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

Conditional
independency
of HMM



Forward Algorithm

- Physics meaning: estimate each hidden state online.

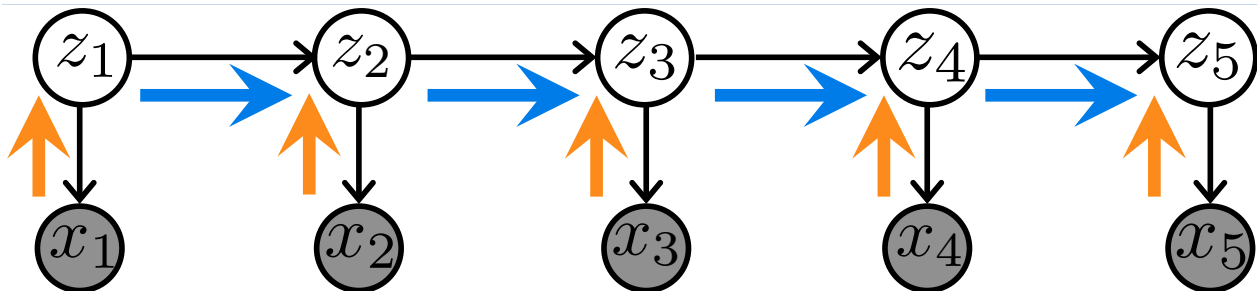
Define $\alpha_t(\mathbf{z}_t) = P(\mathbf{z}_t | \mathbf{x}_{1:t})$

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

$$\text{Posterior distribution} \quad \alpha_t(\mathbf{z}_t) = c_t P(\mathbf{x}_t | \mathbf{z}_t) \underset{\substack{\uparrow \\ \text{Normalizer}}}{P(\mathbf{z}_t | \mathbf{x}_{1:t-1})}$$

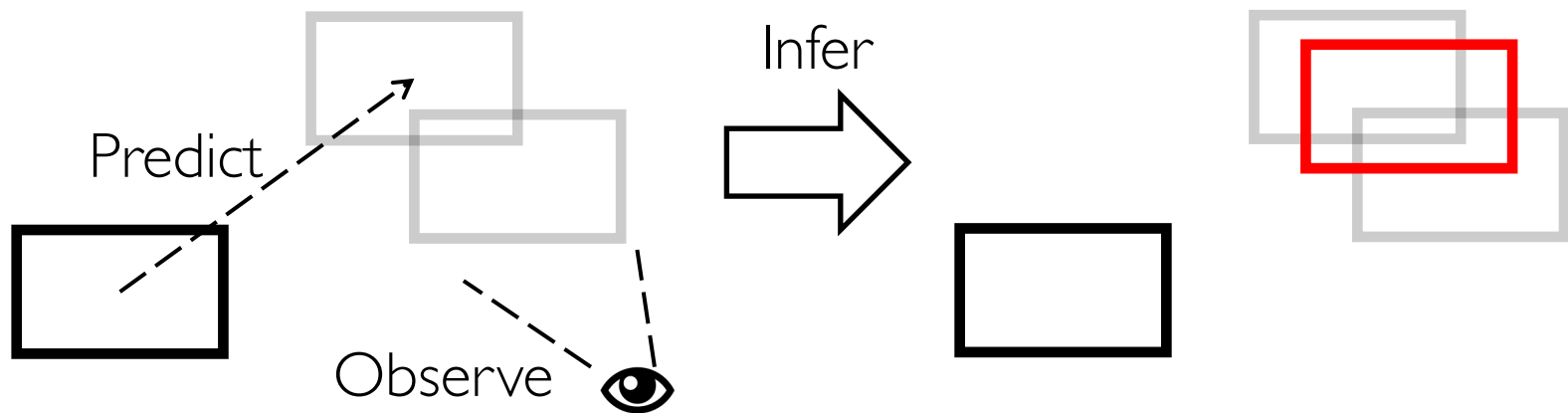
- Recursively proceed forward through the HMM network:

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



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

$$\begin{aligned} P(\mathbf{z}_t | \mathbf{x}_{1:t-1}) &= \sum_{\mathbf{z}_{t-1}} P(\mathbf{z}_t, \mathbf{z}_{t-1} | \mathbf{x}_{1:t-1}) \quad (\text{Total probability}) \\ &= \sum_{\mathbf{z}_{t-1}} P(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x}_{1:t-1}) P(\mathbf{z}_{t-1} | \mathbf{x}_{1:t-1}) \\ &\quad \text{zt只依赖zt-1} \\ &= \sum_{\mathbf{z}_{t-1}} P(\mathbf{z}_t | \mathbf{z}_{t-1}) \alpha_{t-1}(\mathbf{z}_{t-1}) \end{aligned}$$

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

Forward Algorithm: Posterior Distribution

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

$$\begin{aligned}\alpha_t(\mathbf{z}_t) = P(\mathbf{z}_t|\mathbf{x}_{1:t}) &= \frac{P(\mathbf{z}_t, \mathbf{x}_t|\mathbf{x}_{1:t-1})}{P(\mathbf{x}_t|\mathbf{x}_{1:t-1})} \\ &\propto P(\mathbf{z}_t, \mathbf{x}_t|\mathbf{x}_{1:t-1}) \quad (\text{Normalize of } \mathbf{x}_t) \\ &= P(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_t)P(\mathbf{z}_t|\mathbf{x}_{1:t-1}) \quad \text{条件概率*边缘分布} \\ &= P(\mathbf{x}_t|\mathbf{z}_t)P(\mathbf{z}_t|\mathbf{x}_{1:t-1}) \quad (\text{Markov dependency})\end{aligned}$$

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

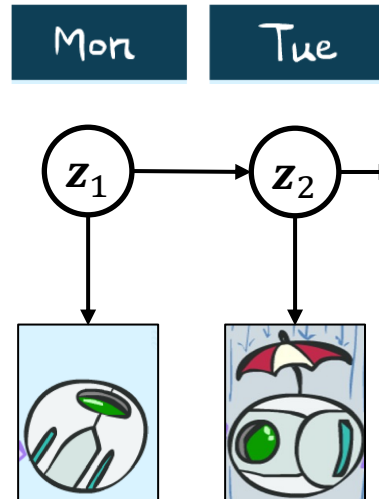
Example: Weather Forecasting

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

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

Emission Matrix

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

Example: Weather Forecasting

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

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

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

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

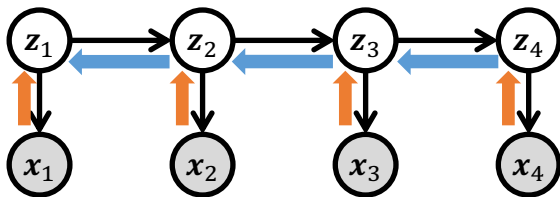
$$\alpha_2(\mathbf{z}_2) \propto P(\mathbf{x}_2 | \mathbf{z}_2) P(\mathbf{z}_2 | \mathbf{x}_1) = \begin{pmatrix} 0.2 \\ 0.9 \end{pmatrix} \times \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} = \begin{pmatrix} 0.12 \\ 0.36 \end{pmatrix}$$

\searrow $P(\text{sun})=0.25$
 $P(\text{rain})=0.75$

Backward Algorithm

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

$$\begin{aligned}
 \beta_t(\mathbf{z}_t) &= P(\mathbf{x}_{t+1:T} | \mathbf{z}_t) = \sum_{\mathbf{z}_{t+1}} P(\mathbf{x}_{t+1:T}, \mathbf{z}_{t+1} | \mathbf{z}_t) \quad \text{加法公式} \\
 &= \sum_{\mathbf{z}_{t+1}} P(\mathbf{x}_{t+1:T} | \mathbf{z}_{t+1}, \mathbf{z}_t) P(\mathbf{z}_{t+1} | \mathbf{z}_t) \\
 &\quad \text{条件独立性} \\
 &= \sum_{\mathbf{z}_{t+1}} P(\mathbf{x}_{t+1:T} | \mathbf{z}_{t+1}) P(\mathbf{z}_{t+1} | \mathbf{z}_t) \\
 &\quad \text{概率乘法公式} \\
 &= \sum_{\mathbf{z}_{t+1}} \underbrace{P(\mathbf{x}_{t+2:T} | \mathbf{z}_{t+1})}_{\beta_{t+1}(\mathbf{z}_{t+1})} \underbrace{P(\mathbf{x}_{t+1} | \mathbf{z}_{t+1})}_{\text{Emission}} \underbrace{P(\mathbf{z}_{t+1} | \mathbf{z}_t)}_{\text{Transition}}
 \end{aligned}$$



Recursively run this algorithm backward

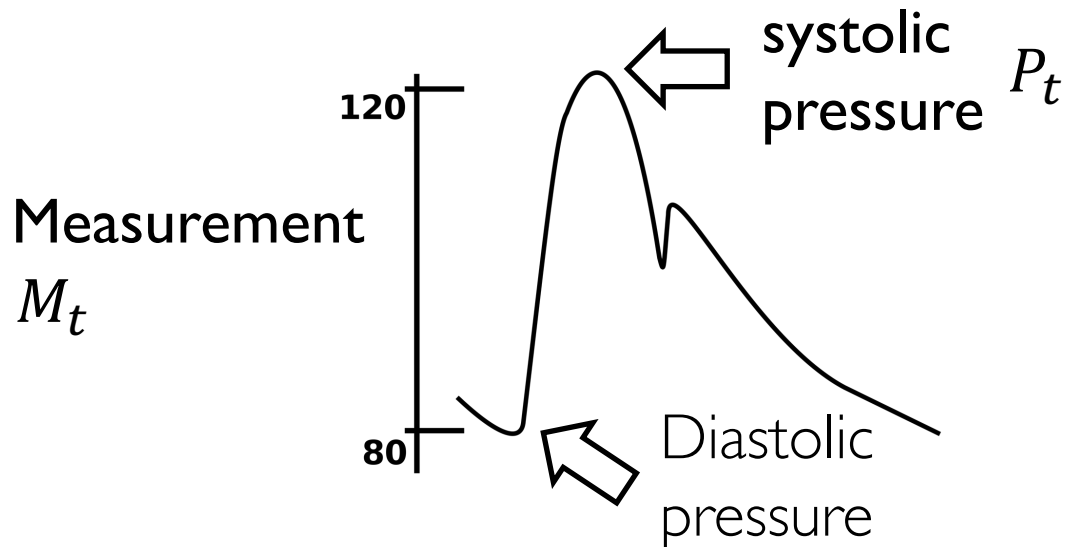
Outline

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



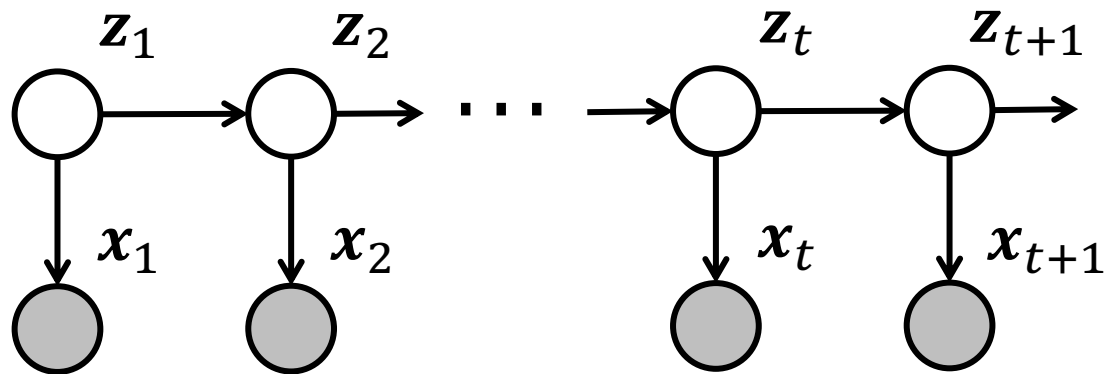
Track of Blood Pressure

- 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**.



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 $P(z_t | z_{1:t-1}, x_{1:t}) = P(z_t | z_{t-1}), \quad z_i \in \mathbb{R}$

Emission $P(x_t | z_{1:t}, x_{1:t}) = P(x_t | z_t), \quad x_i \in \mathbb{R}$

Inference of LDS

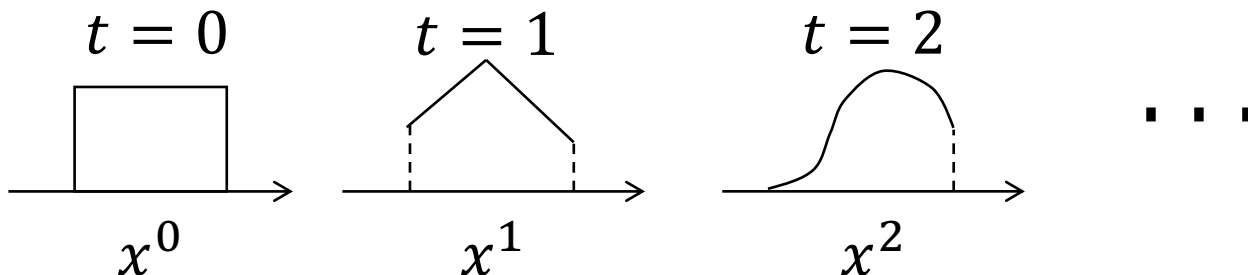
- Recall the forward algorithm of HMM:

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

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

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

- 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

$$F(\theta_0) \quad F(\theta_1) \quad \bullet \bullet \bullet \quad F(\theta_t)$$



Form change

$$F(\theta_0) \quad F^2(\theta_1) \quad \bullet \bullet \bullet \quad F^t(\theta_t)$$



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

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

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

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

Gaussian Distribution

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

Transition $\mathbf{z}_{t+1} = \mathbf{A}\mathbf{z}_t + \mathbf{w}_n, \quad \mathbf{w} \sim \mathcal{N}(\mathbf{w}|0, \mathbf{\Gamma})$

Emission $\mathbf{x}_t = \mathbf{C}\mathbf{z}_t + \mathbf{v}_t, \quad \mathbf{v} \sim \mathcal{N}(\mathbf{v}|0, \mathbf{\Sigma})$

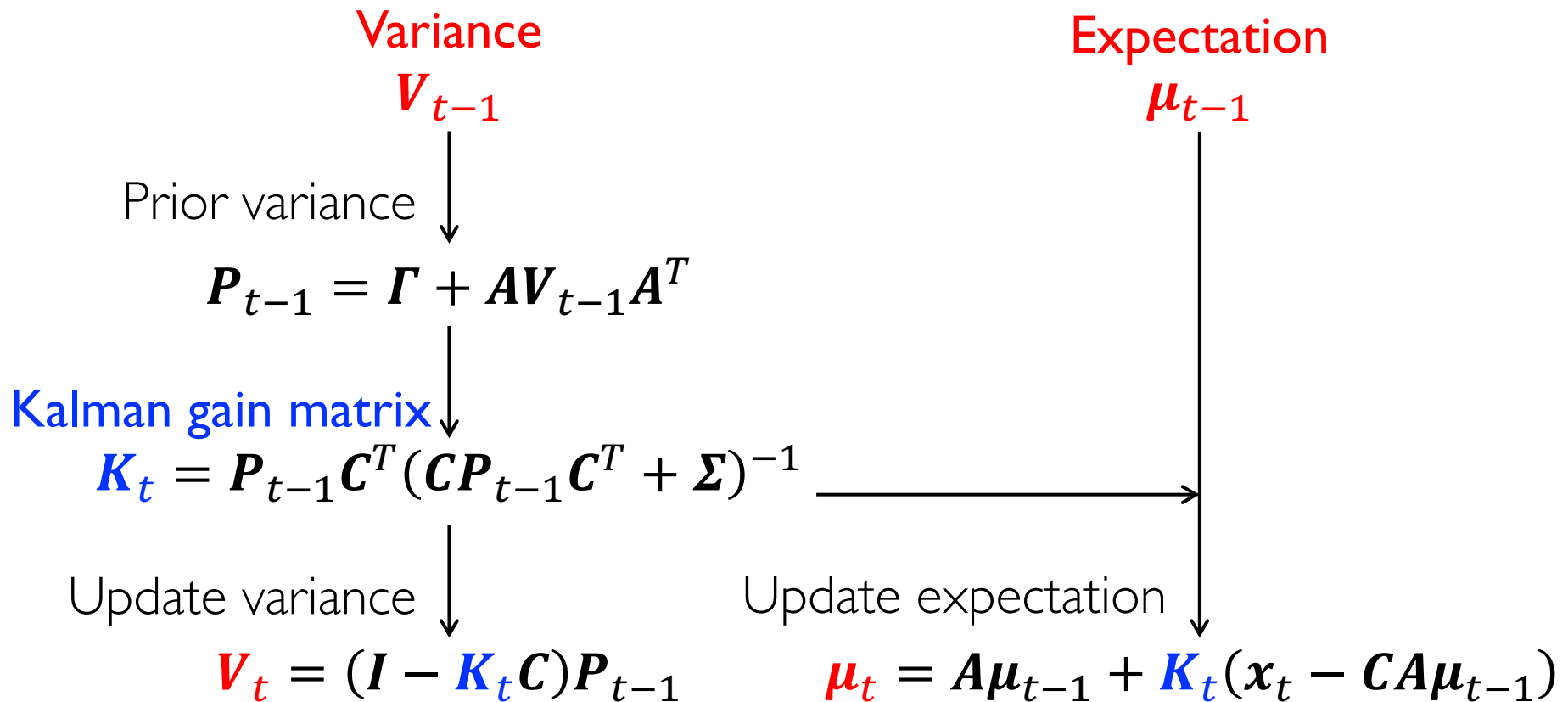
Initialization $\mathbf{z}_1 = \boldsymbol{\mu}_0 + \mathbf{u}, \quad \mathbf{u} \sim \mathcal{N}(\mathbf{u}|0, \mathbf{V}_0)$

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

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

Kalman Filter

- Given the expectation μ_{t-1} variance V_{t-1} , with new observation x_t , we can evaluate the posterior distribution of z_t with **Kalman Filter**:



Interpretation of Expectation Update

$$\mu_t = A\mu_{t-1} + K_t(x_t - CA\mu_{t-1})$$

- We first predict the expectation μ using the transition matrix A :

$$\hat{\mu}_t = A\mu_{t-1}$$

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

$$x_t - C\hat{\mu}_t$$

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

$$\mu_t = \hat{\mu}_t + K_t(x_t - C\hat{\mu}_t)$$

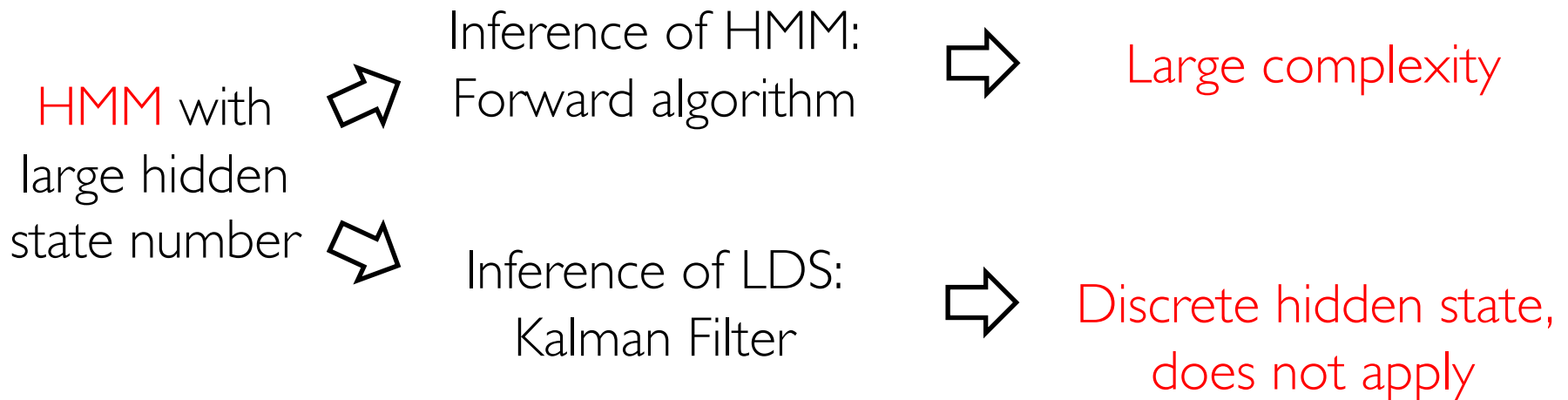
- The coefficient K_t is proportional to the prior variance P_{t-1} : if the prior prediction is accurate (variance is small), the K_t will be small (“trust” the prediction), and vice versa.

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**

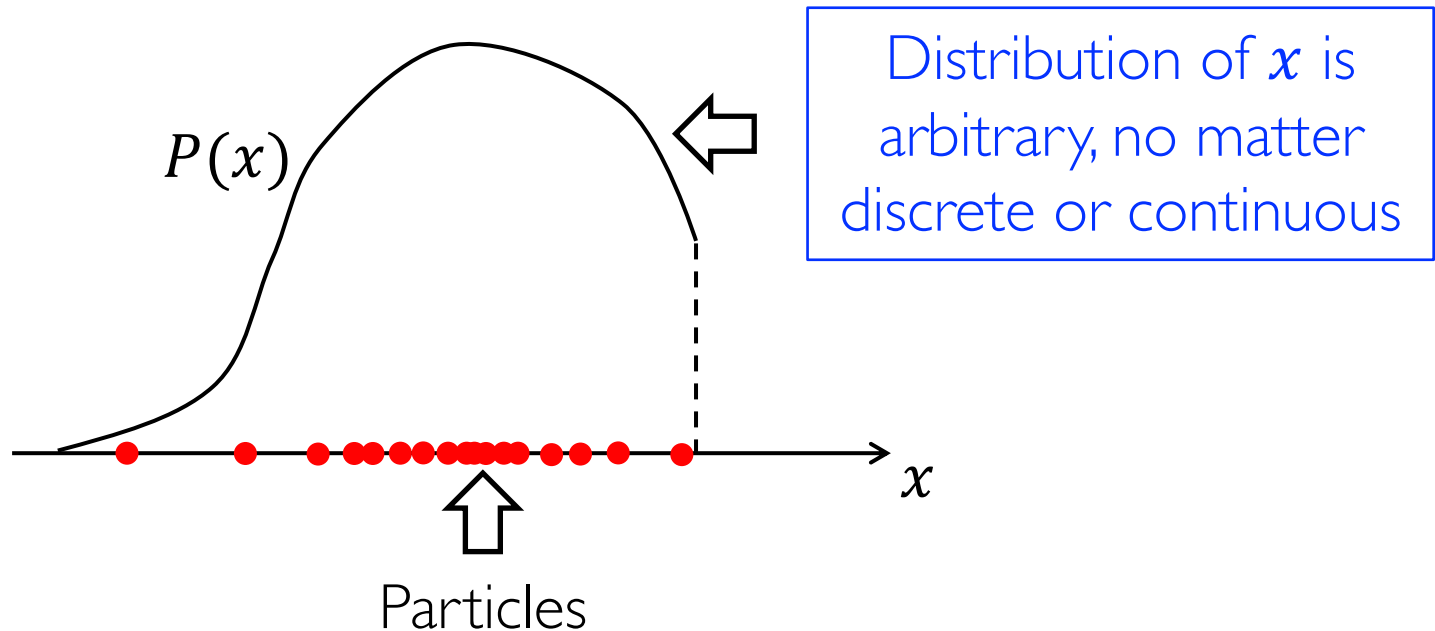
Motivation

- With the assumption of Gaussian distribution, we reduce 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.



Monte Calo Method

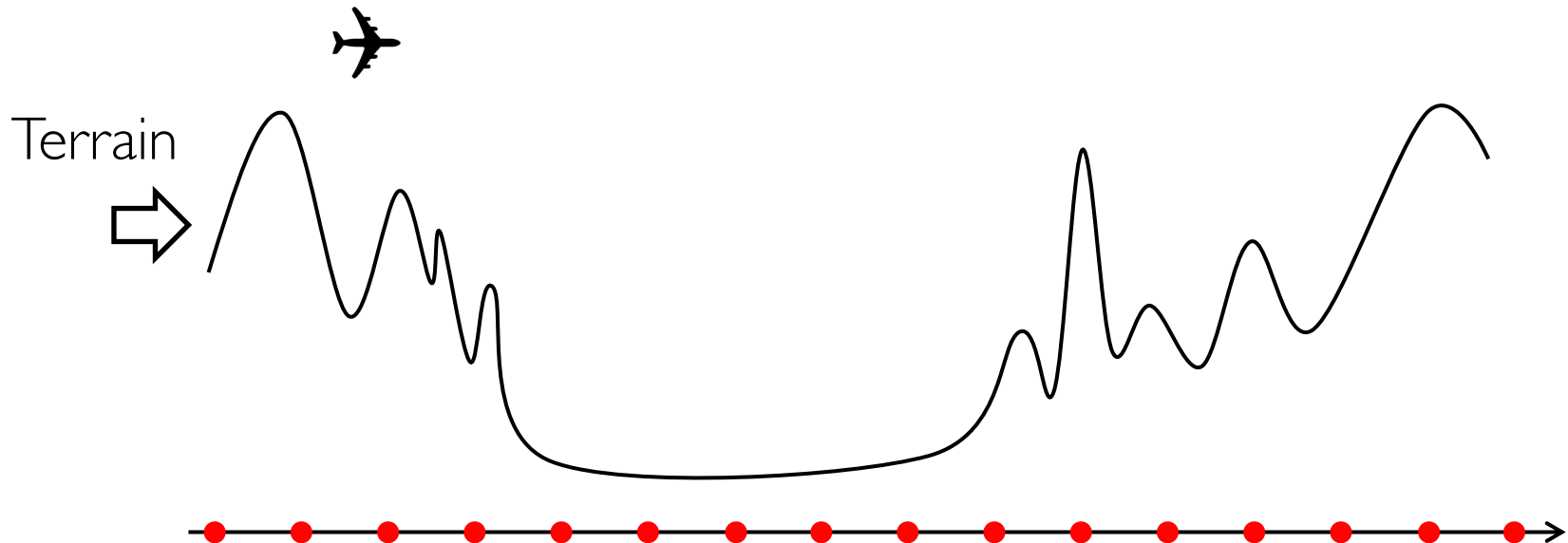
- Particle Filter uses Monte Calo method to solve the problem.
- We use the samples from a distribution to express it, instead of analytical expression:
 - The samples are called **particles**.





Particle Filter

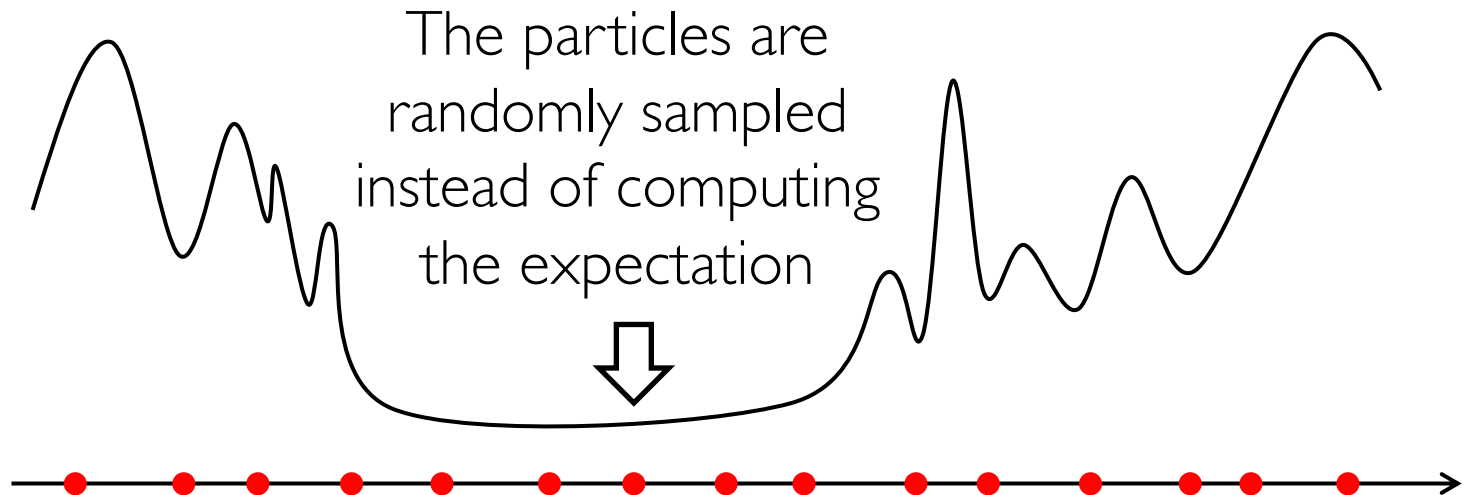
- Now we explain Particle Filter with an example:
 - Position estimation of the plane.
- Step 1: We have particles sampled from $P(\mathbf{z}_{t-1} | \mathbf{x}_{1:t-1})$.





Particle Filter

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

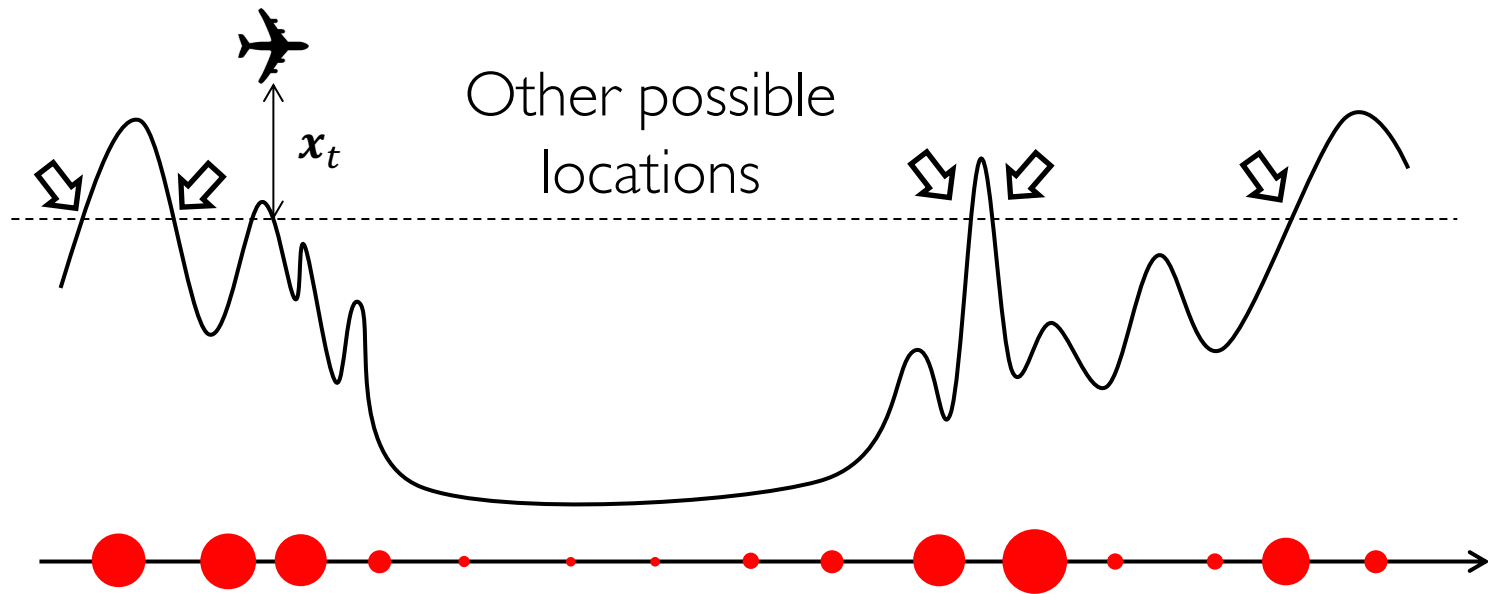




Particle Filter

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

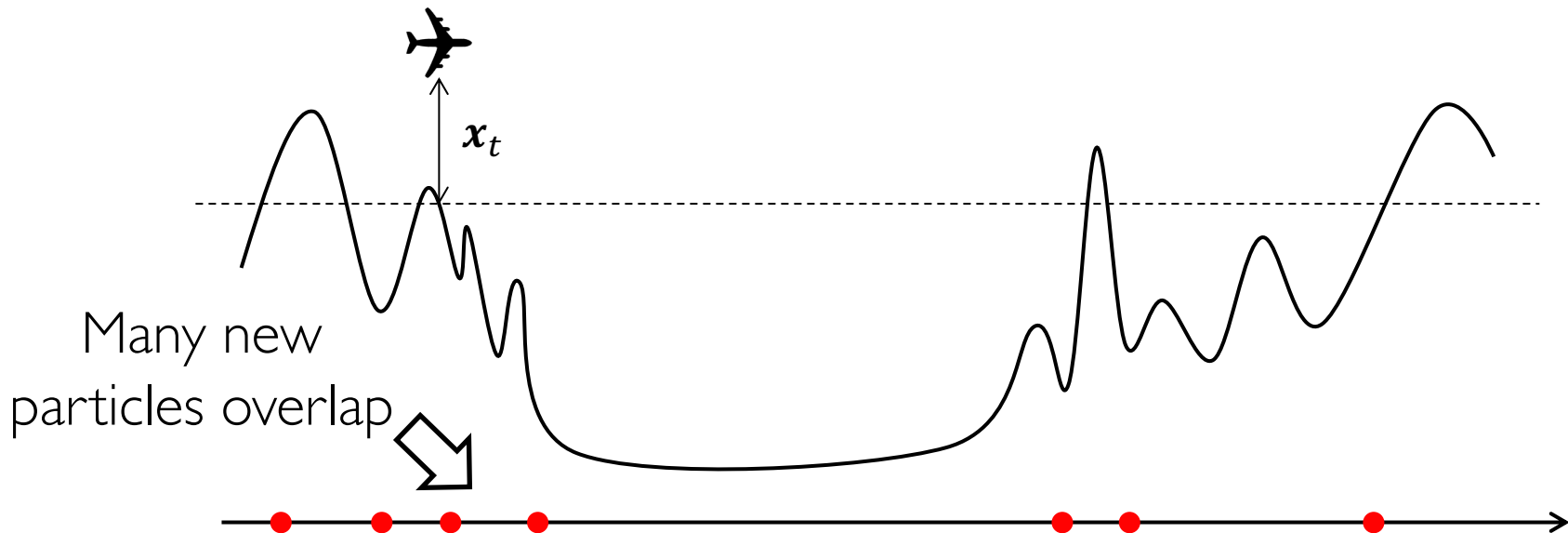
$$\text{weight} \propto P(\mathbf{x}_t | \mathbf{z}_t)$$





Particle Filter

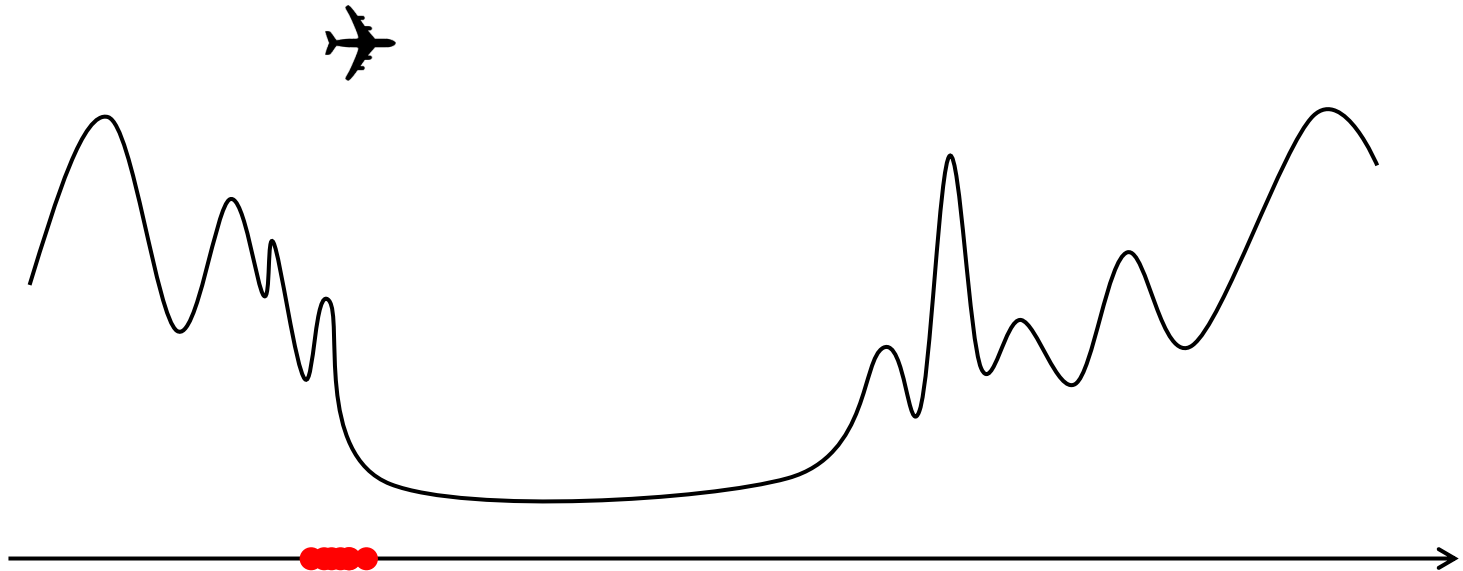
- Step 4: Resample.
- Now sample new particles based on the weights.
- And we get the posterior distribution: $P(\mathbf{z}_t | \mathbf{x}_{1:t})$





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

```
1. function PARTICLE-FILTER(transition, emission, initial, observe):
2.    $n = 1000$       # number of particles
3.   for  $i$  in range( $n$ ):
4.      $x[i] \sim \textit{initial}$                                      (1). Initialize
-----
5.   for  $t$  in range(1,  $T$ ):
6.     for  $i$  in range( $n$ ):
7.        $x\_prior[i] \sim \textit{transition}(x[i])$                (2). Propagate forward
8.        $weight[x\_prior[i]] \leftarrow \textit{emission}(\textit{observe}[t], x\_prior[i])$ 
9.                                     (3). Giving weight based on observation
-----
10.    for  $i$  in range( $n$ ):
11.       $x[i] \sim \textit{weight}$                                    (4). Reweight
12. 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