Machine Learning

Sequential Data

Markov Chains

Hidden Markov Models

State space models

Lesson 11

Sequential Data

- Consider a system which can occupy one of *N* discrete ***states*** or ***categories.***
- x_t : state at time t ***Discrete*** $x_t \in \{s_1, s_2, \ldots, s_K\}$ *or* ***Continue*** $x_t \in R^d$
- Sequential data of length T: $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$
- We are interested in ***stochastic*** systems, in which state evolution is random
- Any ***joint distribution*** can be factored into a series of ***conditional*** distributions:

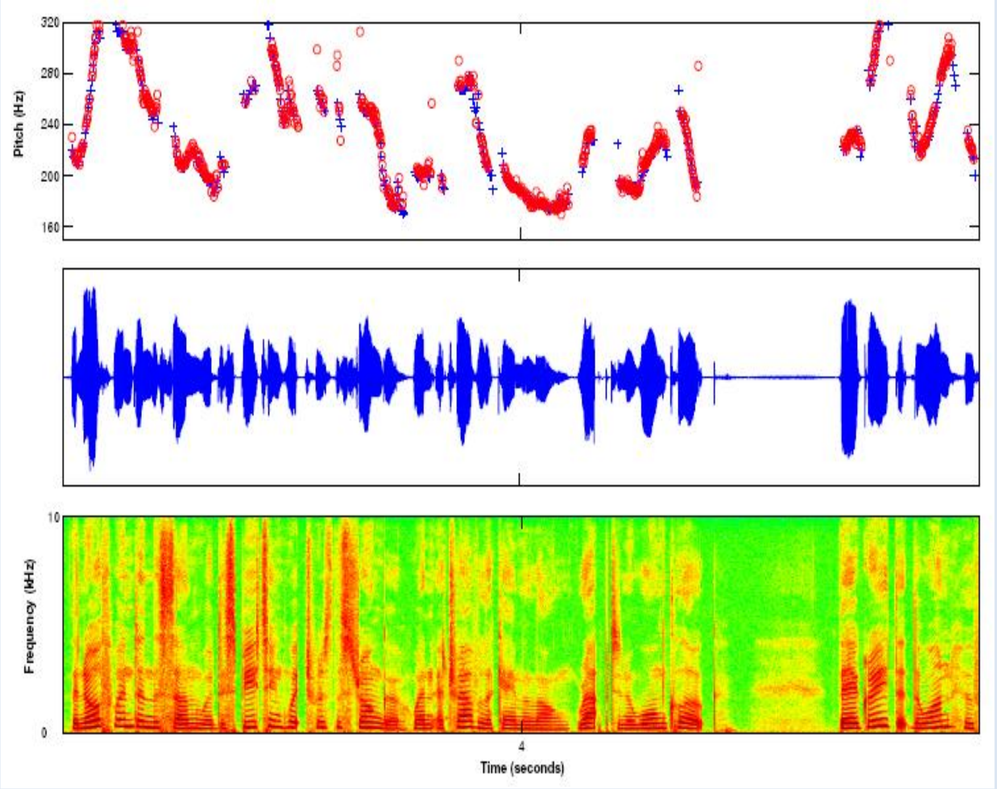
$$p(\mathbf{x}) = p(x_1, x_2, \ldots, x_T) = p(x_1) \prod_{t=2}^{T} p(x_t \mid x_0, \ldots, x_{t-1})$$

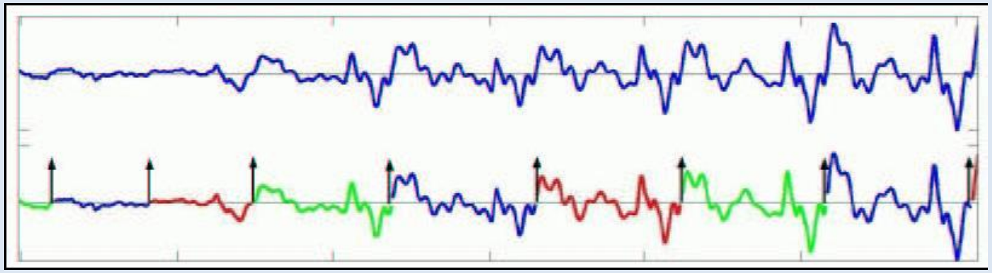
Analysis of Sequential Data

- Sequential structure arises in a huge range of **applications**
  - Repeated measurements of a temporal process
  - Online decision making & control
  - Text, biological sequences, etc
- Standard machine learning methods are often **difficult to directly apply**
  - Do not exploit temporal correlations
  - Computation & storage requirements typically scale poorly to realistic applications

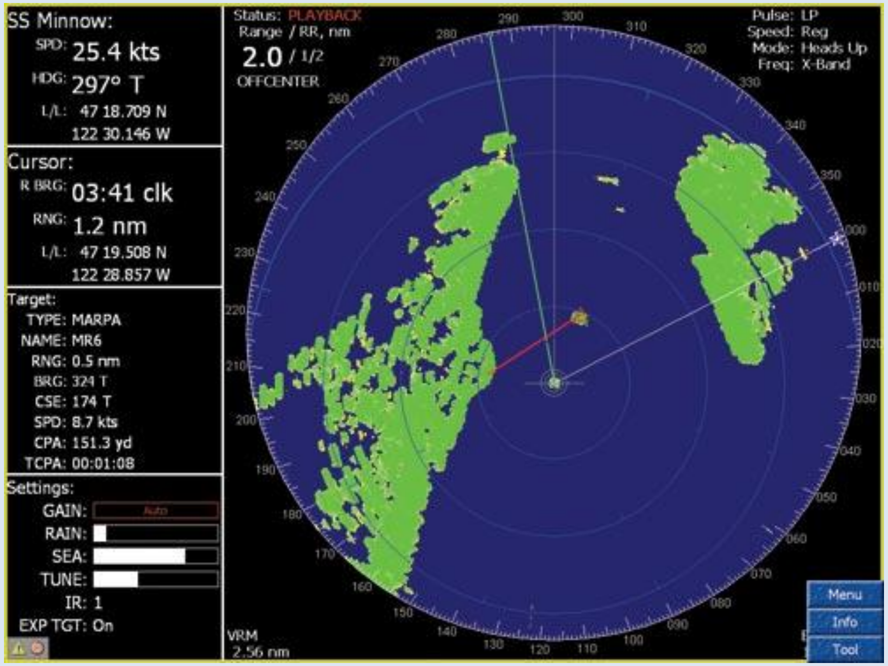
Speech Recognition

- Given an audio waveform, would like to robustly extract & recognize any spoken words
- Statistical models can be used to
  - Provide greater robustness to noise
  - Adapt to accent of different speakers
  - Learn from training

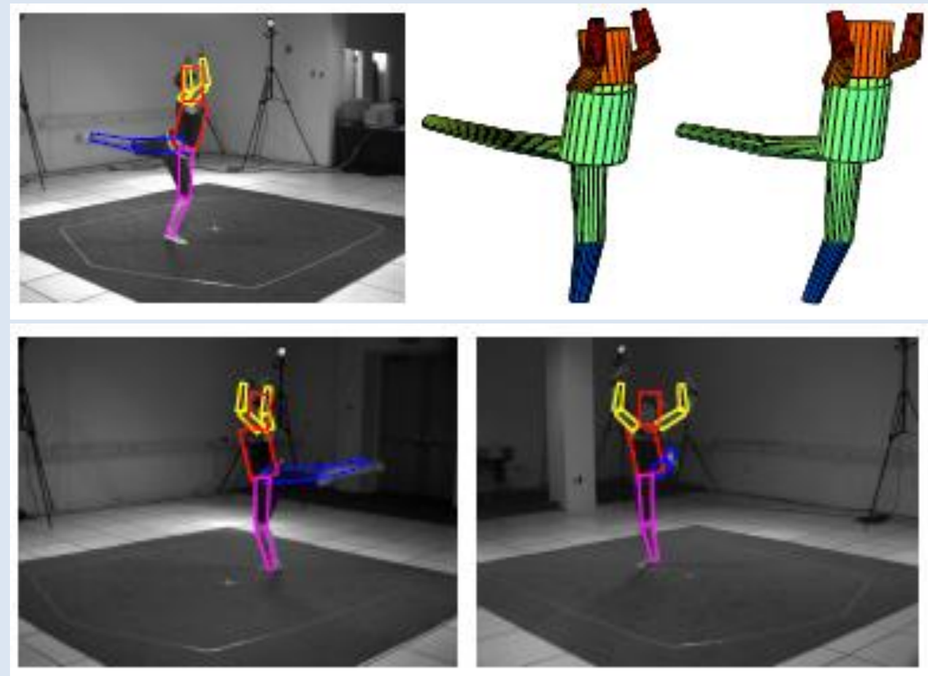




Target Tracking



Radar-based tracking of multiple targets

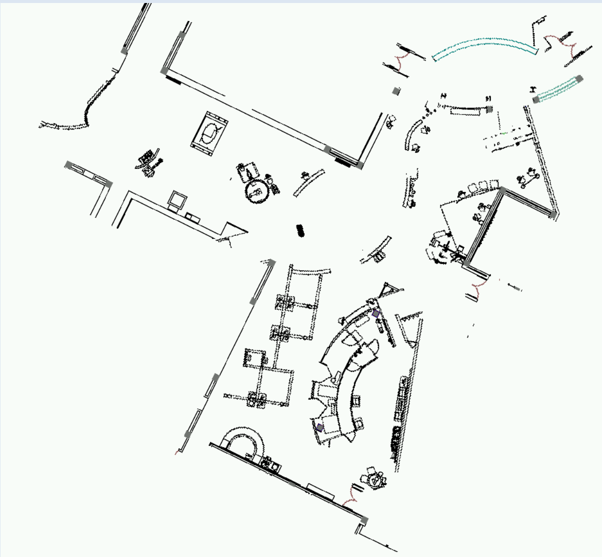


Visual tracking of articulated objects

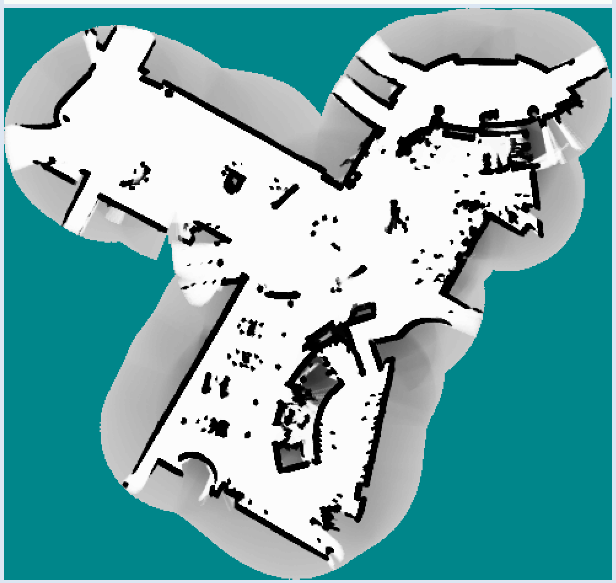
- Estimate motion of targets in 3D world from indirect, potentially noisy measurements

Robot Navigation: *SLAM*

Simultaneous Localization and Mapping



CAD Map

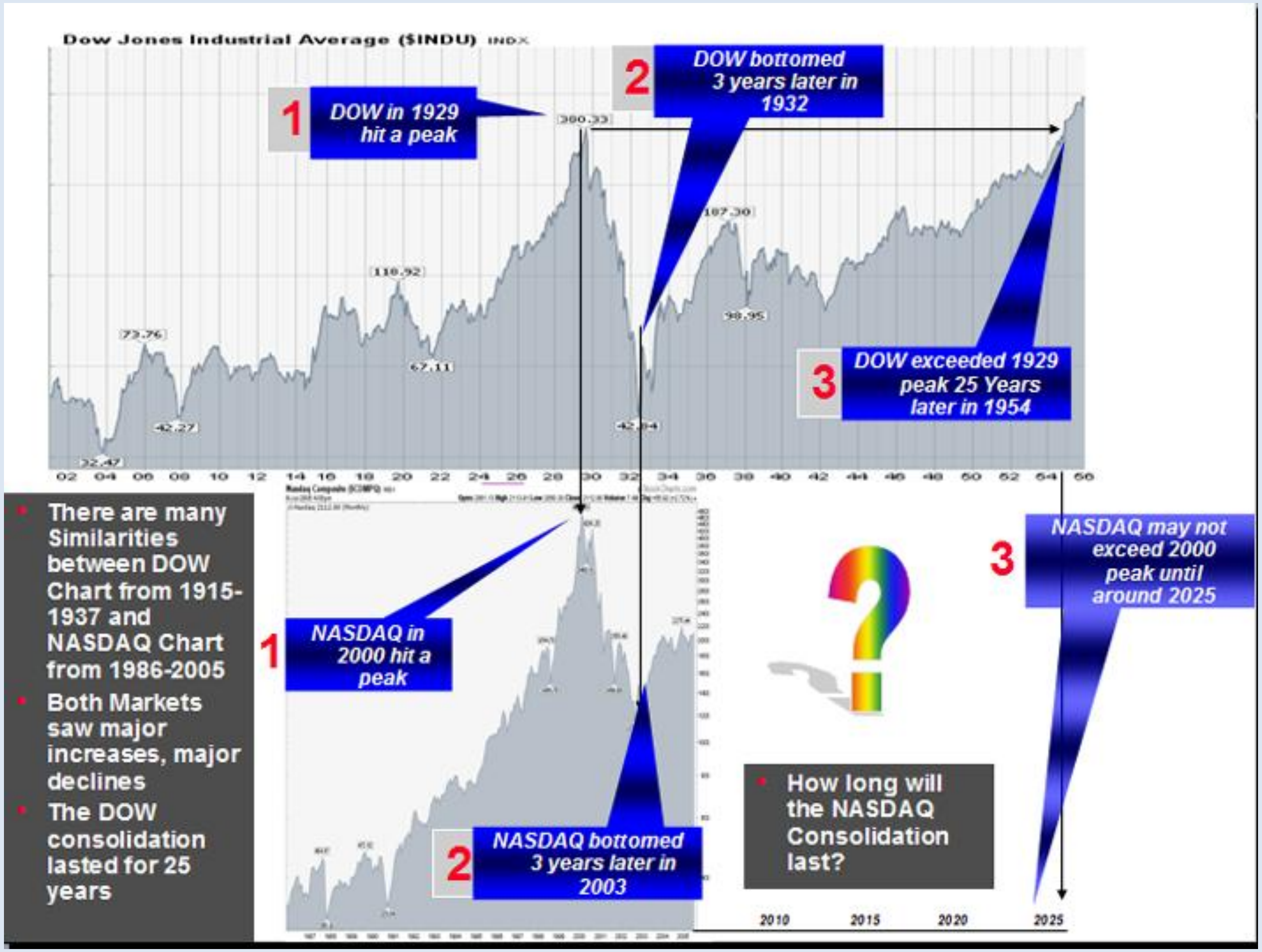


Estimated Map

Landmark SLAM

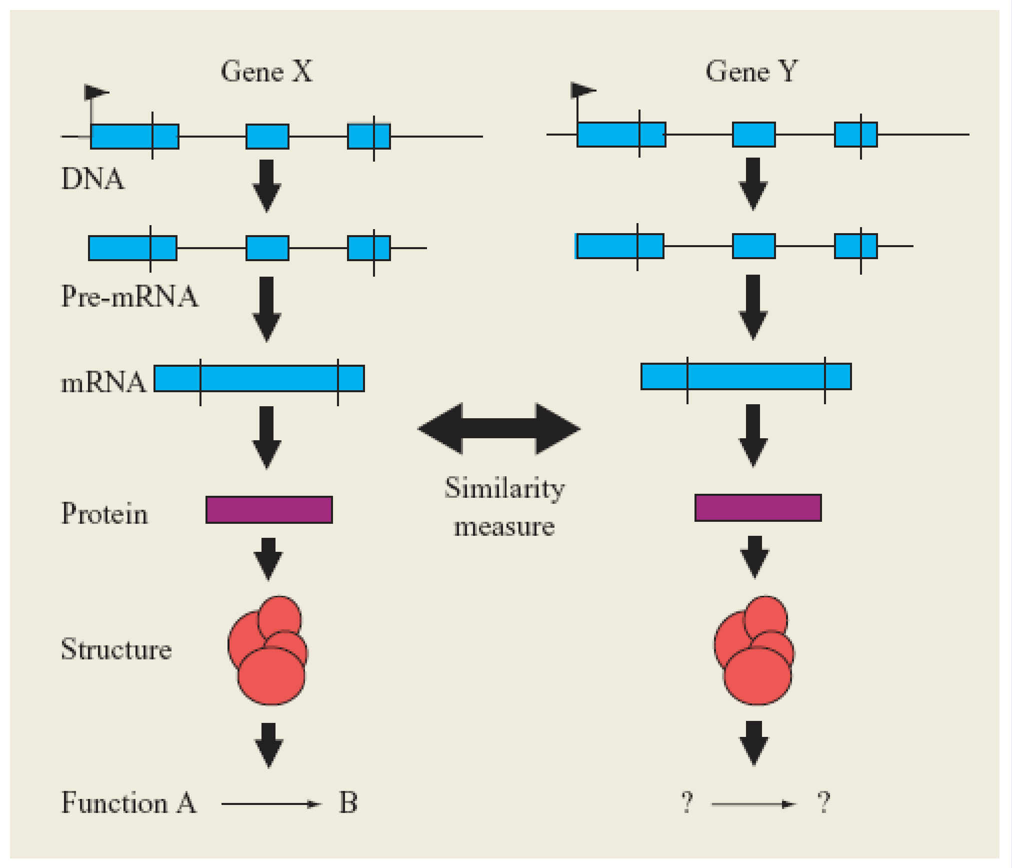
- As robot moves, estimate its pose & world geometry

Financial Forecasting



- Predict future market behavior from historical data, news reports, expert opinions, …

Biological Sequence Analysis

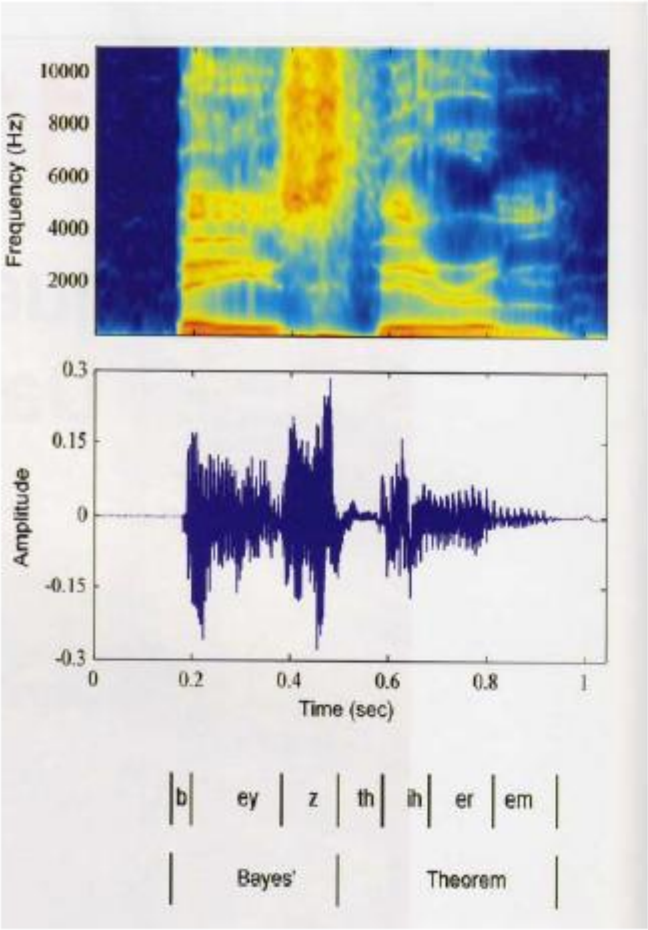


Applications

- Classification of biological sequences
- Motif discovery in biosequences

- Protein or DNA sequences (sequences of characters from a discrete alphabet)

Model Assuming Independence



- Simplest model:
  - Treat as independent
  - Graph without links

States are independent

$$p(x_1, x_2, \ldots, x_T) = p(x_1)p(x_2)\cdots p(x_T)$$

Markov Chains

1st order Markov Chains

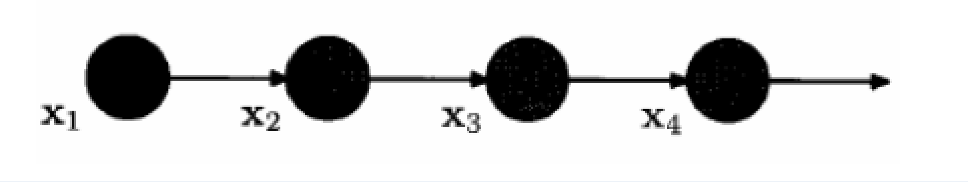
- ***Markov property***: Next state depends only on previous:

$$p(x_t, | x_1, \ldots, x_{t-1}) = p(x_t | x_{t-1})$$

- Joint distribution for a sequence of T states:

$$p(x_1, x_2, \ldots, x_T) = p(x_1) \prod_{t=2}^{T} p(x_t | x_{t-1})$$

- Chain of observations:



Markov Chains

- Elements of a Markov Chain with K states

$$x_t \in \{s_1, s_2, \ldots .s_K\}$$

Markov Chains

- Elements of a Markov Chain with K states

$$x_t \in \{s_1, s_2, \ldots . s_K\}$$

- **initial probabilities**

$$\pi_j = P(x_1 = s_j) \qquad \sum_{j=1}^{K} \pi_j = 1$$

Markov Chains

- Elements of a Markov Chain with K states

$$x_t \in \{s_1, s_2, \ldots . s_K\}$$

- **initial probabilities**

$$\pi_j = P(x_1 = s_j) \qquad \sum_{j=1}^{K} \pi_j = 1$$

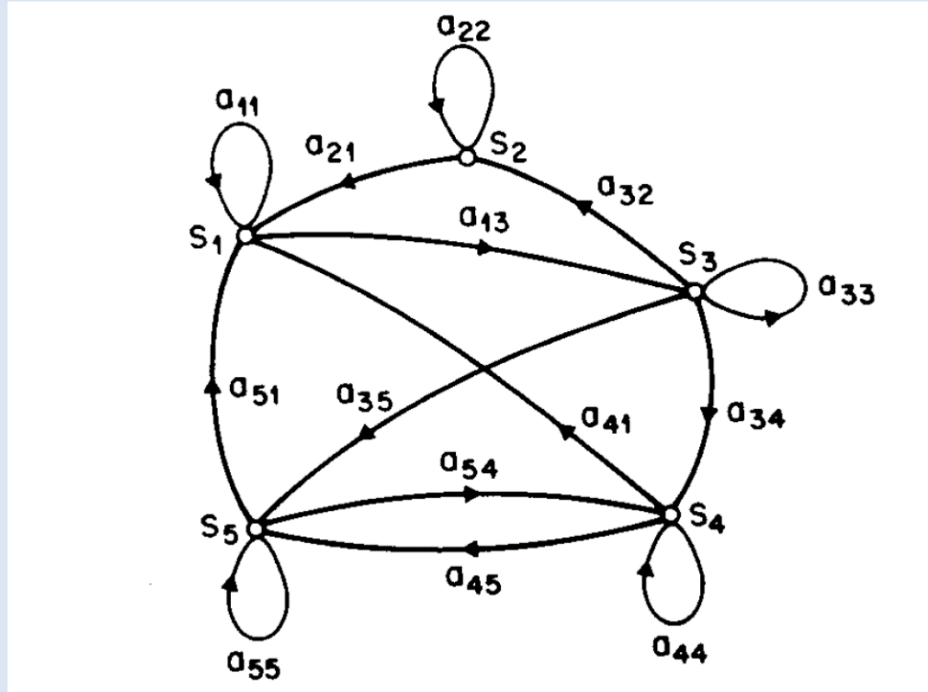
- **transition probabilities**

$$A_{jk} = P(x_{t+1} = s_k \mid x_t = s_j) \qquad \sum_{k=1}^{K} A_{jk} = 1 \quad \forall j$$

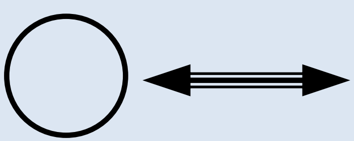
$$\underset{K \times K}{A = [A_{jk}]} \quad j,k = 1, \ldots, K$$

transition matrix

- Markov Chain as **Graphical Model**

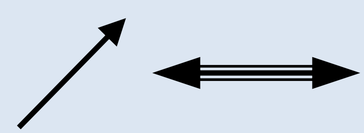


- **Directed Graph (DAG)** with K nodes equal to states and edges with weights equal to transition probabilities.



state values at particular times

nodes

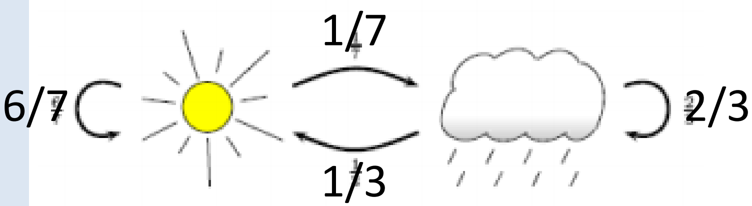


Markov properties

edges

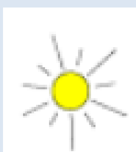
Example (I) of Markov Model

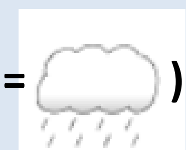
The model, i.e. $p(\mathbf{x}_n \mid \mathbf{x}_{n-1})$:



A sequence of observations:

$$x = \{H, H, R, H, H\}$$

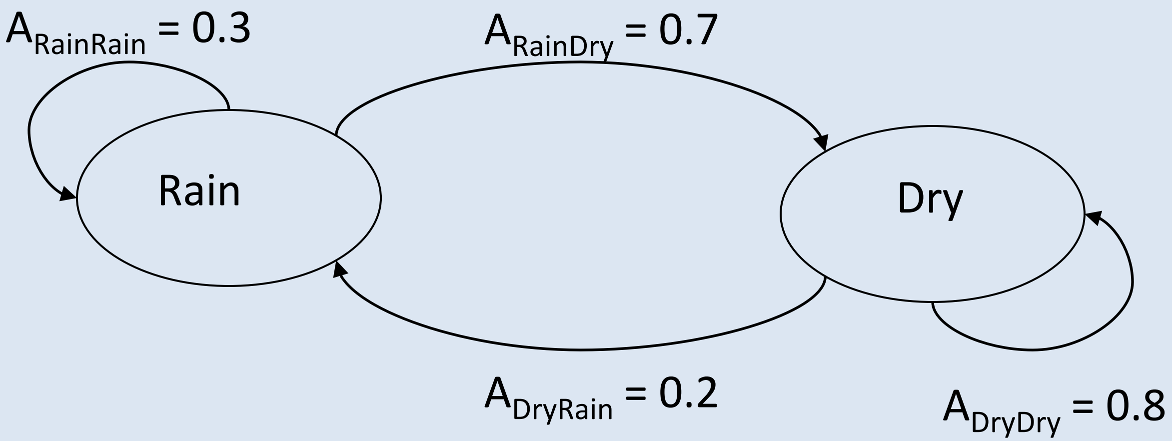
P(x$_1$ =  **) = 0.7**

P(x$_1$ =  **) = 0.3**

p(x) = P(H) P(H|H) P(R|H) P(H|R) P(H|H) =

*= 0.7*6/7*1/7*1/3*6/7*

Example (II) of Markov Model



- **2 states:** ***s_1 = 'Rain'*** και ***s_2 = 'Dry'***
- **Transition Probabilities:** P('Rain' | 'Rain')=0.3 ,

P('Dry' | 'Rain')=0.7 , P('Rain' | 'Dry')=0.2, P('Dry' | 'Dry')=0.8

- **Initial Probabilities:** P('Rain')= π_{Rain} = 0.4 , P('Dry')=π_{Dry} =0.6 .

Probability of a sequence

- Using the Markovian property:

$$P(x = \{x_1, x_2, \ldots, x_T\}) = P(x_1)P(x_2 \mid x_1) \cdots P(x_T \mid x_{T-1})$$

- Example: *X={'Dry','Dry','Rain',Rain'}*

P({'Dry','Dry','Rain',Rain'}) =

= P('Dry') P('Dry' | 'Dry') P('Rain' | 'Dry') P('Rain' | 'Rain') =

= $\pi_{Dry}\ A_{DryDry}\ A_{DryRain}\ A_{RainRain}$ = 0.6*0.8*0.2*0.3=288x10^{-4}

Estimating the parameters of a Markov Chain

- Input set of N sequences $X = \left(X_1, X_2, \ldots, X_N\right)$

  where $X_i = \left\{x_{i1}, x_{i2}, \ldots, x_{iT_i}\right\}$ and $x_{it} \in \left\{s_1, s_2, \ldots . s_K\right\}$

- **Maximum Likelihood** (**ML**) estimators of a MC:

$$I(x,s) = \begin{cases} 1 & x = s \\ 0 & x \neq s \end{cases}$$

$$\hat{A}_{jk} = \frac{\sum_{i=1}^{N}\sum_{t=1}^{T_i-1} I\left(x_{it}, s_j\right) I\left(x_{i,t+1}, s_k\right)}{\sum_{i=1}^{N}\sum_{t=1}^{T_i-1} I\left(x_{it}, s_j\right)} = \frac{n_{jk}}{n_j} = \frac{\text{obs. frequency}\left(s_j \rightarrow s_k\right)}{\text{obs. visit}\left(s_j \rightarrow \#\right)}$$

$$\hat{\pi}_j = \frac{\sum_{i=1}^{N} I\left(x_{i1}, s_j\right)}{N}$$

Relative frequency of using state s_j as initial state

Stationary distribution & Reversibility condition

- Define: $p_{jk}(n) = P(x_{t+n} = s_k \mid x_t = s_j)$ $\quad (p_{jk}(1) = A_{jk})$

$$p_k(n) = P(x_n = s_k) = \sum_j P(x_n = s_k \mid x_{n-1} = s_j) P(x_{n-1} = s_j) = \sum_j A_{jk} p_j(n-1) \Rightarrow$$

$$p(n) = Ap(n-1) \Rightarrow \qquad p(n) = A \cdots A\pi = A^n \pi$$

- **As n→∞ then we have the stationary distribution:**

$$\lim_{n\to\infty} p(n) = \varphi$$

it holds:

$$p(n) = Ap(n-1) \overset{n\to\infty}{\Rightarrow} \varphi = A\varphi$$

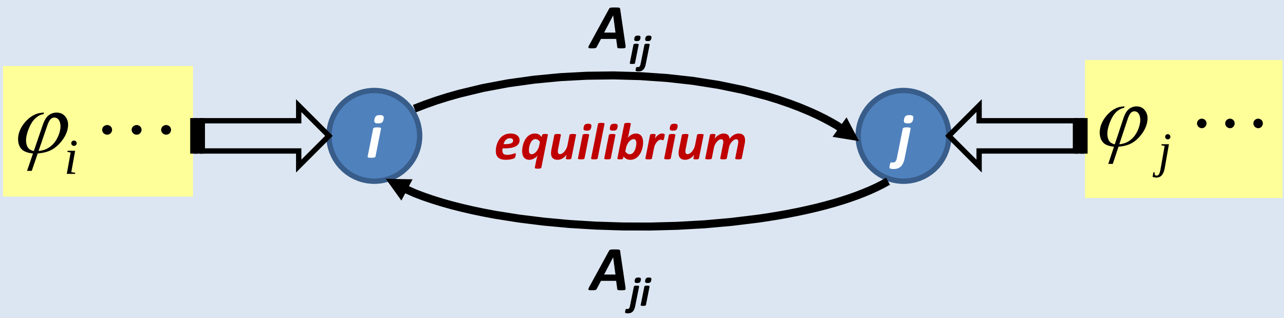
Stationary distribution & Reversibility condition

- **The reversibility condition states:**

A Markov Cain with stationary distribution φ is reversible if:

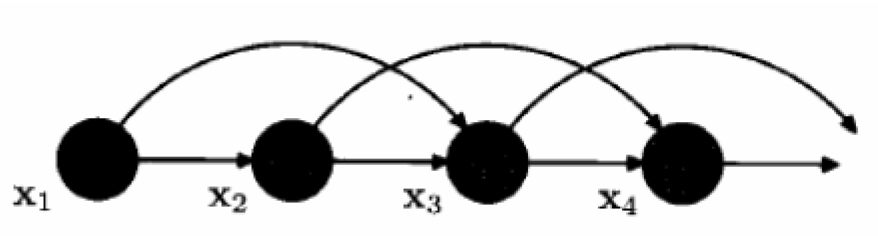
$$\varphi_i A_{ij} = \varphi_j A_{ji}$$

for any two states i, j .



MC of 2nd order

- State $\boldsymbol{x_n}$ depends on two previous states $\boldsymbol{x_{n-1}}$, $\boldsymbol{x_{n-2}}$

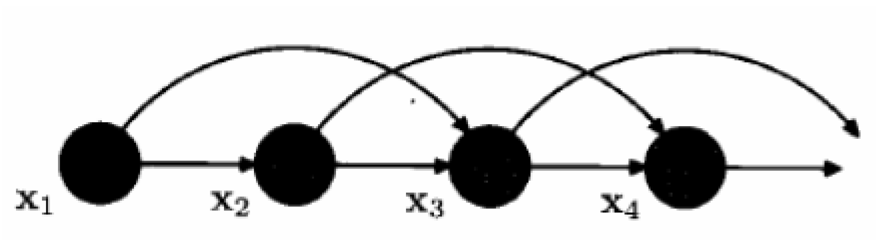


$$p(x_1,..x_N) = p(x_1)\,p(x_2 \mid x_1)\prod_{n=1}^{N} p(x_n \mid x_{n-1}, x_{n-2})$$

- Equivalent to a 1st order MC (?)

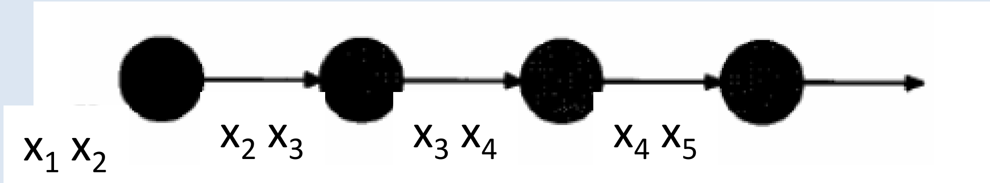
MC of 2nd order

- State x_n depends on two previous states x_{n-1} , x_{n-2}



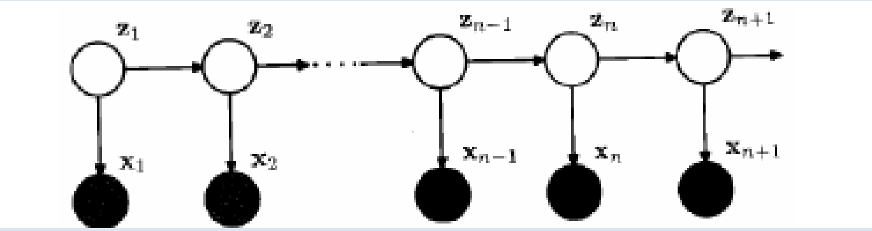
$$p(x_1,\ldots x_N) = p(x_1)\,p(x_2 \mid x_1)\prod_{n=1}^{N} p(x_n \mid x_{n-1}, x_{n-2})$$

- Equivalent to a 1st order MC (?)



Hidden Markov Models - HMMs

- Introduce the notion of **hidden states** (or hidden variables) that describe the **graphical model** that generates the data
- Hidden states are organized to be on a **Markovian grid topology**



- Every hidden state has its own distribution.

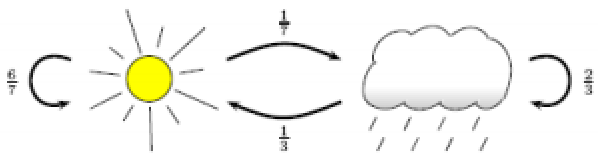
Definition:

A Hidden Markov Model (HMM) is a sequence of random variables whose distribution depends only on the (hidden) state of an associated Markov chain.

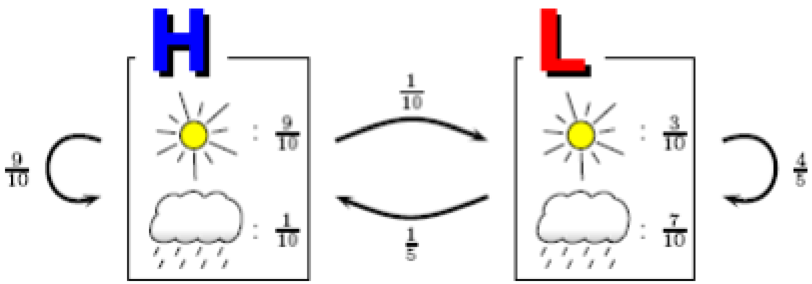
Hidden Markov Models - HMMs

- States are **latent variables**

Markov Model

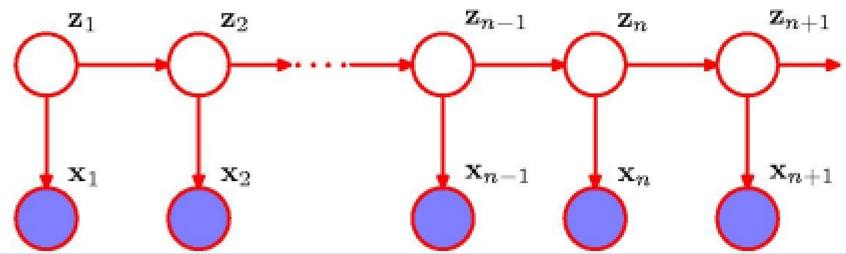


Hidden Markov Model



Latent values

Observations



Hidden Markov Models - HMMs

- States are hidden
- For every **observation (sequence)** there is a **hidden sequence of states**

$$\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$$

$$\mathbf{z} = \{z_1, z_2, \ldots, z_T\}$$

where $z_t \in \{1, \ldots, K\}$ **assuming discrete states**

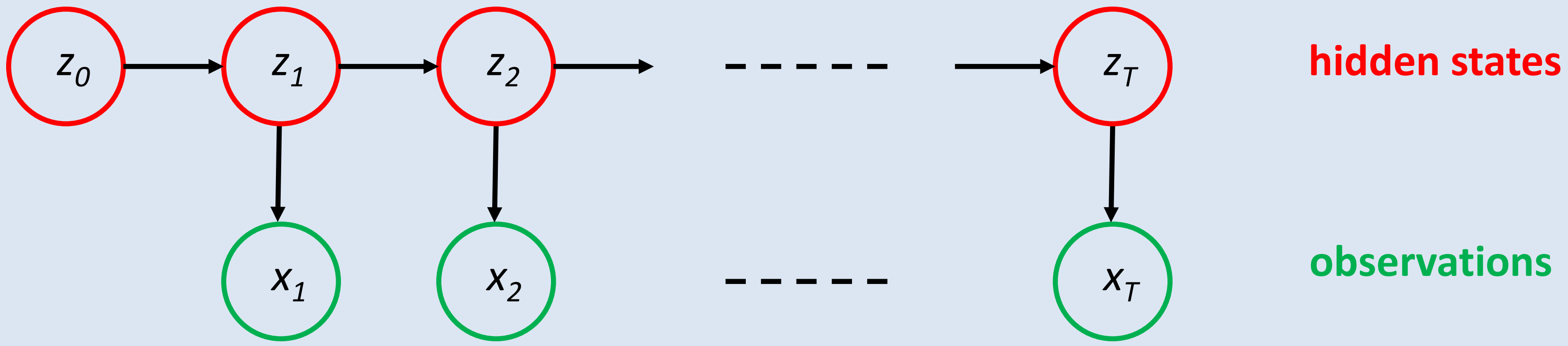
or

$$z_t = (z_{t1}, z_{t1}, \ldots, z_{tK})$$

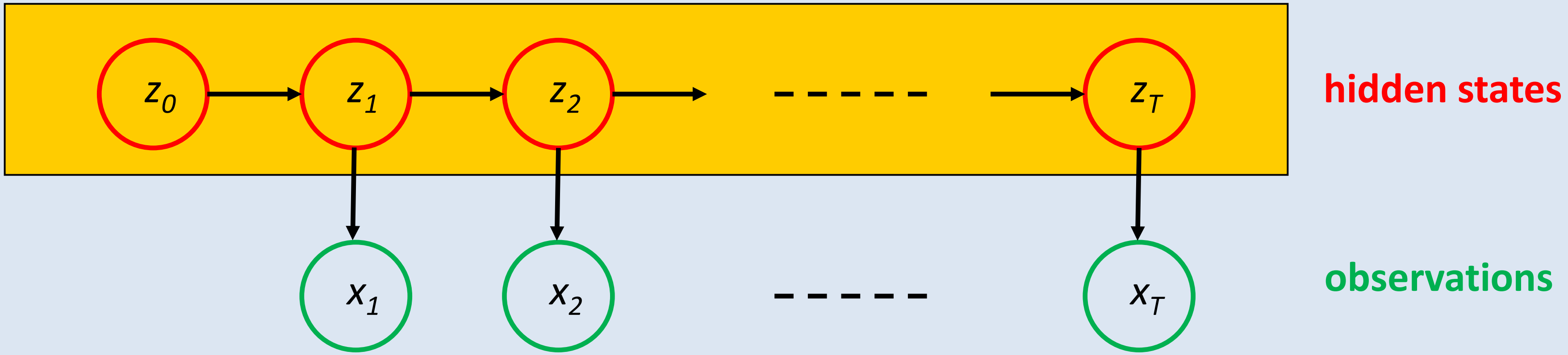
$$z_{tj} = \begin{cases} 1 & use\ state\ s_j\ at\ moment\ t \\ 0 & otherwise \end{cases}$$

binary vector

Hidden Markov Models - HMMs



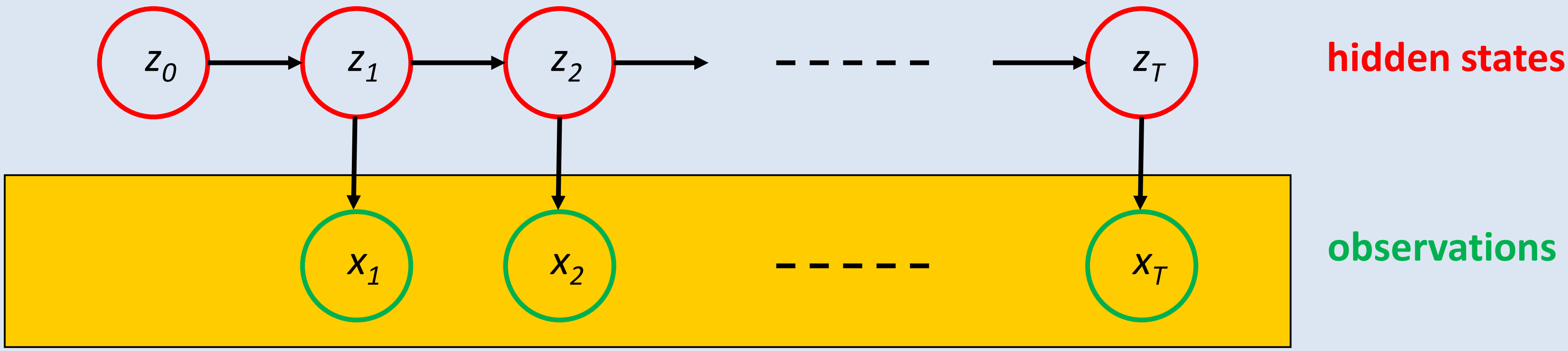
Hidden Markov Models - HMMs



Hidden states

have the Markovian property: Previous state dependence

Hidden Markov Models - HMMs



Observation

depends only on the (hidden) state that is visited at each time step

Parameters of an HMM

- **initial state probabilities**

$$\pi_j = P(z_1 = s_j) = P(z_{1j} = 1) \qquad \sum_{j=1}^{K} \pi_j = 1$$

π *= (π₁ , π₂ , … ,π_K)* : **vector of probabilities**

$$p(z_1) = \prod_{j=1}^{K} (\pi_j)^{z_{1j}} \qquad z_{1j} = \begin{cases} 1 & initial\, state\, s_j \\ 0 & otherwise \end{cases}$$

- ***transition probabilities***

$$A_{jk} = P\left(z_t = s_k \mid z_{t-1} = s_j\right) = P\left(z_{tk} = 1 \mid z_{t-1,k} = 1\right)$$

Transition array *A = [A$_{jk}$]* $\sum_{k=1}^{K} A_{jk} = 1 \quad \forall j$

$$p\left(z_t \mid z_{t-1}\right) = \prod_{j=1}^{K} \prod_{k=1}^{K} \left(A_{jk}\right)^{z_{t-1,j} z_{t,k}}$$

$$z_{tj} = \begin{cases} 1 & state\, s_j \text{ at moment } t \\ 0 & otherwise \end{cases}$$

- ***emission probabilities***

Every hidden state j has its own distribution with a density function **p(x | θ$_j$)** with parameters **θ$_j$**

$$p(x_t \mid z_t) = \prod_{j=1}^{K} \left(p(x \mid \theta_j) \right)^{z_{tj}}$$

It depends on the type of data, e.g.

- **Gaussian (continuous)** $p(x \mid \theta_j) = N(\mu_j, \Sigma_j)$
- **Multinomial (discrete)** $p(x \mid \theta_j) = Mul(\theta_j) = \prod_{m=1}^{M} \left(\theta_m^{(j)} \right)^{I(x,m)}$
- ……

$$\Theta = \{\theta_j\}_{j=1}^{K}$$ set of parameters of K distributions

- In total, the **parameters of an HMM** are:

$$\lambda = \{\pi, A, \Theta\}$$

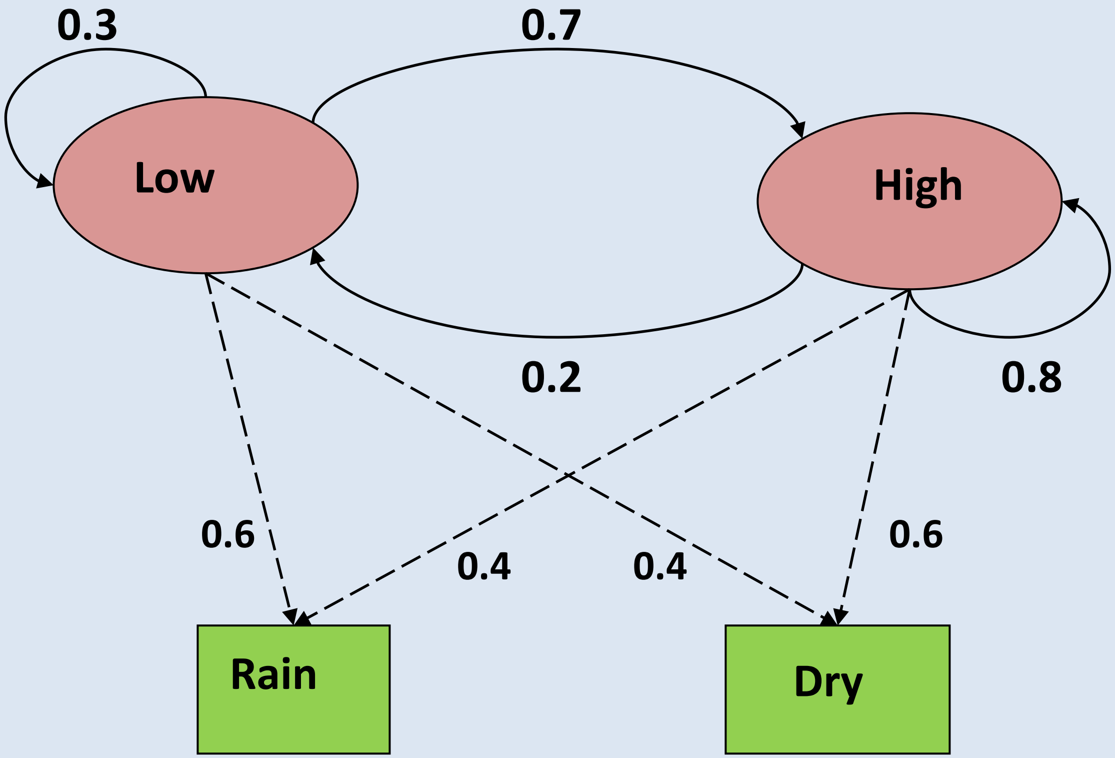
- Joint-distribution of (x,z)

$$p(x, z \mid \lambda) = p(z \mid \lambda)\, p(x \mid z, \lambda) =$$
$$= \left(p(z_1) \prod_{t=2}^{T} p(z_t \mid z_{t-1}) \right) \left(\prod_{t=1}^{T} p(x_t \mid z_t) \right)$$

Markovian property

Independence among observations

An example



- **2 states** : ‘Low’ and ‘High’ (atmospheric pressure)
- **Observations** : { ‘Rain’ , ‘Dry’ }
- **Transition probabilities:**
P(‘Low’|‘Low’)=0.3 , P(‘High’|‘Low’)=0.7 ,
P(‘Low’|‘High’)=0.2, P(‘High’|‘High’)=0.8
- **Emission probabilities:**
P(‘Rain’|‘Low’)=0.6 , P(‘Dry’|‘Low’)=0.4 ,
P(‘Rain’|‘High’)=0.4 , P(‘Dry’|‘High’)=0.6
- **Initial probabilities:**
P(‘Low’)=0.4 , P(‘High’)=0.6

Probability computation

P({'Dry','Rain'}) =

Probability computation

- 4 possible sequences (paths) of states:

P({'Dry','Rain'}) =
P({'Dry','Rain'} , {'Low','Low'}) +
P({'Dry','Rain'} , {'Low','High'}) +
P({'Dry','Rain'} , {'High','Low'}) +
P({'Dry','Rain'} , {'High','High'})

όπου (π.χ.):
P({'Dry','Rain'} , {'Low','Low'})=
P({'Dry','Rain'} | {'Low','Low'}) P({'Low','Low'}) =
P('Dry'|'Low')P('Rain'|'Low') P('Low')P('Low'|'Low')
= 0.4*0.4*0.6*0.4*0.3

Problems of HMMs

1. Likelihood calculation
2. Most probable path
3. Parameter estimation
4. Making prediction

[1]. Likelihood calculation

- **Likelihood** $p(x \mid \lambda)$
- **Marginal** to all possible paths

$$p(x \mid \lambda) = \sum_{z} p(x, z \mid \lambda) = \sum_{z} p(z \mid \lambda) p(x \mid z, \lambda) =$$

$$= \sum_{z=(z_1, z_2, \ldots, z_T)} \left\{ p(z_1) \prod_{t=1}^{T-1} p(z_{t+1} \mid z_t) \prod_{t=1}^{T} p(x_t \mid z_t) \right\}$$

- **There are K^T different paths (huge complexity)**

Forward / Backward

- Dynamic programming algorithm
- Define **forward** variable

$$a(z_t) = p(x_1, \ldots, x_t, z_t)$$

Forward / Backward

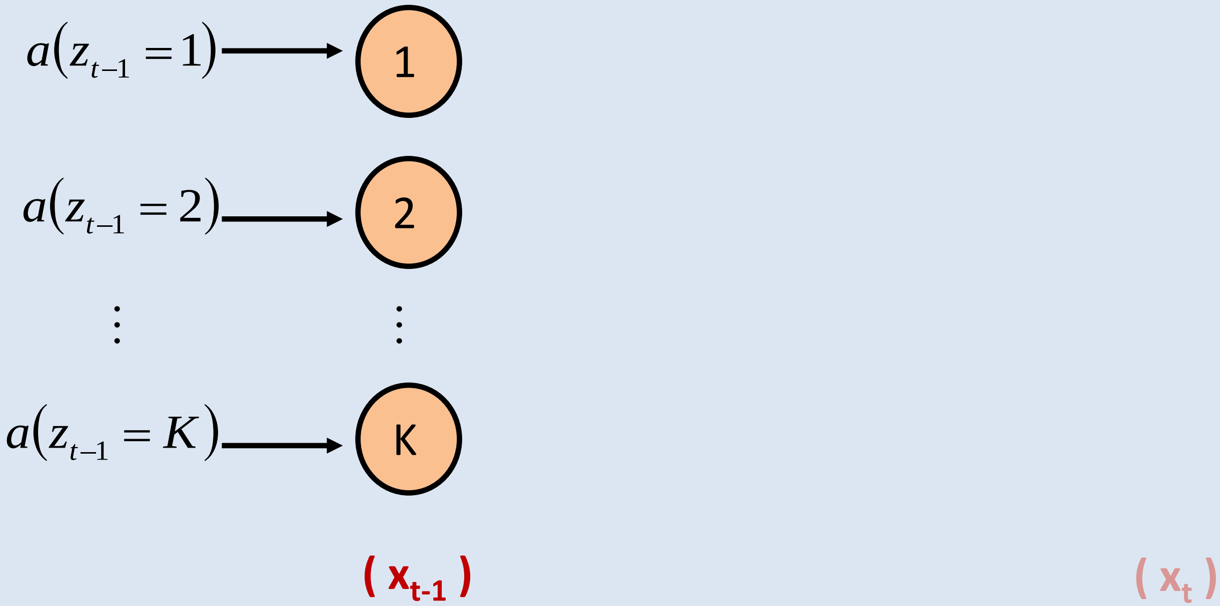
- Dynamic programming algorithm
- Define **forward** variable:

$$a(z_t) = p(x_1,\ldots,x_t,z_t)$$

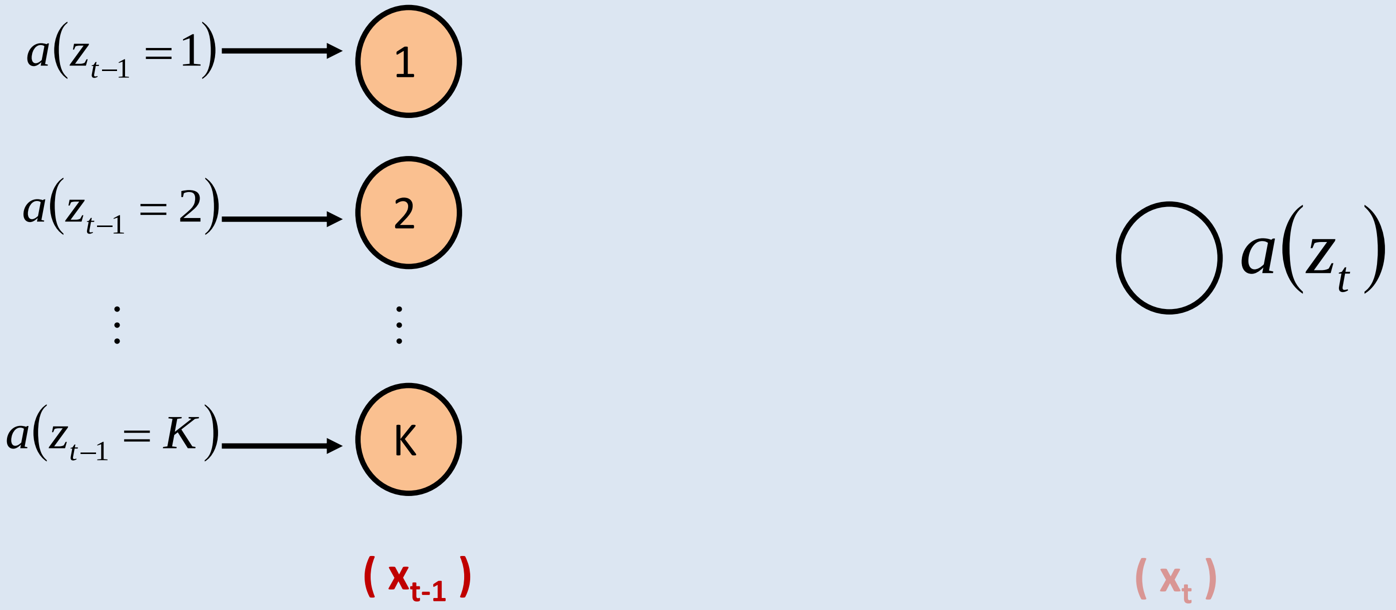
- **Initially** $a(z_1) = p(x_1,z_1) = p(z_1)p(x_1 \mid z_1)$
- **Recursively**

$$a(z_t) = p(x_1,\ldots,x_{t-1},x_t,z_t) = p(x_t \mid z_t)p(x_1,\ldots,x_{t-1},z_t) =$$
$$= p(x_t \mid z_t)\sum_{z_{t-1}} p(x_1,\ldots,x_{t-1},z_{t-1},z_t) = p(x_t \mid z_t)\sum_{z_{t-1}} a(z_{t-1})p(z_t \mid z_{t-1})$$

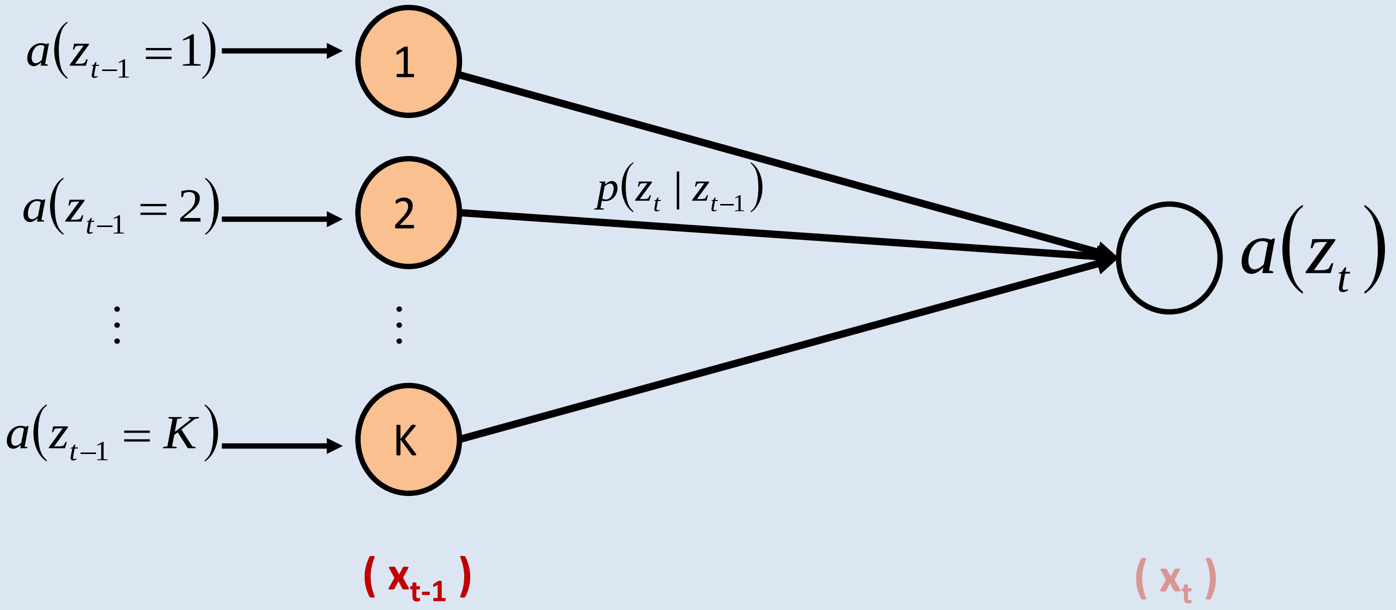
$$a(z_{t-1}) = p(x_1, \ldots, x_{t-1}, z_{t-1})$$



$$a(z_t) = p(x_1, \ldots, x_{t-1}, x_t, z_t)$$

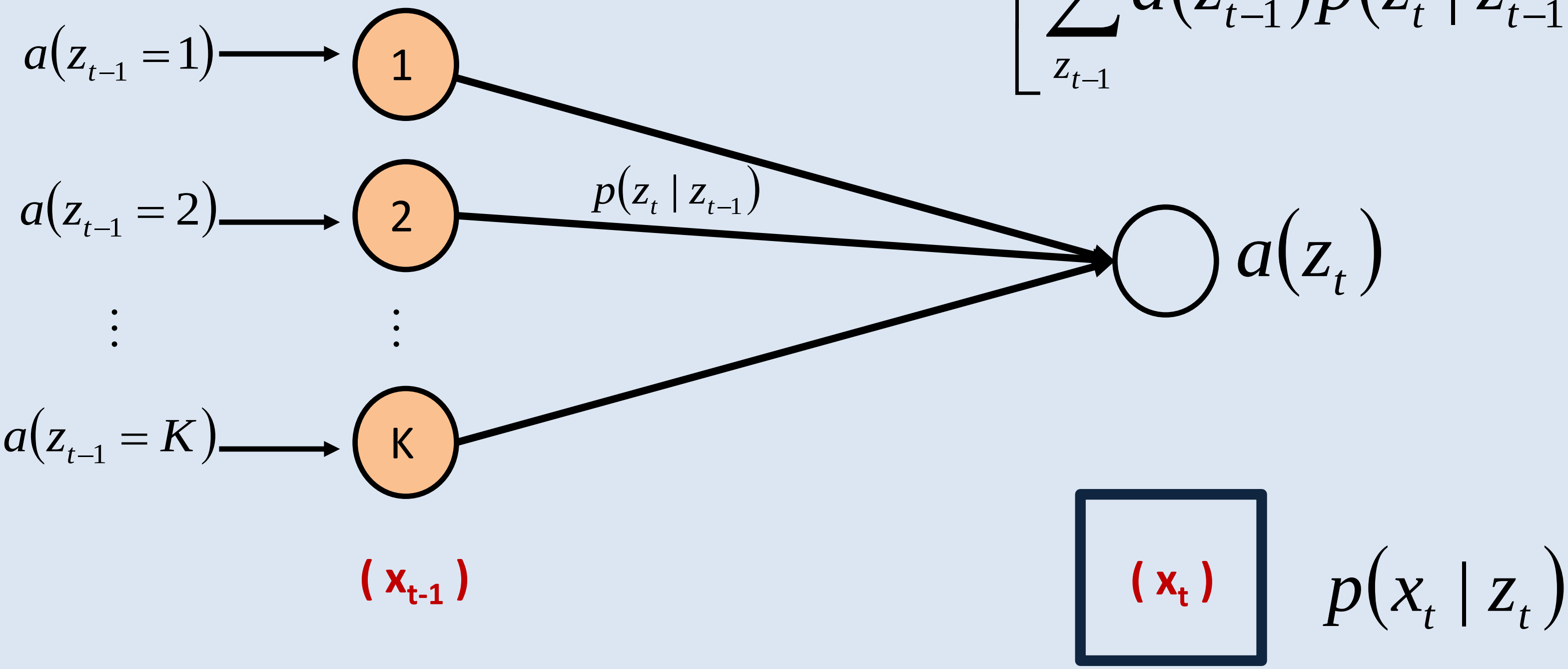


$$a(z_t) = p(x_1, \ldots, x_{t-1}, x_t, z_t) = \sum_{z_{t-1}} p(x_1, \ldots, x_{t-1}, x_t, z_{t-1}, z_t)$$

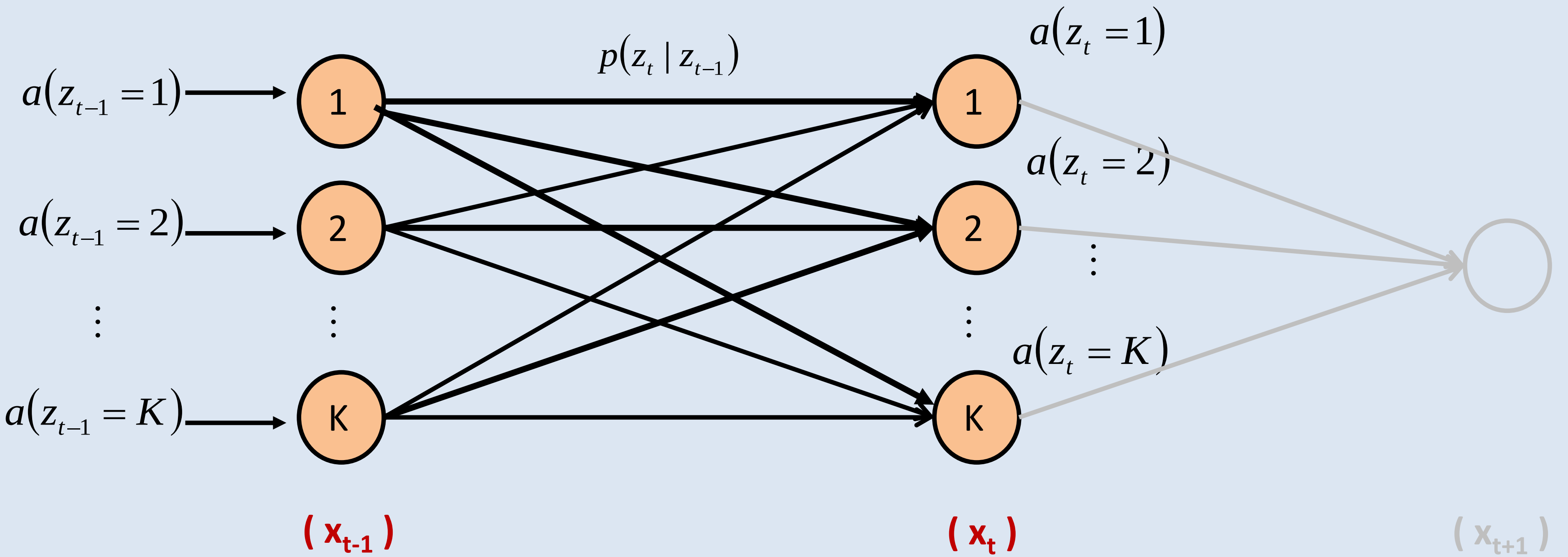


$$a(z_t) = p(x_1,\ldots,x_{t-1},x_t,z_t) = \sum_{z_{t-1}} p(x_1,\ldots,x_{t-1},x_t,z_{t-1},z_t)$$

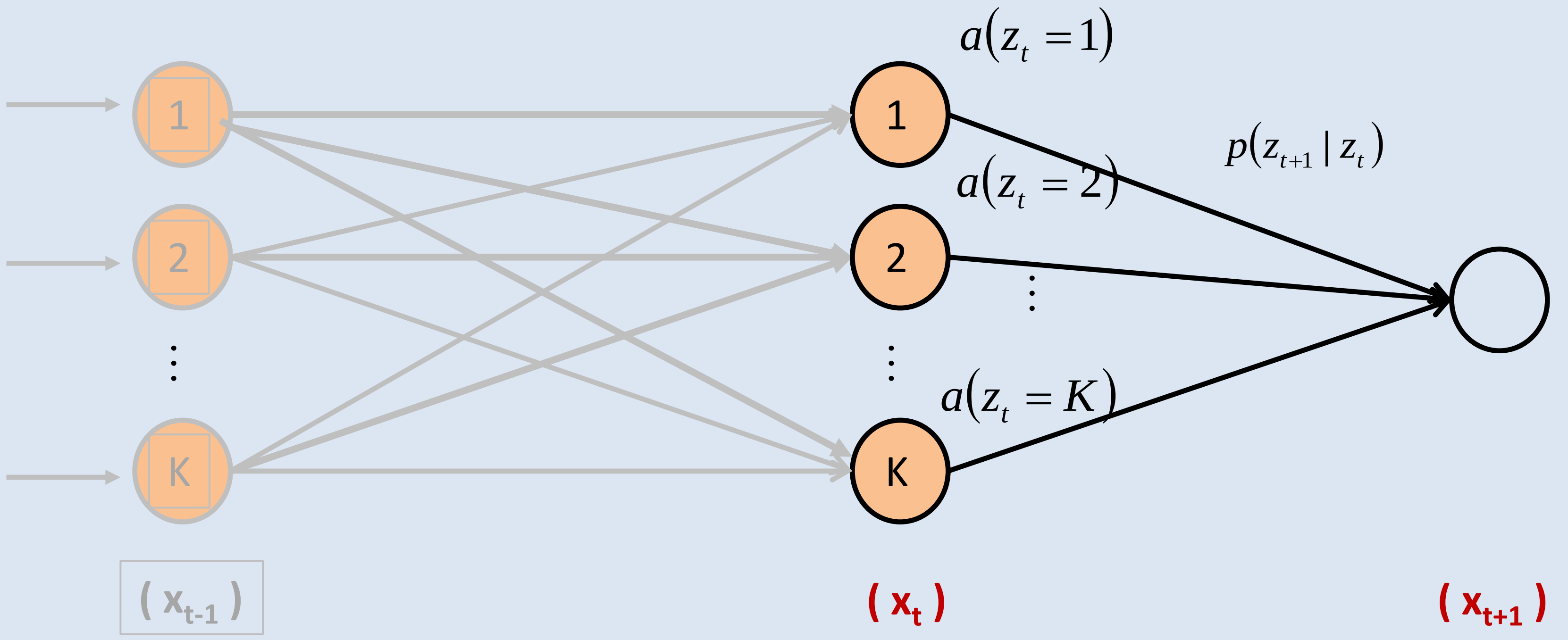
$$= \left[\sum_{z_{t-1}} a(z_{t-1})\,p(z_t \mid z_{t-1})\right] p(x_t \mid z_t)$$



$$a(z_t) = p(x_1, \ldots, x_{t-1}, x_t, z_t) = \left[\sum_{z_{t-1}} a(z_{t-1}) p(z_t \mid z_{t-1})\right] p(x_t \mid z_t)$$



$$a(z_{t+1}) = p(x_1, \ldots, x_t, x_{t+1}, z_t) = \left[\sum_{z_t} a(z_t) p(z_{t+1} \mid z_t) \right] p(x_{t+1} \mid z_{t+1})$$



Likelihood calculation:

$$p(\mathbf{x} \mid \lambda) = \sum_{z_T} p(\mathbf{x}, z_T \mid \lambda) = \sum_{z_T} a(z_T)$$

- **Going backward …..**

- Define **backward μεταβλητή**

$$\beta(z_t) = p(x_{t+1}, \ldots, x_T \mid z_t)$$

- **Initially** $\beta(z_T) = 1$

- **Recursively**

- **Going backward .....**
- Define **backward μεταβλητή**

$$\beta(z_t) = p(x_{t+1}, \ldots, x_T \mid z_t)$$

- **Initially** $\beta(z_T) = 1$

$$\beta(z_t) = \sum_{z_{t+1}} p(x_{t+1}, \ldots, x_T, z_{t+1}, \mid z_t) =$$

- **Recursively**

$$= \sum_{z_{t+1}} p(x_{t+2}, \ldots, x_T \mid z_{t+1}) p(x_{t+1} \mid z_{t+1}) p(z_{t+1} \mid z_t) =$$

$$= \sum_{z_{t+1}} \beta(z_{t+1}) p(x_{t+1} \mid z_{t+1}) p(z_{t+1} \mid z_t)$$

Likelihood calculation:

$$p(\mathbf{x} \mid \lambda) = \sum_{z_1} p(\mathbf{x}, z_1 \mid \lambda) =$$

$$= \sum_{z_1} p(\mathbf{x} \mid z_1, \lambda) p(z_1 \mid \lambda) = \sum_{z_1} \beta(z_1) p(z_1)$$

- Likelihood of a sequence **(I)**

$$p(\mathbf{x} \mid \lambda) = \sum_{z_T} p(\mathbf{x}, z_T \mid \lambda) = \sum_{z_T} a(z_T)$$

- Likelihood of a sequence **(I)**

$$p(\mathbf{x} \mid \lambda) = \sum_{z_T} p(\mathbf{x}, z_T \mid \lambda) = \sum_{z_T} a(z_T)$$

- Likelihood of a sequence **(II)**

$$p(\mathbf{x} \mid \lambda) = \sum_{z_1} p(\mathbf{x}, z_1 \mid \lambda) = \sum_{z_1} \beta(z_1) p(z_1)$$

- Likelihood of a sequence **(I)**

$$p(\mathbf{x} \mid \lambda) = \sum_{z_T} p(\mathbf{x}, z_T \mid \lambda) = \sum_{z_T} a(z_T)$$

- Likelihood of a sequence **(II)**

$$p(\mathbf{x} \mid \lambda) = \sum_{z_1} p(\mathbf{x}, z_1 \mid \lambda) = \sum_{z_1} \beta(z_1) p(z_1)$$

- Likelihood of a sequence **(III)**

$$p(\mathbf{x} \mid \lambda) = \sum_{z_t} p(\mathbf{x}, z_t \mid \lambda) = \sum_{z_t} \alpha(z_t) \beta(z_t)$$

[2]. Most probable path

Viterbi Algorithm

- **Define variable**

$$\delta(z_t) = \max_{z_1 \to z_{t-1}} p(x_1, \ldots, x_t, z_t)$$

Max probability among all paths that visit state z_t and produce sub-sequence $x_1 \to x_t$

[2]. Most probable path

Viterbi Algorithm

- **Define variable**

$$\delta(z_t) = \max_{z_1 \rightarrow z_{t-1}} p(x_1, \ldots, x_t, z_t)$$

Max probability among all paths that visit state z_t and produce sub-sequence $x_1 \rightarrow x_t$

- **Initially** $\delta(z_1) = a(z_1)$

- **Recursively**

- **Finally**

- Execute a reverse-time, **backtracking** procedure then picks the maximizing state sequence

[2]. Most probable path

Viterbi Algorithm

- **Define variable**

$$\delta(z_t) = \max_{z_1 \to z_{t-1}} p(x_1, \ldots, x_t, z_t)$$

Max probability among all paths that visit state z_t and produce sub-sequence $x_1 \to x_t$

- **Initially** $\delta(z_1) = a(z_1)$

- **Recursively** $\delta(z_t) = \max_{z_1 \to z_{t-1}} p(x_1, \ldots, x_t, z_t) = p(x_t \mid z_t) \max_{z_1 \to z_{t-1}} \{\delta(z_{t-1}) p(z_t \mid z_{t-1})\}$

- Finally

- Execute a reverse-time, **backtracking** procedure then picks the maximizing state sequence

[2]. Most probable path

Viterbi Algorithm

- **Define variable**

$$\delta(z_t) = \max_{z_1 \to z_{t-1}} p(x_1, \ldots, x_t, z_t)$$

Max probability among all paths that visit state z_t and produce sub-sequence $x_1 \to x_t$

- **Initially** $\delta(z_1) = a(z_1)$

- **Recursively** $\delta(z_t) = \max_{z_1 \to z_{t-1}} p(x_1, \ldots, x_t, z_t) = p(x_t \mid z_t) \max_{z_1 \to z_{t-1}} \{\delta(z_{t-1}) p(z_t \mid z_{t-1})\}$

- **Finally** $z_T^{\;*} = \arg\max_{z_T} \delta(z_T)$ ***Most probable final visited state***

- Execute a reverse-time, **backtracking** procedure then picks the maximizing state sequence

[2]. Most probable path

Viterbi Algorithm

- **Define variable**

$$\delta(z_t) = \max_{z_1 \to z_{t-1}} p(x_1, \ldots, x_t, z_t)$$

Max probability among all paths that visit state z_t and produce sub-sequence $x_1 \to x_t$

- **Initially** $\delta(z_1) = a(z_1)$

- **Recursively** $\delta(z_t) = \max_{z_1 \to z_{t-1}} p(x_1, \ldots, x_t, z_t) = p(x_t \mid z_t) \max_{z_1 \to z_{t-1}} \{\delta(z_{t-1}) p(z_t \mid z_{t-1})\}$

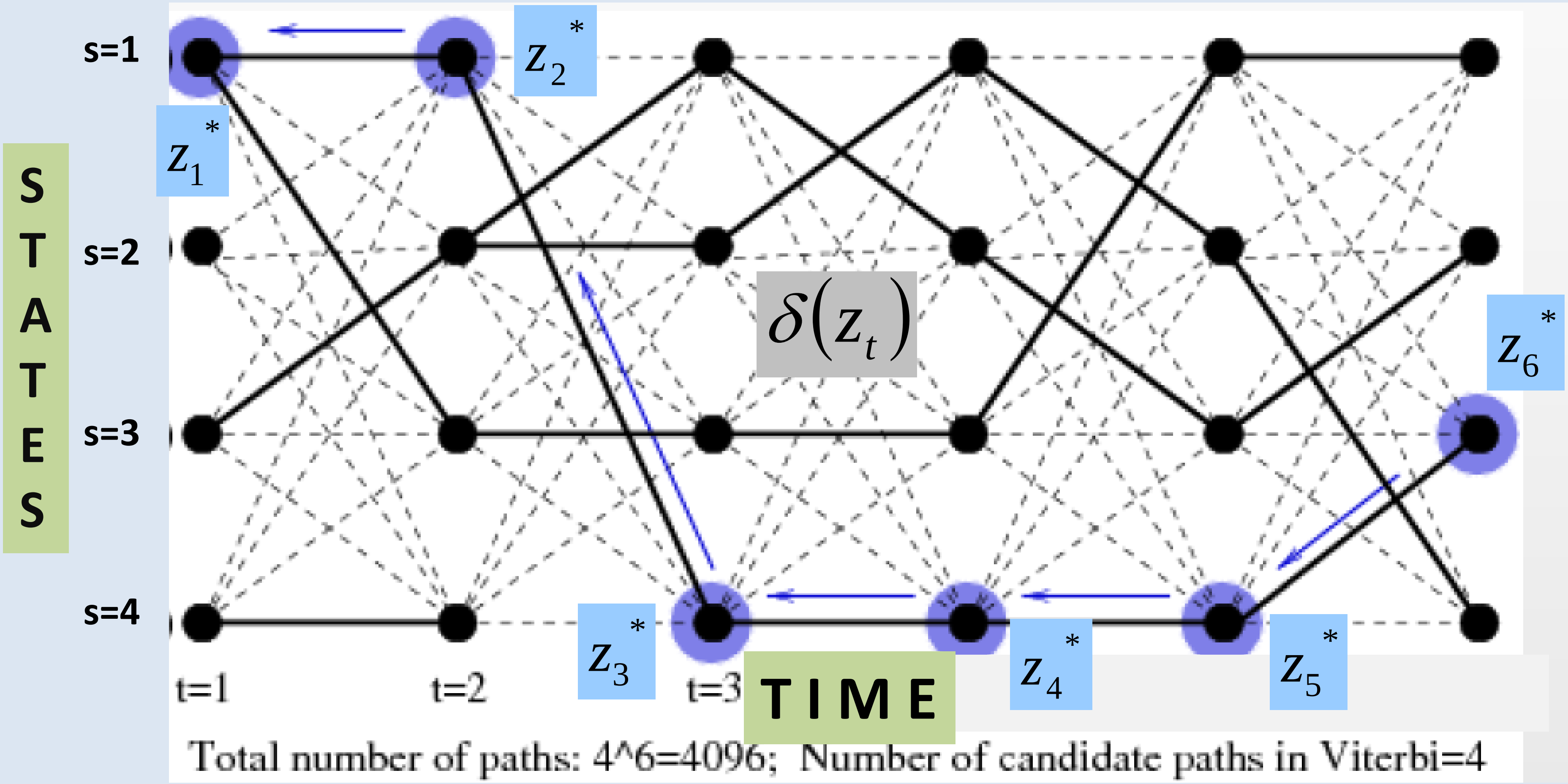
- **Finally** $z_T^* = \arg\max_{z_T} \delta(z_T)$ ***Most probable final visited state***

- Execute a reverse-time, **backtracking** procedure and then picks the maximizing state sequence

An **example** of the Viterbi algorithm

(assume K=4 hidden states – sequence of length T=6)

$$\delta(z_t) = \max_{z_1 \to z_{t-1}} p(x_1,\ldots,x_t,z_t) = p(x_t \mid z_t) \max_{z_1 \to z_{t-1}} \delta(z_{t-1}) p(z_t \mid z_{t-1})$$



[3]. Parameter estimation of an HMM (Training an HMM)

- Parameters of an HMM $\lambda=\{\pi, A, \Theta\}$
- Useful **posterior probabilities**

$$\gamma(z_t) = p(z_t \mid \mathbf{x}) =$$

$$\xi(z_t, z_{t+1}) = p(z_t, z_{t+1} \mid \mathbf{x}) =$$

[3]. Parameter estimation of an HMM (Training an HMM)

- Parameters of an HMM λ={π, A, Θ}

- Useful **posterior probabilities**

$$\gamma(z_t) = p(z_t \mid \mathbf{x}) = \frac{p(\mathbf{x}, z_t)}{p(\mathbf{x})} = \frac{a(z_t)\beta(z_t)}{\sum_{z_t'} a(z_t')\beta(z_t')}$$

$$\xi(z_t, z_{t+1}) = p(z_t, z_{t+1} \mid \mathbf{x}) = \frac{p(x, z_t, z_{t+1})}{p(x)} =$$

$$= \frac{a(z_t)p(z_{t+1} \mid z_t)p(x_{t+1} \mid z_{t+1})\beta(z_{t+1})}{\sum_{z'_t}\sum_{z'_{t+1}} a(z'_t)p(z'_{t+1} \mid z'_t)p(x_{t+1} \mid z'_{t+1})\beta(z'_{t+1})}$$

– **Expectation of the number of visiting (frequency)** state s_k

$$\hat{n}_k = \sum_{t=1}^{T} \gamma(z_t = k) = \sum_{t=1}^{T} p(z_t = k \mid \mathbf{x})$$

– **Expectation of the number of transitions (frequency)** from state s_j to state s_k

$$\hat{n}_{jk} = \sum_{t=1}^{T-1} \xi(z_t = j, z_{t+1} = k) = \sum_{t=1}^{T-1} p(z_t = j, z_{t+1} = k \mid \mathbf{x})$$

[3.1] Baum-Welch algorithm

Update rules for model parameters

$$\pi_k^{(new)} = \ldots$$

Probability of visiting state k at first time, i.e.
(in statistics) relative frequency of visiting state k

$$A_{jk}^{(new)} = \frac{\sum^{T-1} \xi(\ldots}{\sum_{k'=1} \sum_{t=1} \ldots}$$

Probability of transition from state j to state k, i.e.
(in statistics) relative frequency of making j-> k

[3.1] Baum-Welch algorithm

Update rules for model parameters

$$\pi_k^{(new)} = \gamma\left(z_1 = k\right)$$

$$A_{jk}^{(new)} = \frac{\sum_{t=1}^{T-1} \xi\left(z_t = j, z_{t+1} = k\right)}{\sum_{m=1}^{K} \sum_{t=1}^{T-1} \xi\left(z_t = j, z_{t+1} = m\right)} = \frac{\hat{n}_{jk}}{\sum_{m=1}^{K} \hat{n}_{jm}}$$

[3.1] Baum-Welch algorithm

Update rules for model parameters (discrete data)

$$p(x \mid \theta_j) = Mul(\theta_j) = \prod_{m=1}^{M} (\theta_{jm})^{I(x,m)}$$

$$I(x,m) = \begin{cases} 1 & x = m \\ 0 & x \neq m \end{cases}$$

$$\pi_k^{(new)} = \gamma(z_1 = k)$$

$$A_{jk}^{(new)} = \frac{\sum_{t=1}^{T-1} \xi(z_t = j, z_{t+1} = k)}{\sum_{m=1}^{K} \sum_{t=1}^{T-1} \xi(z_t = j, z_{t+1} = m)} = \frac{\hat{n}_{jk}}{\sum_{m=1}^{K} \hat{n}_{jm}}$$

$$\theta_{jm}^{(new)} = \frac{\sum_{t=1}^{T} \gamma(z_t = j) I(x_t, m)}{\sum_{t=1}^{T} \gamma(z_t = j)}$$

[3.1] Baum-Welch algorithm

Update rules for model parameters (continuous-normal data)

$$p(x \mid \theta_j) = N(\mu_j, \Sigma_j)$$

$$\pi_k^{(new)} = \gamma(z_1 = k)$$

$$A_{jk}^{(new)} = \frac{\sum_{t=1}^{T-1} \xi(z_t = j, z_{t+1} = k)}{\sum_{m=1}^{K} \sum_{t=1}^{T-1} \xi(z_t = j, z_{t+1} = m)} = \frac{\hat{n}_{jk}}{\sum_{m=1}^{K} \hat{n}_{jm}}$$

$$\mu_j^{(new)} = \frac{\sum_{t=1}^{T} \gamma(z_t = j) x_n}{\sum_{t=1}^{T} \gamma(z_t = j)}$$

$$\Sigma_k^{(new)} = \frac{\sum_{t=1}^{T} \gamma(z_t = k)\left(x_n - \mu_j^{(new)}\right)\left(x_n - \mu_j^{(new)}\right)^T}{\sum_{t=1}^{T} \gamma(z_t = k)}$$

[3.2] Use EM algorithm

Likelihood function $\lambda = \{\pi, A, \Theta\}$

$$p(x \mid \lambda) = \sum_z p(x, z \mid \lambda) = \sum_z \left\{ p(z_1 \mid \lambda) \prod_{t=1}^{T-1} p(z_{t+1} \mid z_t, \lambda) \prod_{t=1}^{T} p(x_t \mid z_t, \lambda) \right\}$$

p(z|λ) ***p(x|z,λ)***

where

$$p(z_1) = \prod_{j=1}^{K} (\pi_j)^{z_{1j}}$$

$$p(z_{t+1} \mid z_t) = \prod_{j=1}^{K} \prod_{k=1}^{K} (A_{jk})^{z_{t,j} z_{t+1,k}}$$

$$p(x_t \mid z_t) = \prod_{j=1}^{K} \left(p(x \mid \theta_j)\right)^{z_{tj}}$$

Applying EM algorithm for parameter estimation of HMM

Expectation of **complete data (Q-function)**

$$Q\left(\lambda;\lambda^{(old)}\right)=E\left[\ln p\left(x,z\mid\lambda\right)\right]=$$

$$=E\left[\sum_{k=1}^{K} z_{1k}\ln\pi_k+\sum_{t=1}^{T-1}\sum_{j=1}^{K}\sum_{k=1}^{K} z_{tj}z_{t+1k}\ln A_{jk}+\sum_{t=1}^{T}\sum_{k=1}^{K} z_{tk}\ln p\left(x_t\mid\theta_k\right)\right]_{\lambda^{(old)}}=$$

$$\sum_{k=1}^{K}E\left[z_{1k}\right]_{\lambda^{(old)}}\ln\pi_k+\sum_{t=1}^{T-1}\sum_{j=1}^{K}\sum_{k=1}^{K}E\left[z_{tj}z_{t+1k}\right]_{\lambda^{(old)}}\ln A_{jk}+\sum_{t=1}^{T}\sum_{k=1}^{K}E\left[z_{tk}\right]_{\lambda^{(old)}}\ln p\left(x_t\mid\theta_k\right)$$

Applying EM algorithm for parameter estimation of HMM

Expectation of **complete data** (**Q-function**)

$$Q\left(\lambda;\lambda^{(old)}\right)=E\left[\ln p(x,z\,|\,\lambda)\right]=$$

$$=E\left[\sum_{k=1}^{K} z_{1k}\ln\pi_k+\sum_{t=1}^{T-1}\sum_{j=1}^{K}\sum_{k=1}^{K} z_{tj}z_{t+1k}\ln A_{jk}+\sum_{t=1}^{T}\sum_{k=1}^{K} z_{tk}\ln p\left(x_t\,|\,\theta_k\right)\right]_{\lambda^{(old)}}=$$

$$\sum_{k=1}^{K} E\left[z_{1k}\right]_{\lambda^{(old)}}\ln\pi_k+\sum_{t=1}^{T-1}\sum_{j=1}^{K}\sum_{k=1}^{K} E\left[z_{tj}z_{t+1k}\right]_{\lambda^{(old)}}\ln A_{jk}+\sum_{t=1}^{T}\sum_{k=1}^{K} E\left[z_{tk}\right]_{\lambda^{(old)}}\ln p\left(x_t\,|\,\theta_k\right)$$

$$Q\left(\lambda;\lambda^{(old)}\right)=$$

$$\sum_{k=1}^{K}\gamma_{1k}{}^{(old)}\ln\pi_k+\sum_{t=1}^{T-1}\sum_{j=1}^{K}\sum_{k=1}^{K}\xi_{jk}{}^{(old)}\ln A_{jk}+\sum_{t=1}^{T}\sum_{k=1}^{K}\gamma_{tk}{}^{(old)}\ln p\left(x_t\,|\,\theta_k\right)$$

$\gamma\left(z_1=k\right)=p\left(z_{ik}=1\right)$ $\qquad$ $\xi\left(z_t=j,z_{t+1}=k\right)$ $\qquad$ $\gamma\left(z_t=k\right)$

Applying EM algorithm for training HMM (cont.)

E-step: ***Calculation of posterior (old) values of last step***

$$\gamma_{tk}^{(old)} = \gamma^{(old)}(z_t = k \mid x) = \frac{a^{(old)}(z_t = k)\beta^{(old)}(z_t = k)}{\sum_{j=1,\dots,K} a^{(old)}(z_t = j)\beta^{(old)}(z_t = j)}$$

$$\xi_{jk}^{(old)} = \xi^{(old)}(z_t = j, z_{t+1} = k) =$$

$$= \frac{a^{(old)}(z_t = j) A_{jk}^{(old)} p\left(x_t \mid \theta_k^{(old)}\right)\beta^{(old)}(z_t = k)}{\sum_{m=1}^{K}\sum_{n=1}^{K} a^{(old)}(z_t = m) A_{mn}^{(old)} p\left(x_t \mid \theta_n^{(old)}\right)\beta^{(old)}(z_t = n)}$$

Applying EM algorithm for training HMM (cont.)

M-step : *maximization of Q-function*

$$\max_{\lambda=\{\pi,A,\Theta\}} \left\{ \sum_{k=1}^{K} \gamma_{1k}^{(old)} \ln \pi_k + \sum_{t=1}^{T-1} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi_{jk}^{(old)} \ln A_{jk} + \sum_{t=1}^{T} \sum_{k=1}^{K} \gamma_{tk}^{(old)} \ln p(x_t \mid \theta_k) \right\}$$

$$s.t. \ \sum_{k=1}^{K} \pi_k = 1 \qquad \forall j = 1, \ldots, K \ \ \sum_{k=1}^{K} A_{jk} = 1$$

constraints

Update rules:

$$\pi_k = \gamma_{1k}^{(old)}$$

$$A_{jk} = \frac{\sum_{t=1}^{T-1} \xi_{jk}^{(old)}}{\sum_{k'=1}^{K} \sum_{t=1}^{T-1} \xi_{jk'}^{(old)}}$$

$$\mu_j = \frac{\sum_{t=1}^{T} \gamma_j^{(old)} x_t}{\sum_{t=1}^{T} \gamma_j^{(old)}}$$

$$\Sigma_k = \frac{\sum_{t=1}^{T} \gamma_k^{(old)} (x_t - \mu_j)(x_t - \mu_j)^T}{\sum_{t=1}^{T} \gamma_k^{(old)}}$$

[4]. Making predictions with HMM

Calculating: $$p\left(x_{T+1} \mid \mathbf{x}\right) = p\left(x_{T+1} \mid x_1, x_2, \ldots, x_T\right)$$

[4]. Making predictions with HMM

Calculating:

$$p\left(x_{T+1} \mid \mathbf{x}\right) = p\left(x_{T+1} \mid x_1, x_2, \ldots, x_T\right)$$

$$p\left(x_{T+1} \mid \mathbf{x}\right) = \sum_{z_{T+1}} p\left(x_{T+1}, z_{T+1} \mid \mathbf{x}\right) = \sum_{z_{T+1}} p\left(x_{T+1} \mid z_{T+1}\right) p\left(z_{T+1} \mid \mathbf{x}\right) =$$

[4]. Making predictions with HMM

Calculating: $$p\left(x_{T+1} \mid \mathbf{x}\right) = p\left(x_{T+1} \mid x_1, x_2, \ldots, x_T\right)$$

$$p\left(x_{T+1} \mid \mathbf{x}\right) = \sum_{z_{T+1}} p\left(x_{T+1}, z_{T+1} \mid \mathbf{x}\right) = \sum_{z_{T+1}} p\left(x_{T+1} \mid z_{T+1}\right) p\left(z_{T+1} \mid \mathbf{x}\right) =$$

$$= \sum_{z_{T+1}} p\left(x_{T+1} \mid z_{T+1}\right) \sum_{z_T} p\left(z_{T+1}, z_T \mid \mathbf{x}\right) =$$

[4]. Making predictions with HMM

Calculating: $p(x_{T+1} \mid \mathbf{x}) = p(x_{T+1} \mid x_1, x_2, \ldots, x_T)$

$$p(x_{T+1} \mid \mathbf{x}) = \sum_{z_{T+1}} p(x_{T+1}, z_{T+1} \mid \mathbf{x}) = \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) p(z_{T+1} \mid \mathbf{x}) =$$

$$= \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) \sum_{z_T} p(z_{T+1}, z_T \mid \mathbf{x}) =$$

$$= \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) \sum_{z_T} p(z_{T+1} \mid z_T) p(z_T \mid \mathbf{x}) =$$

$\gamma(z_T)$

[4]. Making predictions with HMM

Calculating: $$p(x_{T+1} \mid \mathbf{x}) = p(x_{T+1} \mid x_1, x_2, \ldots, x_T)$$

$$p(x_{T+1} \mid \mathbf{x}) = \sum_{z_{T+1}} p(x_{T+1}, z_{T+1} \mid \mathbf{x}) = \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) p(z_{T+1} \mid \mathbf{x}) =$$

$$= \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) \sum_{z_T} p(z_{T+1}, z_T \mid \mathbf{x}) =$$

$$= \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) \sum_{z_T} p(z_{T+1} \mid z_T) p(z_T \mid \mathbf{x}) =$$

$$\gamma(z_T)$$

$$= \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) \sum_{z_T} p(z_{T+1} \mid z_T) \frac{a(z_T)}{p(\mathbf{x})}$$

[4]. Making predictions with HMM

Calculating:

$$p(x_{T+1} \mid \mathbf{x}) = p(x_{T+1} \mid x_1, x_2, \ldots, x_T)$$

$$p(x_{T+1} \mid \mathbf{x}) = \sum_{z_{T+1}} p(x_{T+1}, z_{T+1} \mid \mathbf{x}) = \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) p(z_{T+1} \mid \mathbf{x}) =$$

$$= \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) \sum_{z_T} p(z_{T+1}, z_T \mid \mathbf{x}) =$$

$$= \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) \sum_{z_T} p(z_{T+1} \mid z_T) p(z_T \mid \mathbf{x}) =$$

$\gamma(z_T)$

$$= \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) \sum_{z_T} p(z_{T+1} \mid z_T) \frac{a(z_T)}{p(\mathbf{x})}$$

$$p(x_{T+1} \mid \mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) \sum_{z_T} p(z_{T+1} \mid z_T) a(z_T)$$

[4]. Making prediction with HMM

$$p(x_{T+1} \mid \mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) \sum_{z_T} p(z_{T+1} \mid z_T) a(z_T)$$

- Appropriate for **real time applications.** Rapid computation.

[4]. Making prediction with HMM

$$p\left(x_{T+1} \mid \mathbf{x}\right)=\frac{1}{p(\mathbf{x})}\sum_{z_{T+1}} p\left(x_{T+1} \mid z_{T+1}\right)\sum_{z_T} p\left(z_{T+1} \mid z_T\right)a\left(z_T\right)$$

- Appropriate for **real time applications.** Rapid computation.
- **Prediction distribution** can be seen as a **mixture model**

[4]. Making prediction with HMM

$$p(x_{T+1} \mid \mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) \sum_{z_T} p(z_{T+1} \mid z_T) a(z_T)$$

- Appropriate for **real time applications.** Rapid computation.
- **Prediction distribution** can be seen as a **mixture model**

$$p(x_{T+1} \mid x) = \frac{1}{p(x)} \sum_{j=1}^{K} P(j) p(x_{T+1} \mid j)$$

[4]. Making prediction with HMM

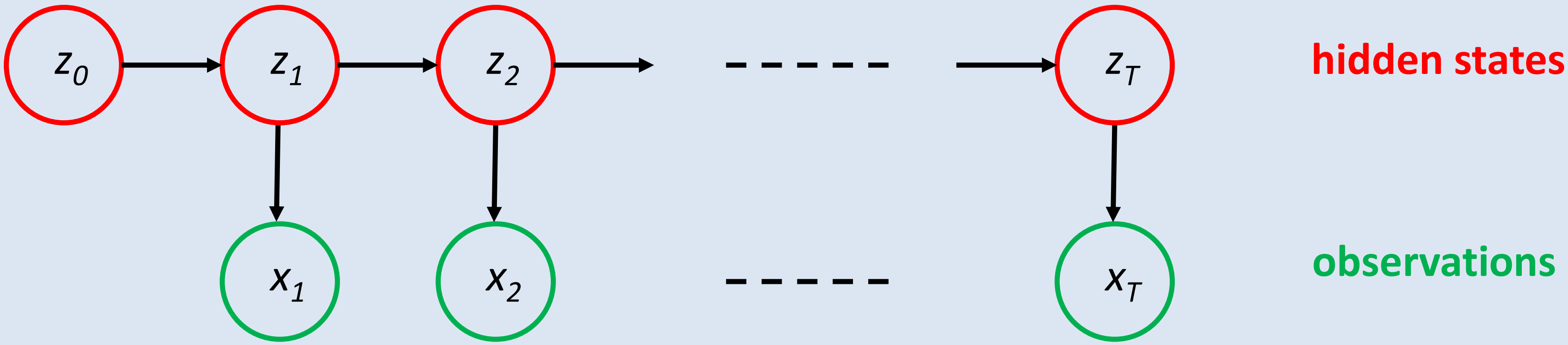
$$p(x_{T+1} \mid \mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{z_{T+1}} p(x_{T+1} \mid z_{T+1}) \sum_{z_T} p(z_{T+1} \mid z_T) a(z_T)$$

- Appropriate for **real time applications.** Rapid computation.
- **Prediction distribution** can be seen as a **mixture model**

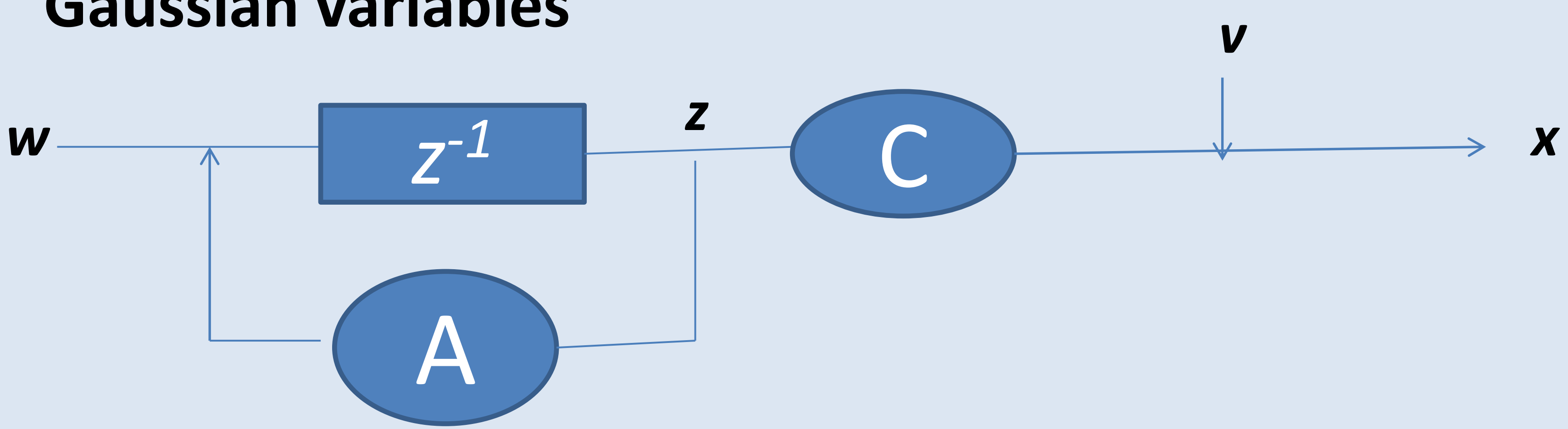
$$p(x_{T+1} \mid x) = \sum_{j=1}^{K} \pi_j \, p(x_{T+1} \mid j)$$

Kalman Filters

- Linear state space models



- **States** & **Observations are continuous and jointly Gaussian variables**

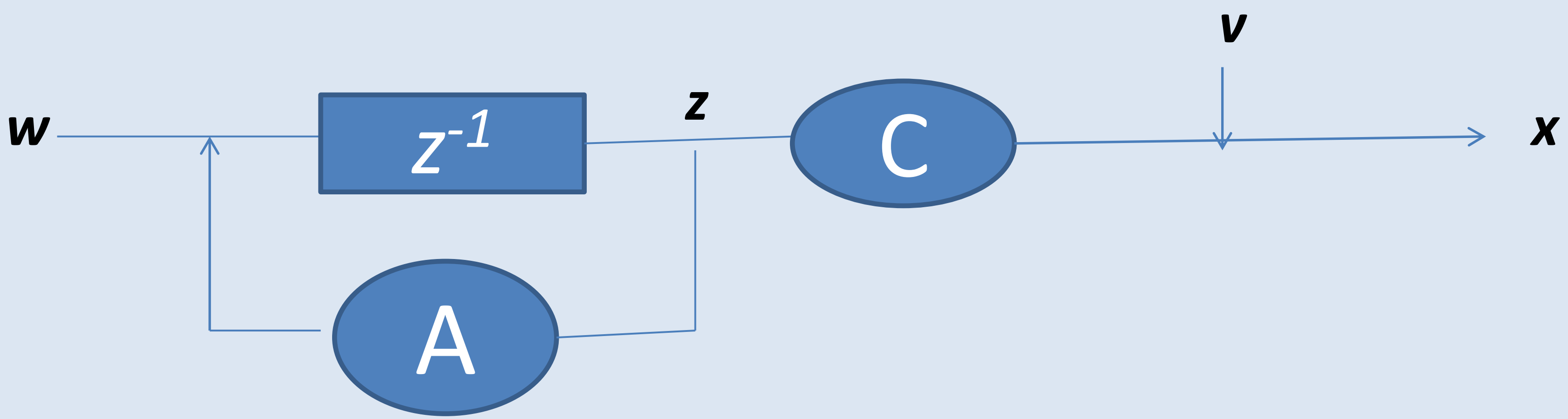


Kalman Filters

$$z_t = Az_{t-1} + w_t \qquad w_t \sim N(0, \Gamma)$$

$$x_t = Cz_t + v_t \qquad v_t \sim N(0, \Sigma)$$

$$z_1 = \mu_0 + u \qquad u \sim N(0, v_0)$$



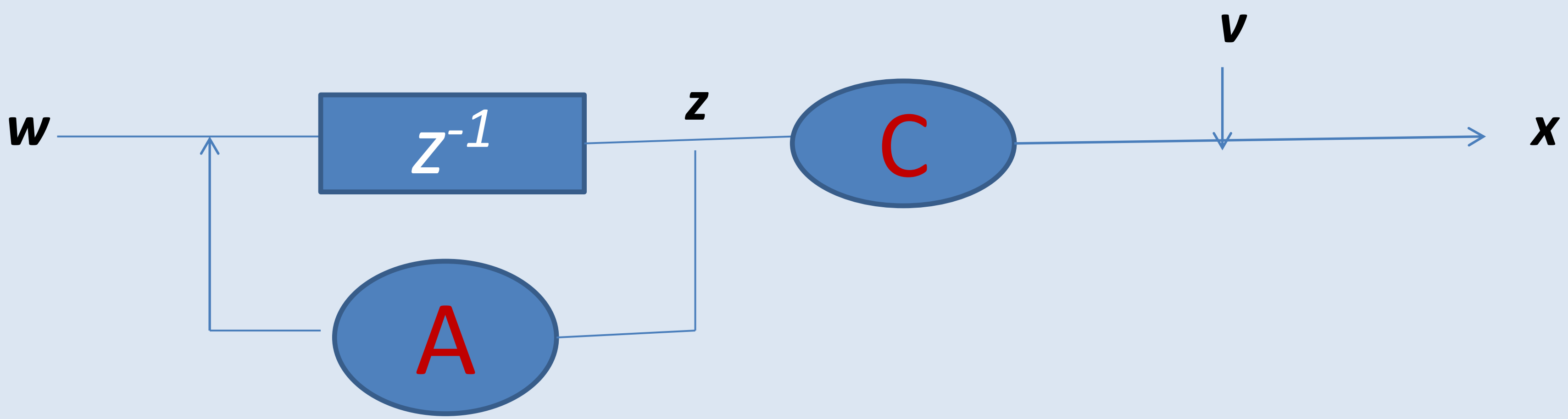
Kalman Filters

$$z_t = Az_{t-1} + w_t \qquad w_t \sim N(0, \Gamma)$$

$$x_t = Cz_t + v_t \qquad v_t \sim N(0, \Sigma)$$

$$z_1 = \mu_0 + u \qquad u \sim N(0, v_0)$$

- Set of parameters: ***$\Theta = \{ A, \Gamma, C, \Sigma, \mu_0, v_0 \}$***



Kalman Filters

$$z_t = Az_{t-1} + w_t \qquad w_t \sim N(0, \Gamma)$$

$$x_t = Cz_t + v_t \qquad v_t \sim N(0, \Sigma)$$

$$z_1 = \mu_0 + u \qquad u \sim N(0, v_0)$$

- Set of parameters: ***Θ={ A, Γ, C, Σ, μ_0, v_0 }***

$$p(z_1) = N(\mu_0, v_0)$$

$$p(z_t \mid z_{t-1}) = N(Az_{t-1}, \Gamma)$$

$$p(x_t \mid z_t) = N(Cz_t, \Sigma)$$

- **Posterior distribution of state**

forward

$$\hat{a}(z_t) = p(z_t \mid x_1,\ldots,x_t) = \frac{p(z_t, x_1,\ldots,x_t)}{p(x_1,\ldots,x_t)} = \frac{\alpha(z_t)}{p(x_1,\ldots,x_t)}$$

- **Posterior of observation**

$$c_n = p(x_n \mid x_1,\ldots,x_{n-1})$$

Join : $p(x_1,\ldots,x_t) = c_1 c_2 \cdots c_t = \prod_{n=1}^{t} c_n$

- **forward:** $a(z_t) = p(x_1,\ldots,x_t,z_t) = \int p(x_1,\ldots,x_{t-1},x_t,z_{t-1},z_t)dz_{t-1} =$

$$= \int p(x_1,\ldots,x_{t-1},z_{t-1})p(x_t,z_{t-1} \mid z_{t-1})dz_{t-1}$$

$$a(z_t) = p(x_t \mid z_t)\int a(z_{t-1})p(z_t \mid z_{t-1})dz_{t-1}$$

- **By combining the relations:**

$$\hat{a}(z_t) = \frac{\alpha(z_t)}{p(x_1,\ldots,x_t)} \Rightarrow \alpha(z_t) = \hat{a}(z_t)p(x_1,\ldots,x_t)$$

$$p(x_1,\ldots,x_t) = \prod_{n=1}^{t} c_n \qquad c_t = p(x_t \mid x_1,\ldots,x_{t-1})$$

$$a(z_t) = p(x_t \mid z_t)\int a(z_{t-1})p(z_t \mid z_{t-1})dz_{t-1}$$

- **we obtain:**

$$\hat{a}(z_t)\prod_{n=1}^{t} c_n = p(x_t \mid z_t)\int \hat{a}(z_{t-1})\prod_{m=1}^{t-1} c_m p(z_t \mid z_{t-1})dz_{t-1}$$

$$c_t\hat{a}(z_t) = p(x_t \mid z_t)\int \hat{a}(z_{t-1})p(z_t \mid z_{t-1})dz_{t-1}$$

- Since **all distributions are Gaussians**

$$a(z_t) = p(x_t \mid z_t) \int a(z_{t-1}) p(z_t \mid z_{t-1}) dz_{t-1}$$ **is Gaussian**

The distribution of state's prediction is also Gaussian:

$$\hat{a}(z_t) = p(z_t \mid x_1, \ldots, x_t) = \cdots = N(\mu_t, V_t)$$

- Therefore, the recursion equation becomes:

$$c_t \hat{a}(z_t) = p(x_t \mid z_t) \int \hat{a}(z_{t-1}) p(z_t \mid z_{t-1}) dz_{t-1}$$

$$\Downarrow$$

$$c_t N(\mu_t, V_t) = N(C z_t, \Sigma) \int N(\mu_{t-1}, V_{t-1}) N(A z_{t-1}, \Gamma) dz_{t-1}$$

are already known (μ_{t-1}, V_{t-1})

- Following the Gaussian properties we obtain:

$$\mu_t = A\mu_{t-1} + K_t\left(x_t - CA\mu_{t-1}\right)$$
$$V_t = P_{t-1} - K_t C P_{t-1}$$

Prediction (posterior) of state

$$c_t = N\left(x_t \mid CA\mu_{t-1}, CP_{t-1}C^T + \Sigma\right)$$

Prediction (posterior) of observation

$$K_t = P_{t-1}C^T\left(CP_{t-1}C^T + \Sigma\right)^{-1}$$

Kalman gain matrix

$$P_{t-1} = \Gamma + AV_{t-1}A^T$$

- **Interpretation**

$$z_t = Az_{t-1} + w_t \qquad w_t \sim N(0, \Gamma)$$

$$x_t = Cz_t + v_t \qquad v_t \sim N(0, \Sigma)$$

$$z_1 = \mu_0 + u \qquad u \sim N(0, v_0)$$

- **Interpretation**

$$z_t = Az_{t-1} + w_t \qquad w_t \sim N(0, \Gamma)$$
$$x_t = Cz_t + v_t \qquad v_t \sim N(0, \Sigma)$$
$$z_1 = \mu_0 + u \qquad u \sim N(0, v_0)$$

- [1]. Make prediction of next observation $\hat{x}_t$

$$c_t = N\left(x_t \mid CA\mu_{t-1}, CP_{t-1}C^T + \Sigma\right)$$

- **Interpretation**

$$z_t = Az_{t-1} + w_t \qquad w_t \sim N(0,\Gamma)$$
$$x_t = Cz_t + v_t \qquad v_t \sim N(0,\Sigma)$$
$$z_1 = \mu_0 + u \qquad u \sim N(0,v_0)$$

- [1]. Make prediction of next observation $\hat{x}_t$

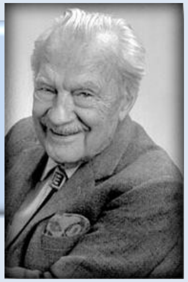
$$c_t = N\left(x_t \mid CA\mu_{t-1}, CP_{t-1}C^T + \Sigma\right)$$

- [2]. After obtaining the observation, making correction

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

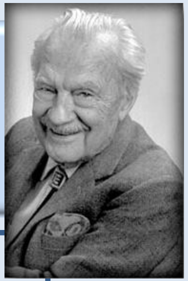
$$V_t = P_{t-1} - K_t C P_{t-1}$$

Metropolis sampling Nick Metropolis, 1953



- Generate samples from a distribution p(x)
- Transformation method
- Rejection method
- **Metropolis method**

Metropolis sampling Nick Metropolis, 1953



- Nick Metropolis (1915-99): Greek-American physicist
- Suppose we want to generate samples from p(x).
- **Idea:** Create a Markov Chain such that p(x) to be its **stationary distribution**.
- Thus, after reaching the stationary state, every movement we make is a sample from p(x).

Metropolis sampling Nick Metropolis, 1953

- Start from an initial state $x^{(0)}$ and generate a sequence of transitions $\{x^{(0)}, x^{(1)}, \ldots, x^{(t)}, x^{(t+1)}, \ldots\}$.

- Use transition function $q(y|x^{(t)})$ move into a new state y.

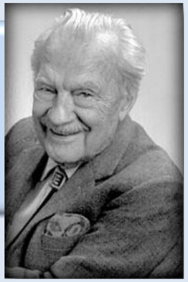
- **Assumption:** function $q(y|x)$ is symmetric.

- Take the likelihood ratio $a(x,y) = p(y)/p(x)$

- If $a(x,y) > 1$ => accept y ,i.e. $x^{(t+1)} = y$

  else if $a(x,y) < 1$ accept y with probability $a(x,y)$ and reject it with probability $(1-a(x,y))$.

Metropolis sampling Nick Metropolis, 1953



- In general, we accept a new state with probability:

$$a(x,y) = \min\left\{\frac{p(y)}{p(x)}, 1\right\}$$

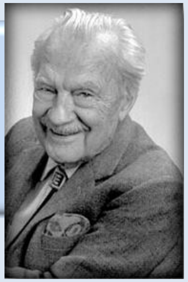
- Thus, we generate a sequence of states with increased probability:

$$p\left(x^{(0)}\right) < p\left(x^{(1)}\right) < p\left(x^{(2)}\right) < \ldots < p\left(x^{(t)}\right) < p\left(x^{(t+1)}\right) < p\left(x^{(t+2)}\right) < \ldots$$

$$x^{(0)}, x^{(1)}, x^{(2)}, \ldots, x^{(t)}, x^{(t+1)}, x^{(t+2)}, \ldots$$

burn-in period | ***samples from p(x)***

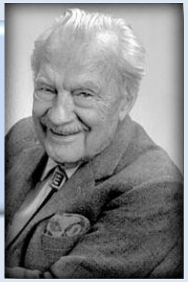
Metropolis-Hastings, 1970



- **Generalization of Metropolis**.
- Cancellation of the assumption of symmetric transition function, i.e. $q(y|x) \neq q(x|y)$.
- Probability of accepting a new state:

$$a(x,y) = \min\left\{ \frac{q(x|y)\,p(y)}{q(y|x)\,p(x)}, 1 \right\}$$

Metropolis-Hastings, 1970



- Transition Probability

$$P(x \mapsto y) = q(y \mid x)a(x, y) = \min\left\{q(x \mid y)\frac{p(y)}{p(x)}, q(y \mid x)\right\}$$

- Respectively

$$P(y \mapsto x) = q(x \mid y)a(y, x) = \min\left\{q(y \mid x)\frac{p(x)}{p(y)}, q(x \mid y)\right\}$$

- Then, **p(x) is stationary distribution since**:

$$p(x)P(x \mapsto y) = p(y)P(y \mapsto x)$$