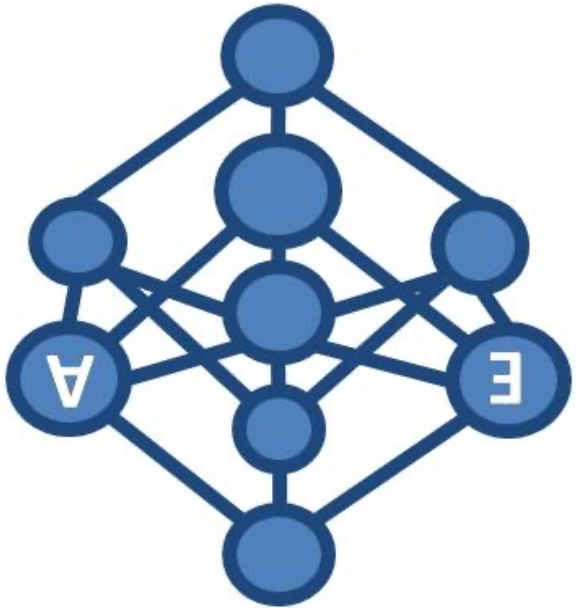
Probabilistic Graphical Models*

Dynamic Bayesian Networks

TECHNISCHE UNIVERSITÄT DARMSTADT



*Thanks to Carlos Guestrin, Pedro Domingos and many others for making their slides publically available

So far

- Bayesian networks
- Inference within Bayesian networks
- Learning Bayesian networks

- **All rather static situations …**

What's next

- The world is not static but evolves over time
  - Monitoring freeway traffic (from an autonomous driver or for traffic analysis)
  - Monitoring patient's symptoms
- **How can we model dynamic environments?**

- Hidden Markov Models
- (Bayes Filter and Kalman Filters)
- Dynamic Bayesian Networks

Hidden Markov Models

Let‘s consider two examples

1. Part-of-speech tagging
2. Sequence Alignment

Example 1: Parts of Speech

- **Part of speech (POS):** Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc.
- **POS tagging**: The process of assigning (labeling) a part-of-speech or other lexical class marker to each word in a sentence (or a corpus)
  - Decide whether each word is a noun, verb, adjective, …
- **Also called word-category disambiguation**

POS examples for English

N	**noun**	chair, bandwidth, pacing
V	**verb**	study, debate, munch
ADJ	**adj**	purple, tall, ridiculous
ADV	**adverb**	unfortunately, slowly,
P	**preposition**	of, by, to
PRO	**pronoun**	I, me, mine
DET	**determiner**	the, a, that, those

POS tagging is not trivial

- Words often have more than one POS, e.g. ***back***
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB

POS Tagging Example

Word	POS listing in Brown		
heat	noun	***verb***	
oil	***noun***		
in	***prep***	noun	adv
a	***det***	noun	noun-proper
large	***adj***	noun	adv
pot	***noun***		

Some Applications

- Language Models: Tell what words are likely to occur in a word's vicinity
  - E.g. the vicinity of the possessive or person pronouns
- Tell the pronunciation of a word
  - DIScount (noun) and disCOUNT (verb) …
- Advanced language models
  - Word-class N-grams
- …

Entity Extraction

Angiotensin-converting enzyme 2 **GENE_OR_GENOME** (ACE2 **GENE_OR_GENOME**) as a SARS-CoV-2 **CORONAVIRUS** receptor: molecular mechanisms and potential therapeutic target. SARS-CoV-2 **CORONAVIRUS** has been sequenced [3 **CARDINAL**] . A phylogenetic **EVOLUTION** analysis [3 **CARDINAL** , 4 **CARDINAL**] found a bat **WILDLIFE** origin for the SARS-CoV-2 **CORONAVIRUS** . There is a diversity of possible intermediate hosts for SARS-CoV-2 **CORONAVIRUS** , including pangolins **WILDLIFE** , but not mice **EUKARYOTE** and rats **EUKARYOTE** [5 **CARDINAL**] . There are many similarities of SARS-CoV-2 **CORONAVIRUS** with the original SARS-CoV **CORONAVIRUS** . Using computer modeling , Xu et al . [6 **CARDINAL**] found that the spike proteins **GENE_OR_GENOME** of SARS-CoV-2 **CORONAVIRUS** and SARS-CoV **CORONAVIRUS** have almost identical 3-D structures in the receptor binding domain that maintains Van der Waals forces **PHYSICAL_SCIENCE** . SARS-CoV spike proteins **GENE_OR_GENOME** has a strong binding affinity to human ACE2 **GENE_OR_GENOME** , based on biochemical interaction studies and crystal structure analysis [7 **CARDINAL**] . SARS-CoV-2 **CORONAVIRUS** and SARS-CoV spike proteins **GENE_OR_GENOME** share identity in amino acid sequences and ……

https://github.com/covidgraph/documentation/issues/46

Determining Part of Speech Tags

Automatically learn Part of Speech (POS) assignment.

- The best techniques achieve 96-97% accuracy per word (and even more) on new materials, given large training corpora.

Statistical Approaches: Idea 1

Simply assign each word its most likely POS.

Success rate: 91%!

Word	POS listings in Brown		
heat	noun/89	**verb/5**	
oil	**noun/87**		
in	**prep/20731**	noun/1	adv/462
a	**det/22943**	noun/50	noun-proper/30
large	**adj/354**	noun/2	adv/5
pot	**noun/27**		

Statistical Approaches: Idea 2

For a string of words

$$W = w_1 w_2 w_3 \ldots w_n$$

find the string of POS tags

$$T = t_1\, t_2\, t_3 \ldots t_n$$

which maximizes P(T|W)

- i.e., the probability of tag string T given that the word string was W, i.e., that W was tagged T

The Sparse Data Problem …

A simple but unrealistic approach to compute P(T|W):

Count instances of the string "heat oil in a large pot" in the training corpus, and **pick the most common tag assignment** to the string.

So, what can we estimate ?

- What parameters can we estimate with a million words of hand tagged training data?
  - Assume a uniform distribution of 5000 words and 40 part of speech tags..

Event	Count	Estimate Quality?
tags	40	Excellent
tag bigrams	1600	Excellent
tag trigrams	64,000	OK
tag 4-grams	2.5M	Poor
words	5000	Very Good
word bigrams	25M	Poor
word x tag pairs	200,000	OK

- **Rich Models often require vast amounts of data**
- Good estimates of models with bad assumptions often outperform better models which are badly estimated (recall Naïve Bayes!)

So, what c

- What parame
  words of hand

However, Google changes the business:
5-gram language model trained on a
training corpus of about 1 trillion words

  - Assume a uniform distribution o ch tags..

Event	Count	Estimate Quality?
tags	40	Ex nt
tag bigrams	1600	Excellent
tag trigrams	64,000	OK
tag 4-grams	2.5M	Po r
words	5000	Very Good
word bigrams	25M	Poor
word x tag pairs	200,000	OK

- **Rich Models often require vast amounts of data**
- Good estimates of models with bad assumptions often outperform better models which are badly estimated (recall Naïve Bayes!)

A Practical Statistical Tagger

By Bayes' Rule:

$$\mathbf{P(T|W)} = \frac{P(W|T) * P(T)}{P(W)}$$

so to maximize $P(T|W)$, need to maximize $P(W|T) * P(T)$.

To compute P(T): By the chain rule,

$$P(T) = P(t_1) * P(t_2|t_1) * P(t_3|t_1 t_2) * \ldots * P(t_n|t_1 \ldots t_{n-1})$$

A Practical Statistical Tagger II

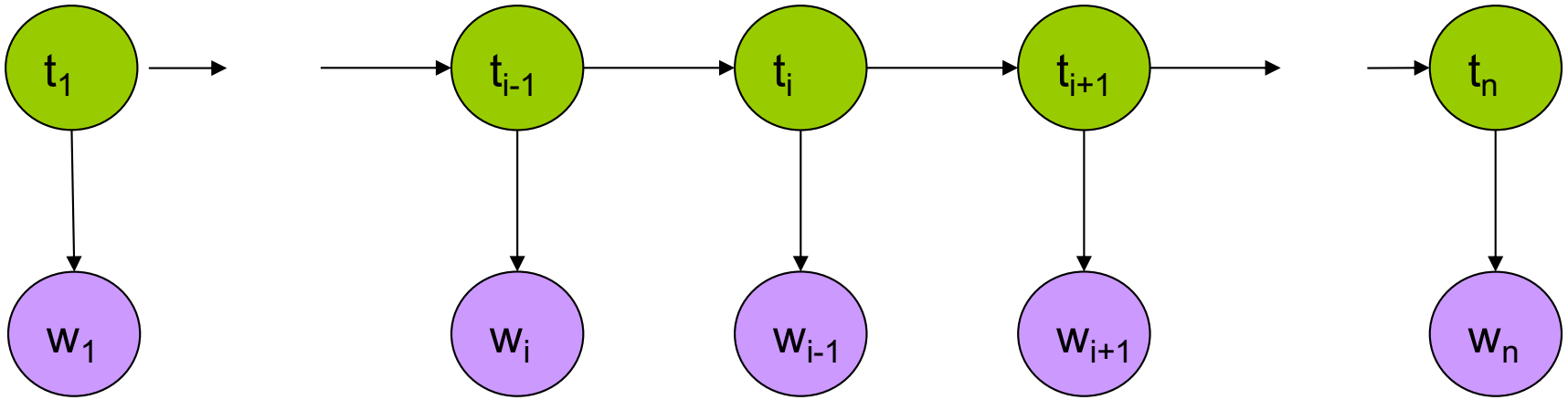
But we cannot accurately estimate more than tag bigrams! Thus, we simply change to a model that we CAN estimate:

A Markov Assumption: $\mathrm{P}(\mathrm{t_i}|\mathrm{t_1} \ldots \mathrm{t_{i-1}}) = \mathrm{P}(\mathrm{t_i}|\mathrm{t_{i-1}})$

By which

$$P(T) = P(t_1) * P(t_2|t_1) * P(t_3|t_2) * \ldots * P(t_n|t_{n-1})$$

A Graphical Representation



$$P(T) * P(W|T) =$$

$$P(t_1) * P(t_2|t_1) * P(t_3|t_2) * \ldots * P(t_n|t_{n-1}) *$$
$$P(w_1|t_1) * P(w_2|t_2) * \ldots * P(w_n|t_n)$$

Note

THIS SIMPLE MODEL HAS A NUMBER OF APPLICATIONS!

Biological Sequence Analysis

- One of the basic tasks in bioinformatics:

Proteins are the biological workhorses that carry out vital functions in every cell. To carry out their task, proteins must fold into a complex three-dimensional structure … but what tells a protein which shape it should be and how does it achieve this?

- E.g., **given a single amino acid target sequence of unknown structure, we want to infer the structure of the resulting protein** (Protein Folding Problem)
- Allows to predict the **function of the protein**!

But wait, that's hard!

- There's physics, chemistry, secondary structure, tertiary structure and all sorts of other nasty stuff to deal with

- Assumption: Proteins with similar functions fold in a similar way

The Sequence Alignment Approach

- We find a similar protein with a structure that we understand, and we see if it makes sense to fold our target into the same sort of shape.
- If not, we try again with the second most similar structure, and so on.
- That is we take advantage of the wealth of knowledge that has been collected in protein and structure databases.

"Recap": Sequence Alignment

TECHNISCHE UNIVERSITÄT DARMSTADT

Global Alignment:

Goal: How similar are two sequences S_1 and S_2

Input: two sequences S_1, S_2 over the same alphabet

Output: two sequences S'_1, S'_2 of equal length

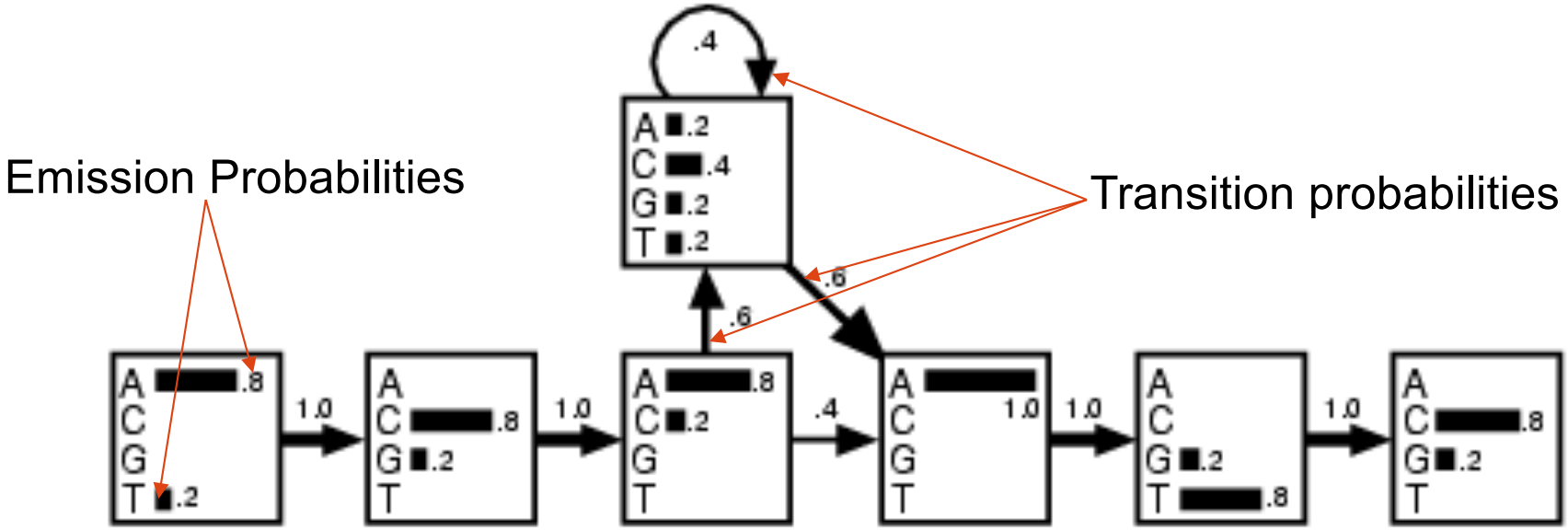
(S'_1, S'_2 are S_1, S_2 with possibly additional gaps)

Example:

- S_1 = GCGCATGGATTGAGCGA
- S_2 = TGCGCCATTGATGACC
- A possible alignment:

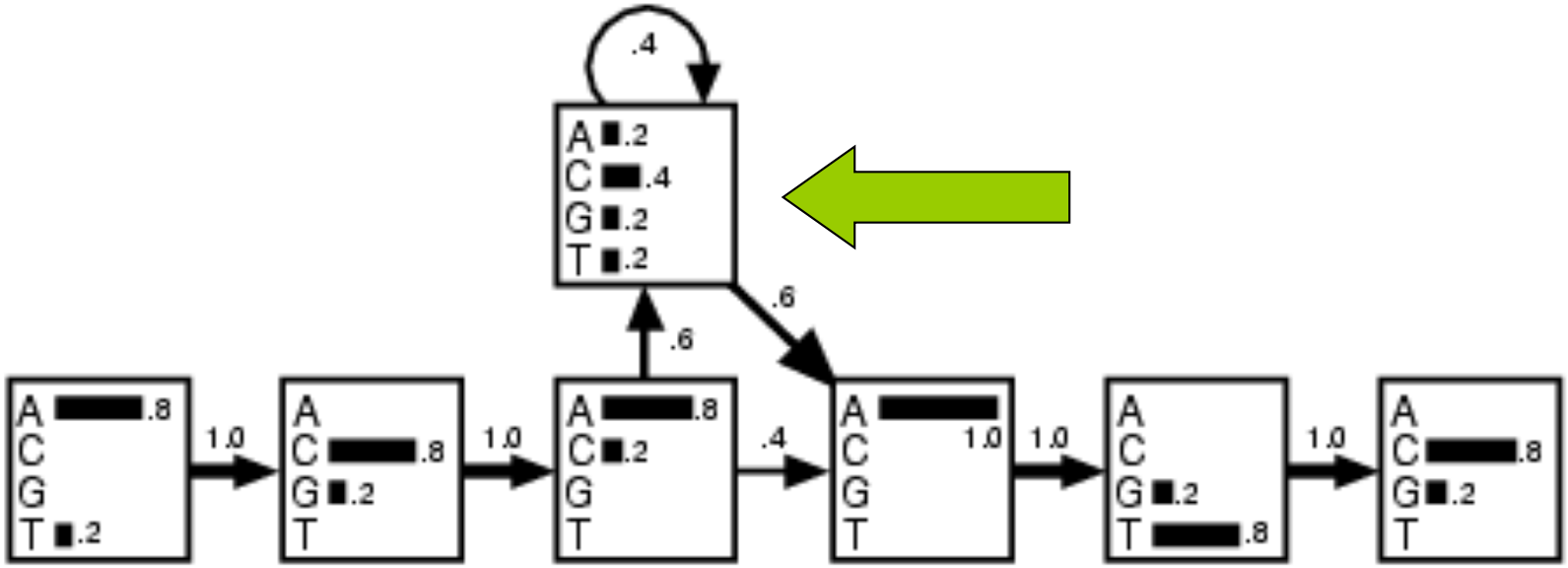
$$S'_1 = -GCGC-ATGGATTGAGCGA$$
$$S'_2 = TGCGCCATTGAT-GACC--$$

HMMs to the rescue!



A C A – – – A T G
T C A A C T A T C
A C A C – – A G C
A G A – – – A T C
A C C G – – A T C

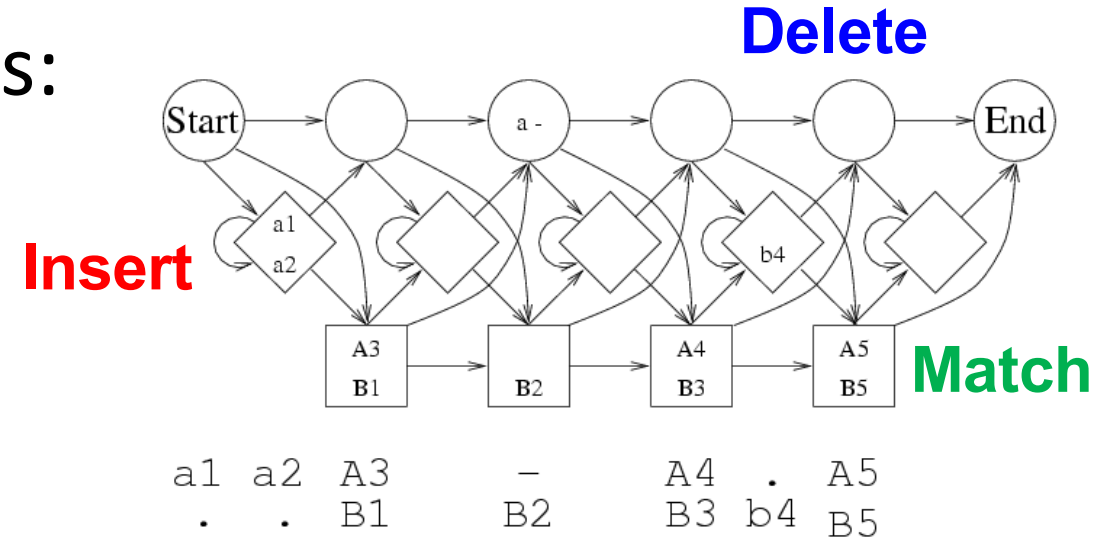
Insert (Loop) States



A	C	A	–	–	–	A	T	G
T	C	A	A	C	T	A	T	C
A	C	A	C	–	–	A	G	C
A	G	A	–	–	–	A	T	C
A	C	C	G	–	–	A	T	C

Let us implement more edit operations: Profile HMMs

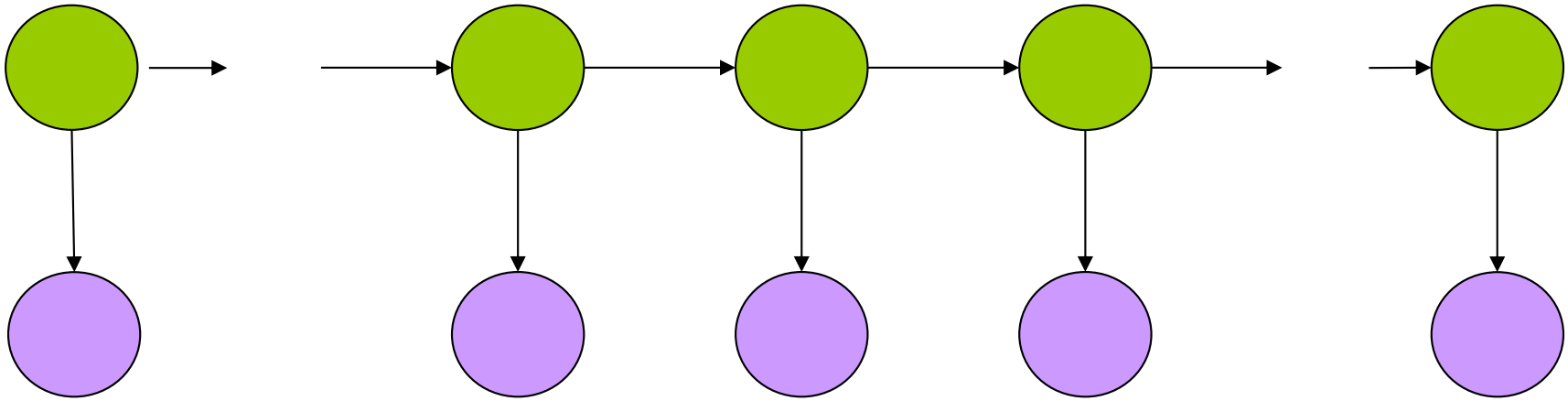
- Three types of states:
  - **Match**
  - **Insert**
  - **Delete**
- One delete and one match per position in model
- One insert per transition in model
- Start and end “dummy” states



We have seen two well-known and relevant examples of HMMs

SO, WHAT ARE HMMS?

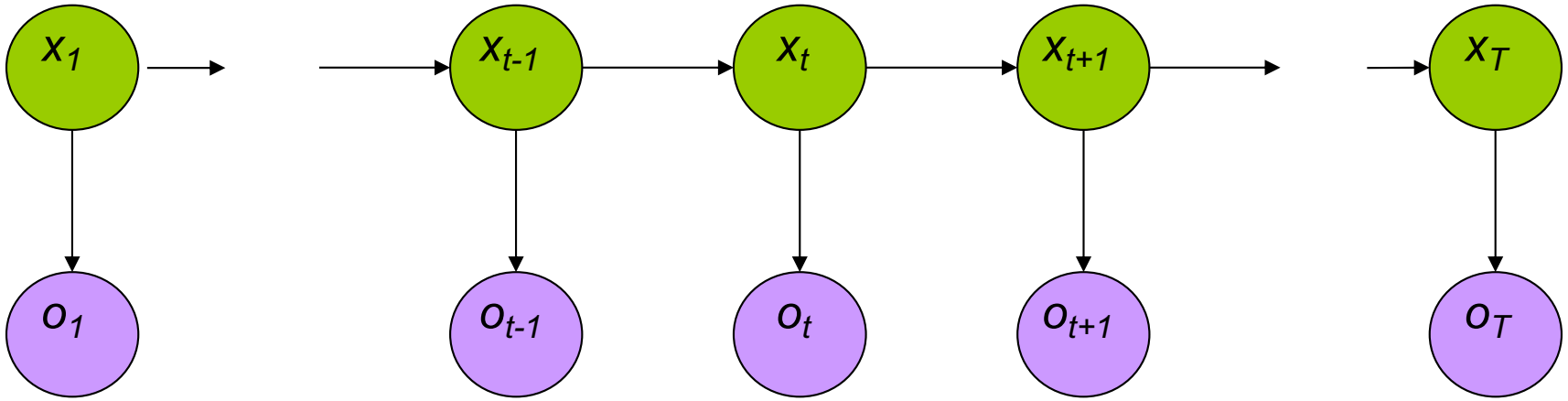
What is an HMM?



- Graphical Model
- Circles indicate states
- Arrows indicate probabilistic dependencies between states

HMM = Hidden Markov Model

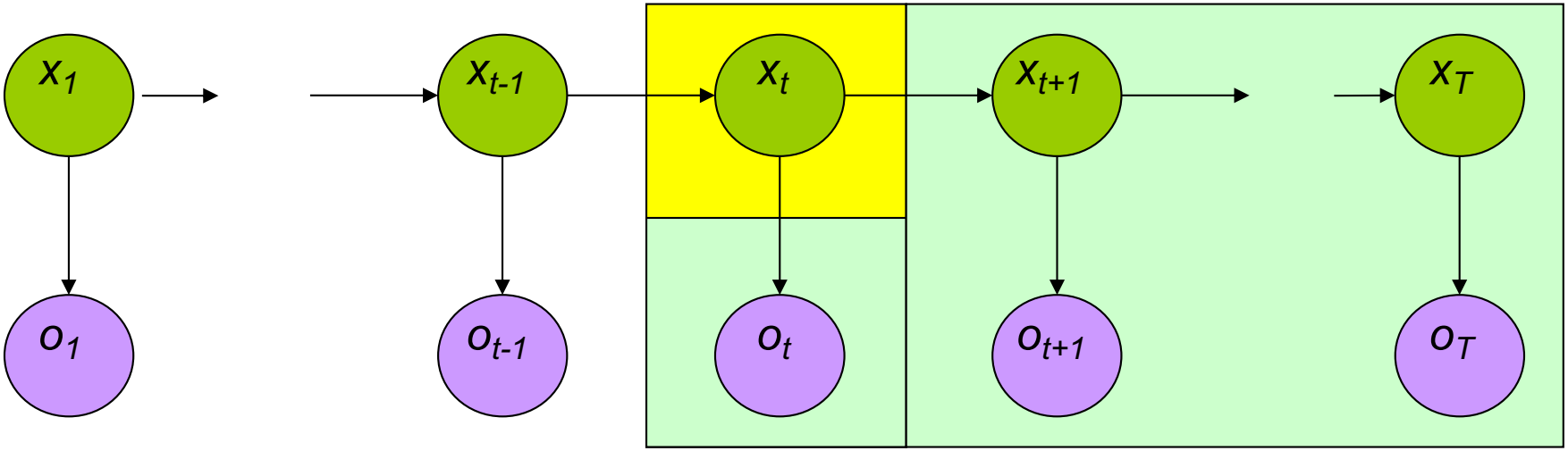
HMMs and Bayesian Nets

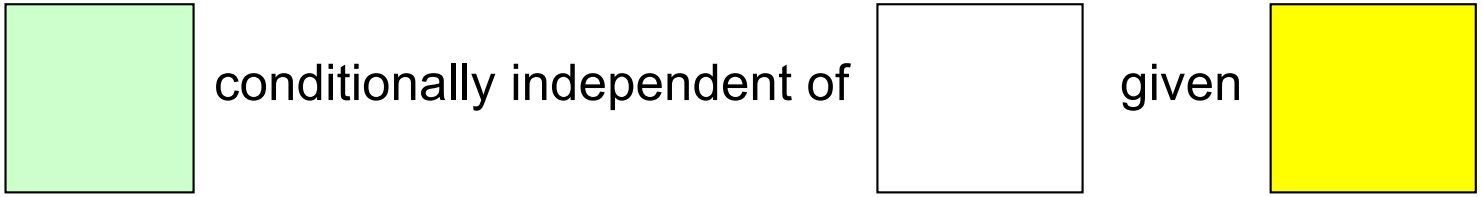


$$P(x_1 \ldots x_T, o_1 \ldots o_T) = P(x_1)P(o_1 \mid x_1)\prod_{i=1}^{T-1} P(x_{i+1} \mid x_i).P(o_{i+1} \mid x_{i+1})$$

$$= \pi_{x_1} b_{x_1 o_1} \prod_{t=1}^{T-1} a_{x_t x_{t+1}} b_{x_{t+1} o_{t+1}}$$

Independence Assumption

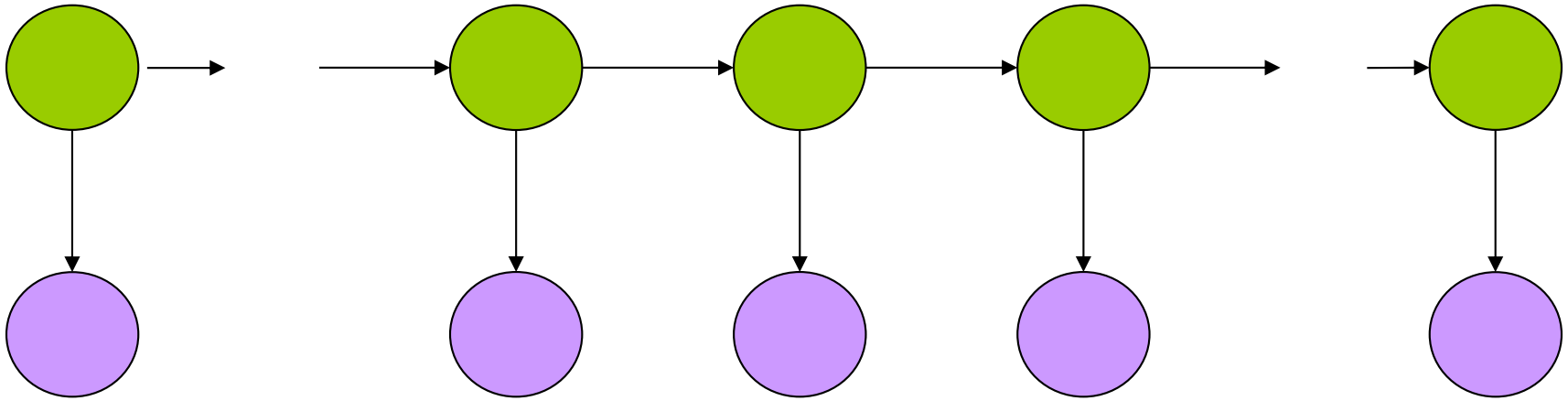




Because of d-separation

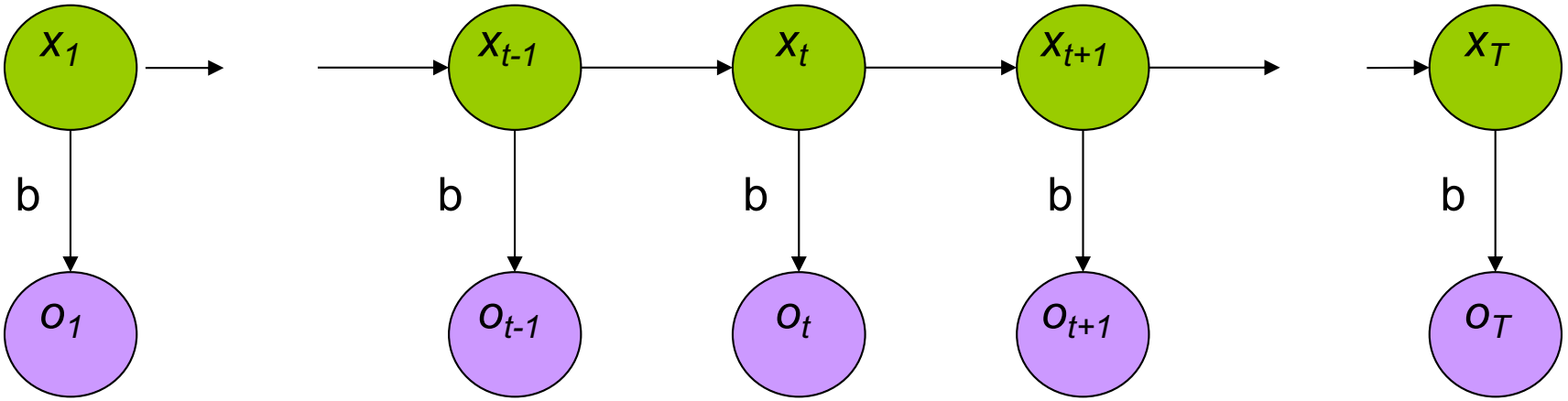
"The past is independent of the future given the present."

Inference in an HMM



1. Compute the probability of a given observation sequence
2. Given an observation sequence, compute the most likely hidden state sequence
3. Given an observation sequence and set of possible models, which model most closely fits the data?

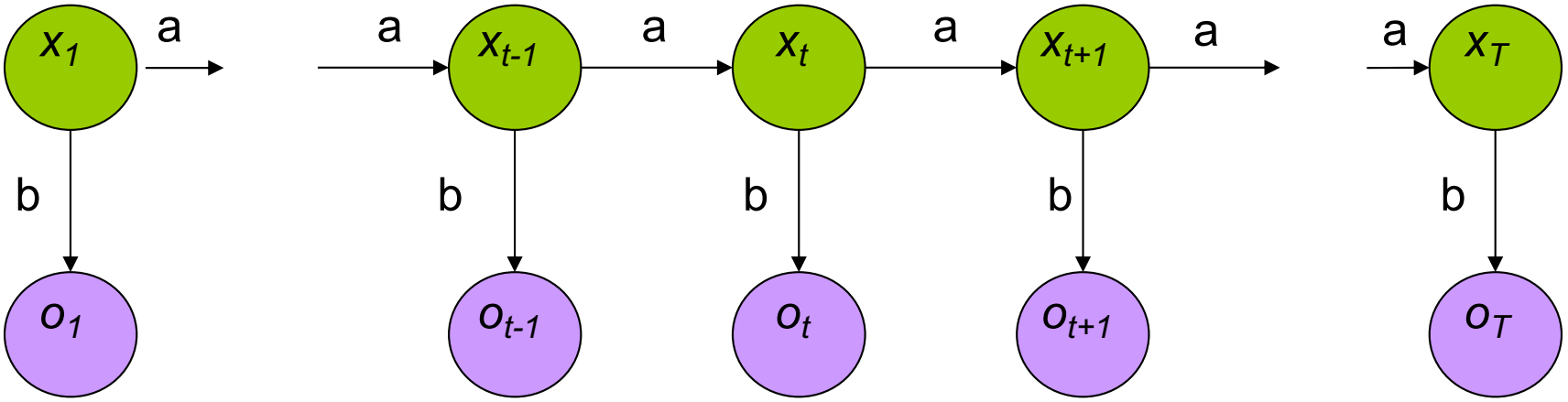
(1) Decoding



$$P(O \mid X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \ldots b_{x_T o_T}$$

μ are the parameters

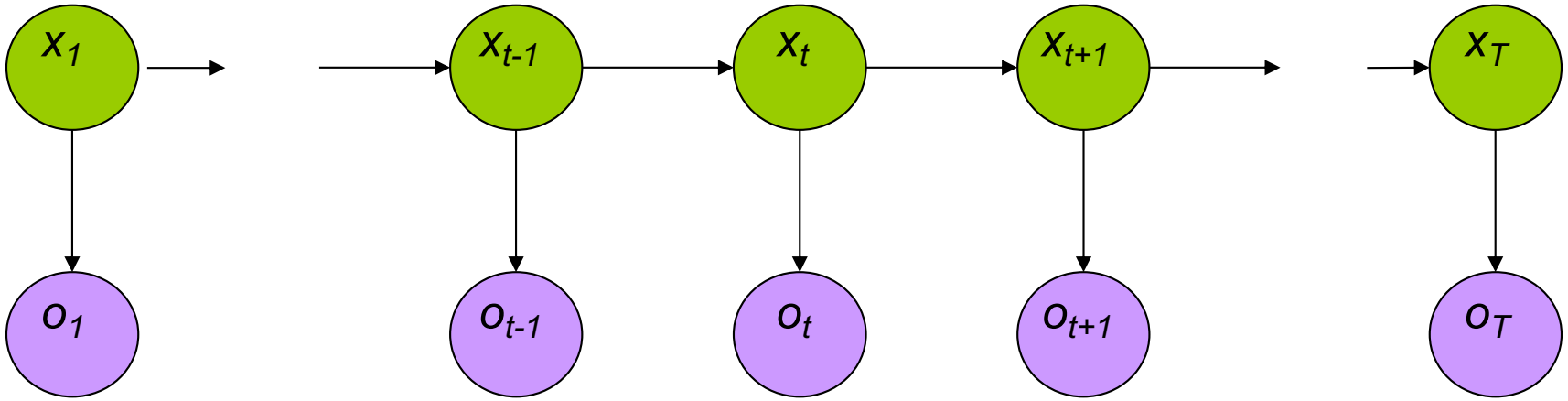
(1) Decoding



$$P(O \mid X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \ldots b_{x_T o_T}$$

$$P(X \mid \mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \ldots a_{x_{T-1} x_T}$$

(1) Decoding

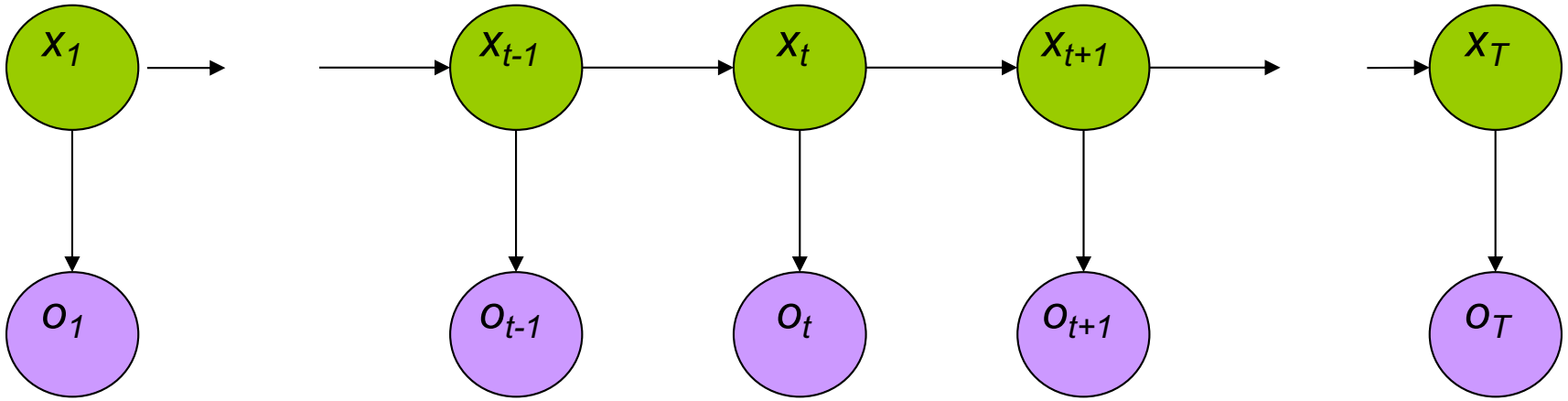


$$P(O \mid X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \ldots b_{x_T o_T}$$

$$P(X \mid \mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \ldots a_{x_{T-1} x_T}$$

$$P(O, X \mid \mu) = P(O \mid X, \mu) P(X \mid \mu)$$

(1) Decoding



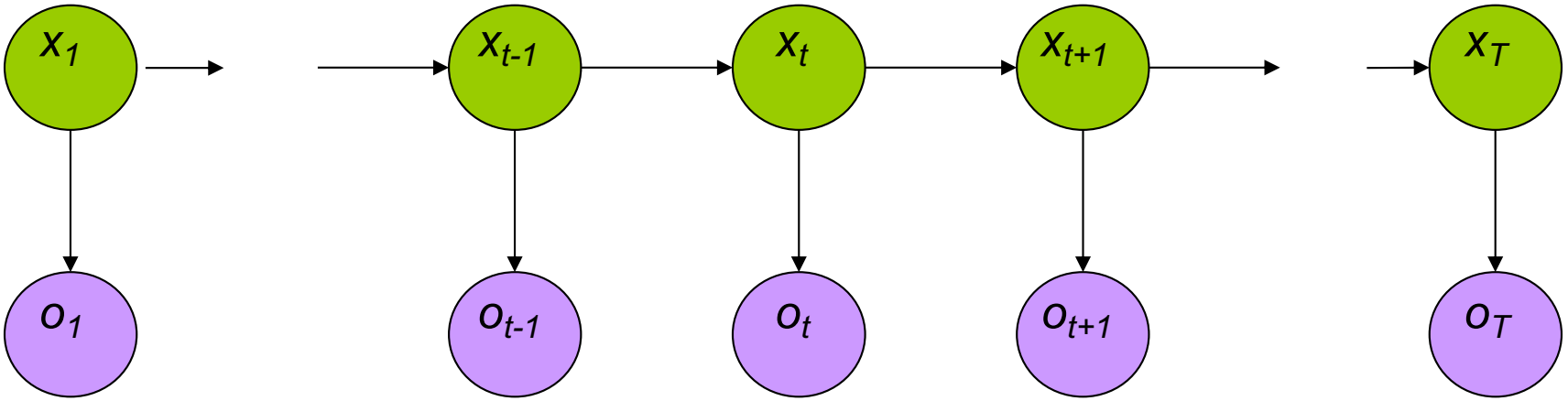
$$P(O \mid X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \ldots b_{x_T o_T}$$

$$P(X \mid \mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \ldots a_{x_{T-1} x_T}$$

$$P(O, X \mid \mu) = P(O \mid X, \mu) P(X \mid \mu)$$

$$P(O \mid \mu) = \sum_X P(O \mid X, \mu) P(X \mid \mu)$$

(1) Decoding



$$P(O \mid \mu) = \sum_X \pi_{x_1} b_{x_1 o_1} \prod_{t=1}^{T-1} a_{x_t x_{t+1}} b_{x_{t+1} o_{t+1}}$$

Complexity $O(N^T .2T)$

$E.g. N = 5, T = 100$ gives $2.100.5^{100} \approx 10^{72}$

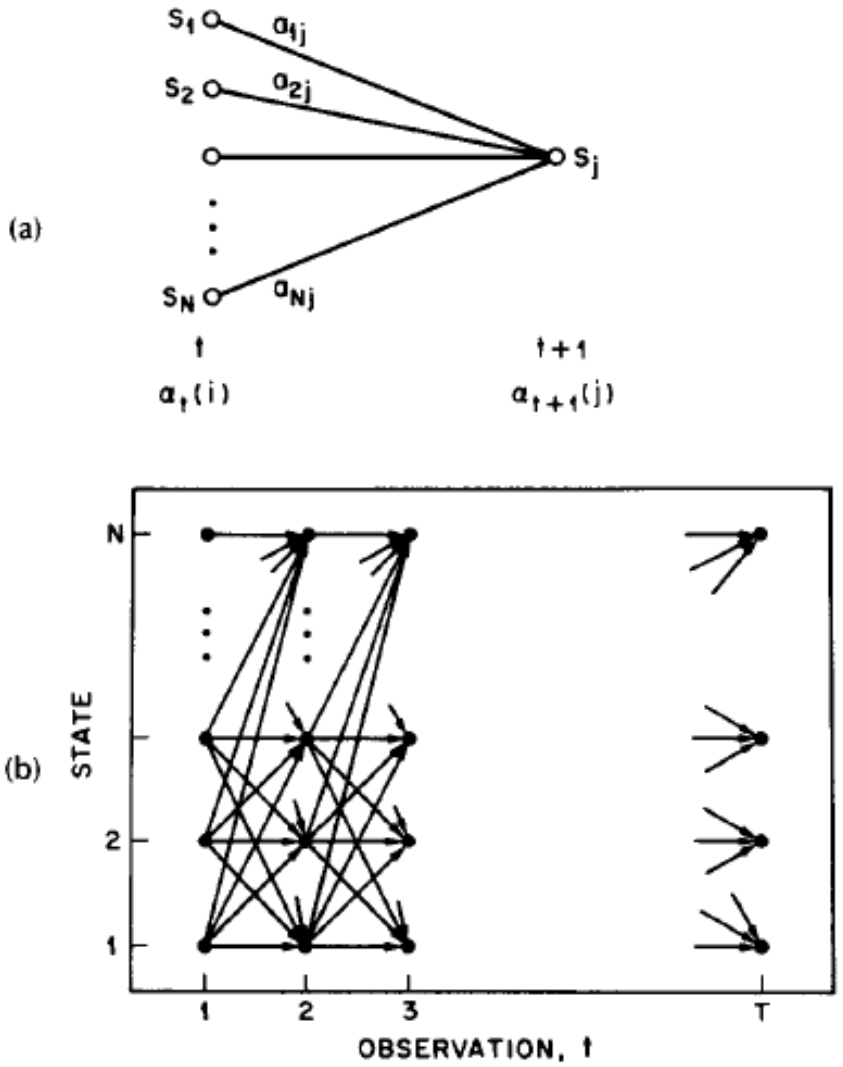
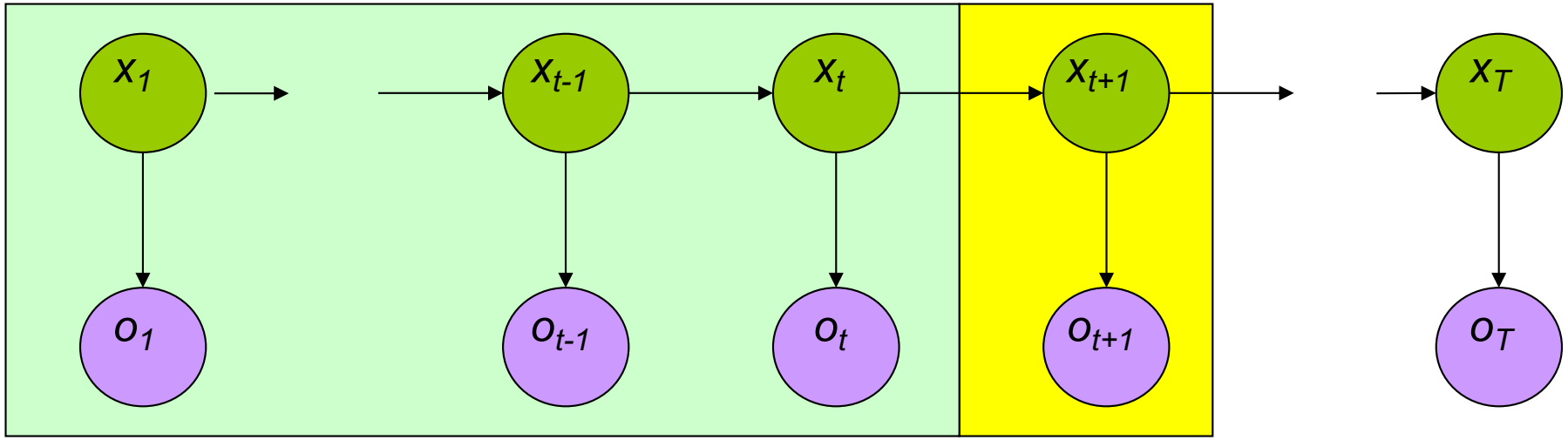


Fig. 4. (a) Illustration of the sequence of operations required for the computation of the forward variable $\alpha_{t+1}(j)$. (b) Implementation of the computation of $\alpha_t(i)$ in terms of a lattice of observations t, and states i.

Dynamic Programming to the rescue!

Forward Procedure

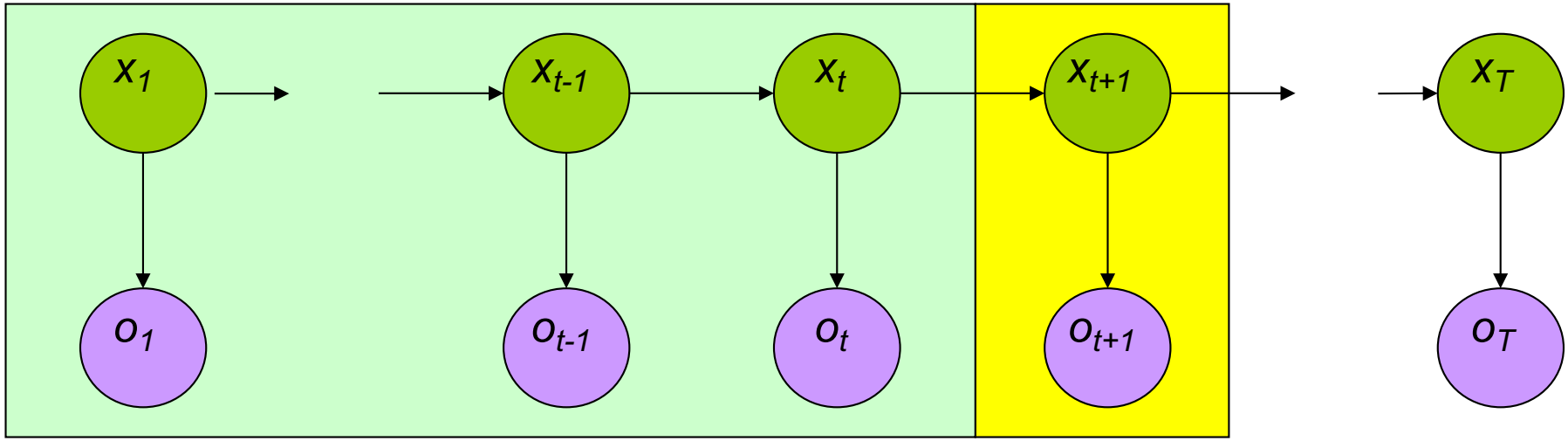


- Special structure gives us an efficient solution using *dynamic programming*
- **Intuition**: Probability of the first *t* observations is the same for all possible *t*+1 length state sequences
- **Define:**

$$\alpha_i(t) = P(o_1 \ldots o_t, x_t = i \mid \mu)$$

$$\alpha_i(1) = P(o_1, x_1 = i \mid \mu)$$
$$= \pi_i \cdot b_{io_1}$$

Forward Procedure



$$\alpha_j(t+1)$$

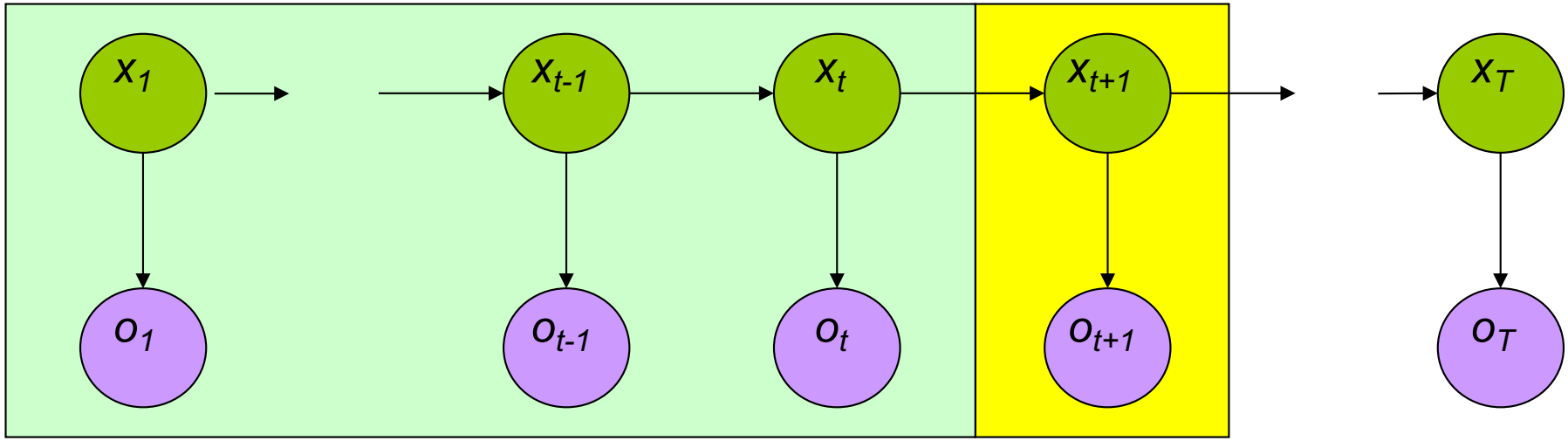
$$= P(o_1 \ldots o_{t+1}, x_{t+1} = j)$$ **definition (μ omitted)**

$$= P(o_1 \ldots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

Forward Procedure



$$\alpha_j(t+1)$$

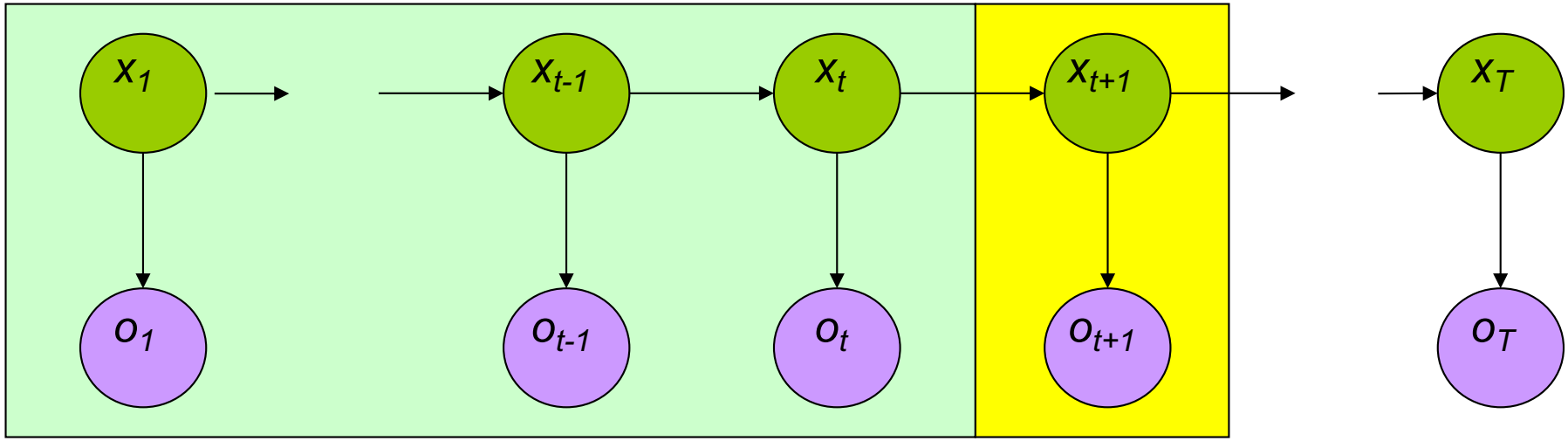
$$= P(o_1 \ldots o_{t+1}, x_{t+1} = j)$$

$$= P(o_1 \ldots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j) \qquad \text{chain rule}$$

$$= P(o_1 \ldots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

Forward Procedure



$$\alpha_j(t+1)$$

$$= P(o_1 \ldots o_{t+1}, x_{t+1} = j)$$

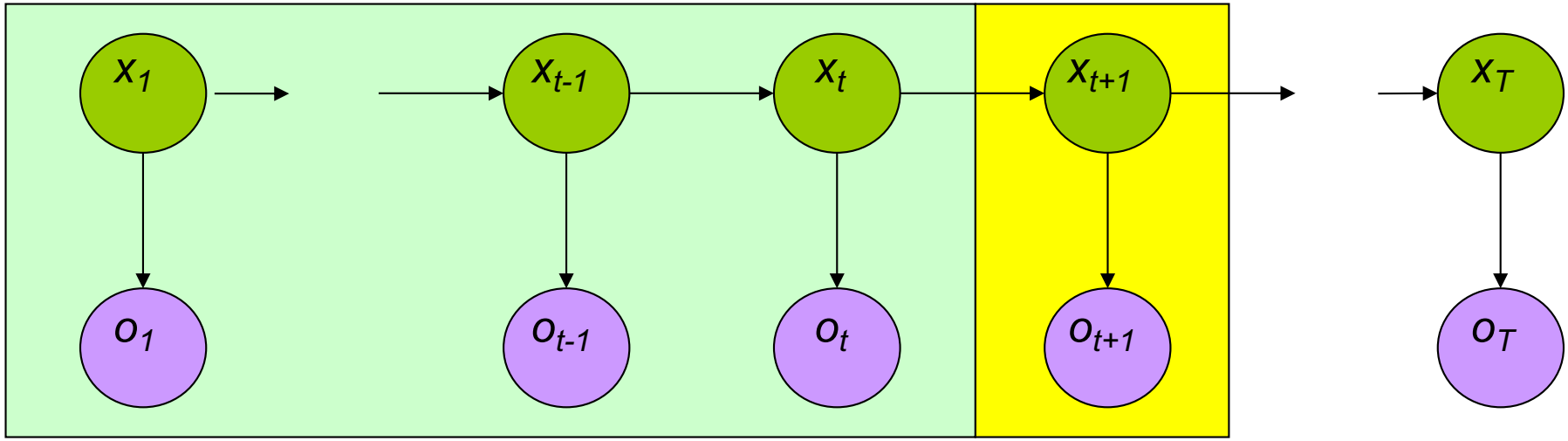
$$= P(o_1 \ldots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

chain rule

Forward Procedure



$$\alpha_j(t+1)$$

$$= P(o_1 \ldots o_{t+1}, x_{t+1} = j)$$

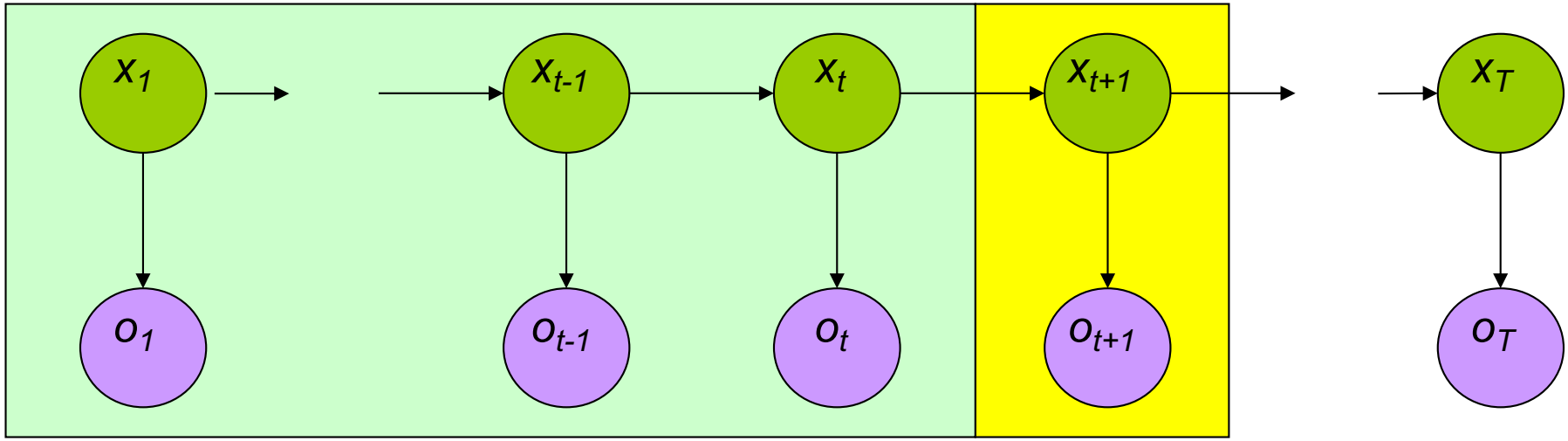
$$= P(o_1 \ldots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

chain rule

Forward Procedure



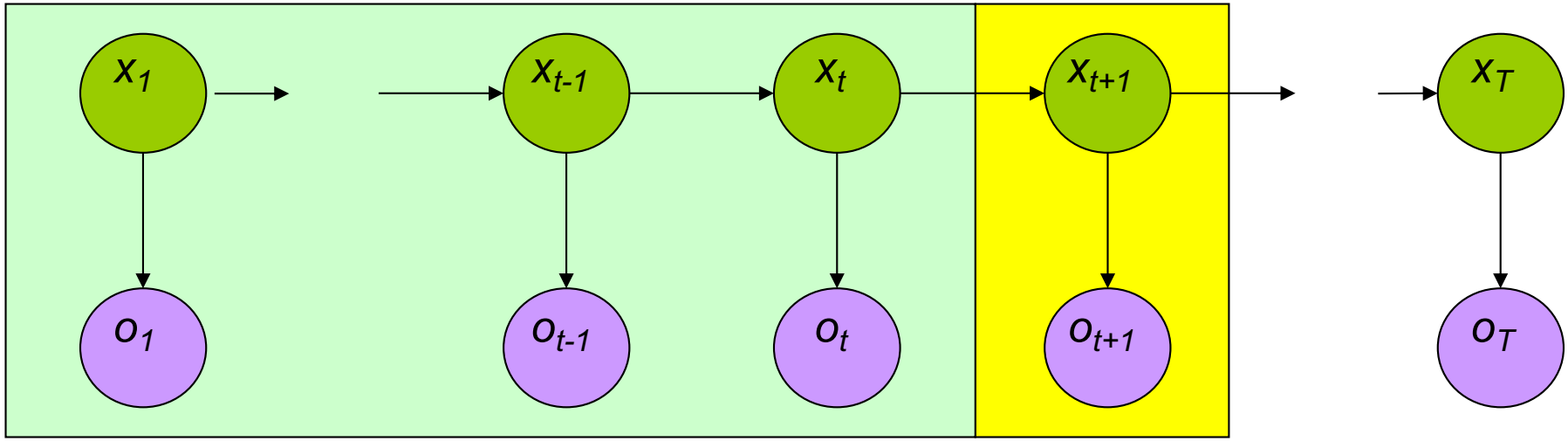
$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$ **Total prob**

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_{t+1} = j \mid x_t = i) P(x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i) P(x_{t+1} = j \mid x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} \alpha_i(t) a_{ij} b_{jo_{t+1}}$$

Forward Procedure



$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

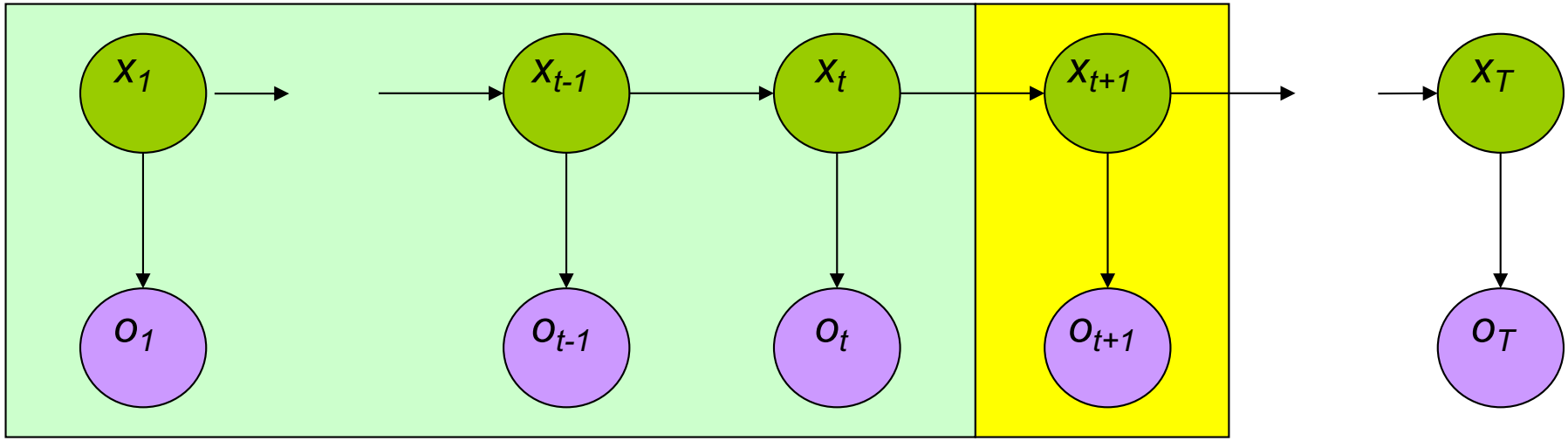
$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_{t+1} = j \mid x_t = i) P(x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

Chain rule

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i) P(x_{t+1} = j \mid x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} \alpha_i(t) a_{ij} b_{jo_{t+1}}$$

Forward Procedure



$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

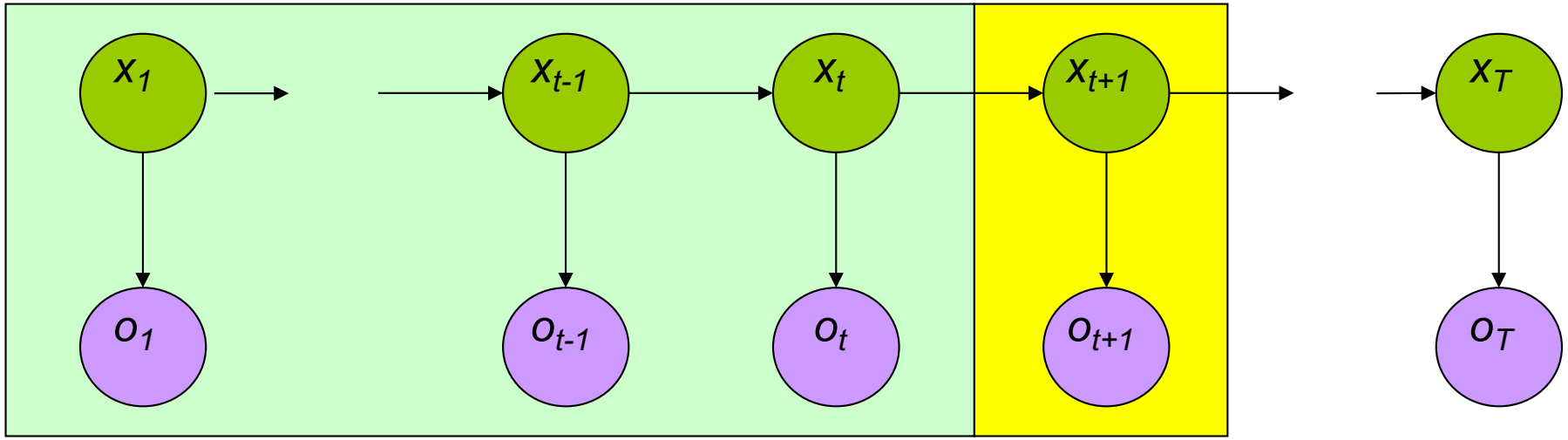
$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_{t+1} = j \mid x_t = i) P(x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i) P(x_{t+1} = j \mid x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

2x Chain rule

$$= \sum_{i=1\ldots N} \alpha_i(t) a_{ij} b_{jo_{t+1}}$$

Forward Procedure



$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_{t+1} = j \mid x_t = i) P(x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i) P(x_{t+1} = j \mid x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} \alpha_i(t) a_{ij} b_{jo_{t+1}}$$

Definition + given parameters

Dynamic Programming

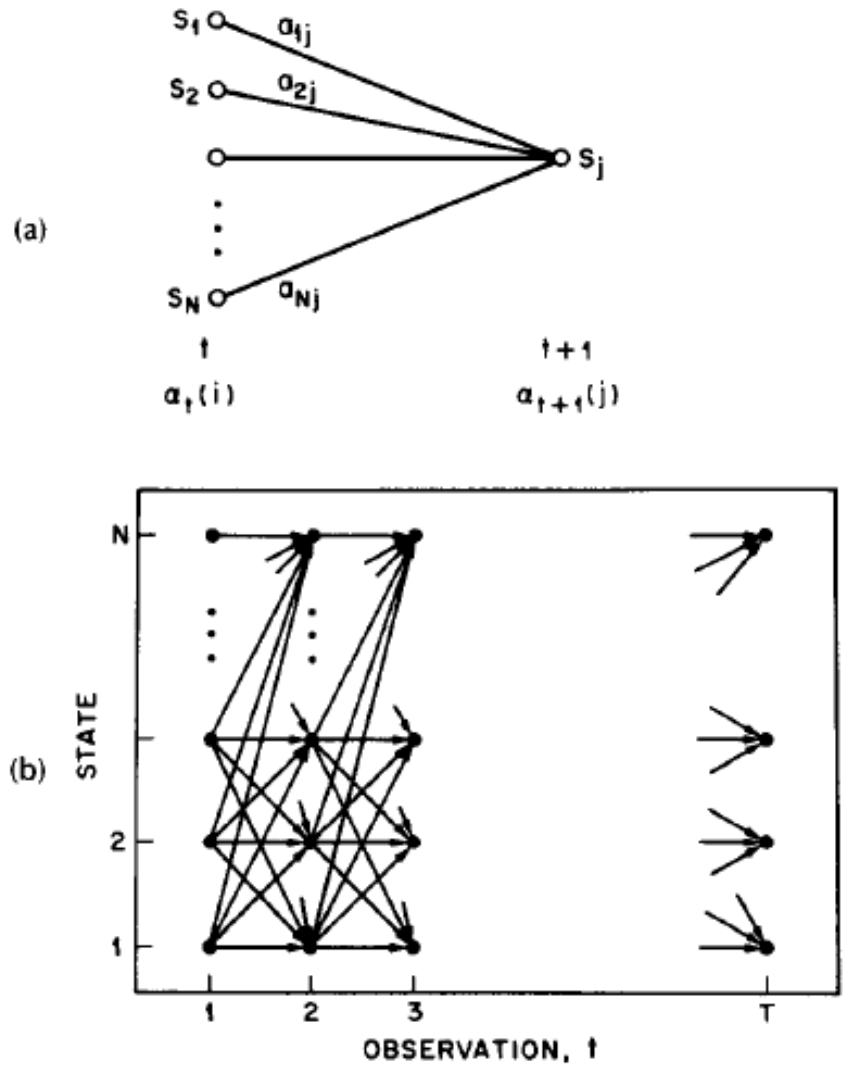


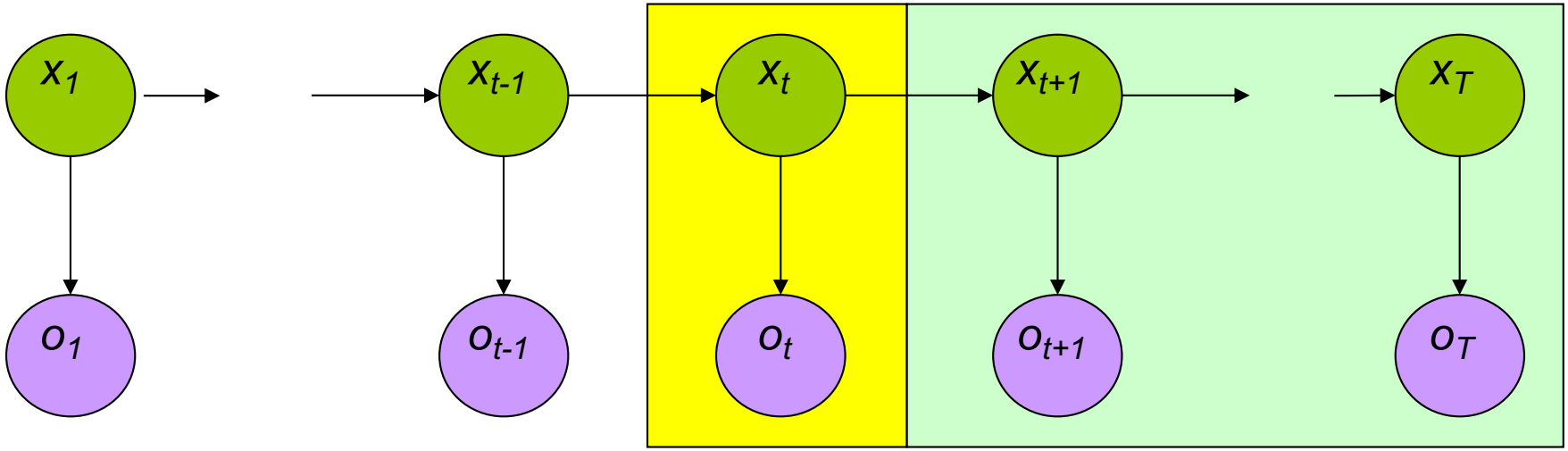
Fig. 4. (a) Illustration of the sequence of operations required for the computation of the forward variable $\alpha_{t+1}(j)$. (b) Implementation of the computation of $\alpha_t(i)$ in terms of a lattice of observations t, and states i.

$$\alpha_j(t+1) = \sum_{i=1\ldots N} \alpha_i(t) a_{ij} b_{jo_{t+1}}$$

Complexity $O(N^2.T)$

$E.g.\ N = 5, T = 100$ gives ≈ 3000

Similar, Backward Procedure



$$\beta_i(T) = 1$$

$$\beta_i(t) = P(o_{t+1} \ldots o_T \mid x_t = i)$$

$$\beta_i(t) = \sum_{j=1 \ldots N} a_{ij} b_{io_{t+1}} \beta_j(t+1)$$

Probability of the rest of the states given the first state

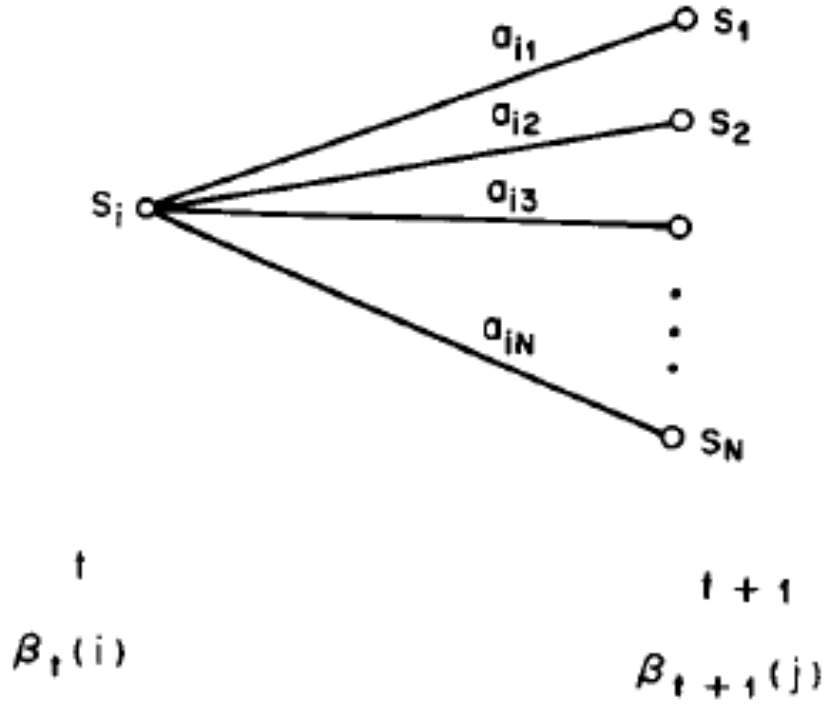
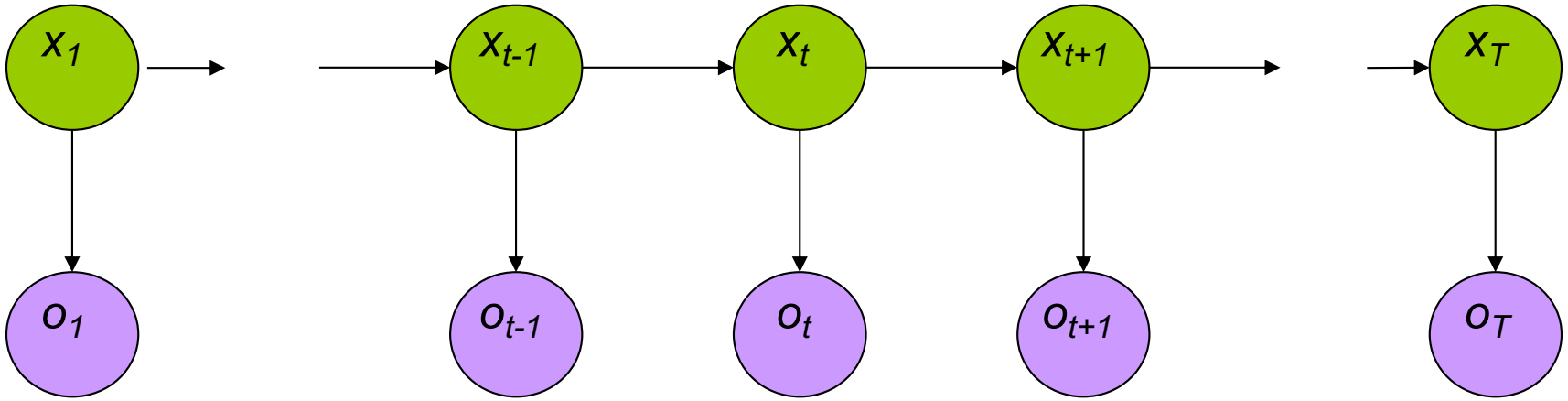


Fig. 5. Illustration of the sequence of operations required for the computation of the backward variable $\beta_t(i)$.

(1) Decoding: Solution



$$P(O \mid \mu) = \sum_{i=1}^{N} \alpha_i(T)$$

Forward Procedure

$$P(O \mid \mu) = \sum_{i=1}^{N} \pi_i \beta_i(1)$$

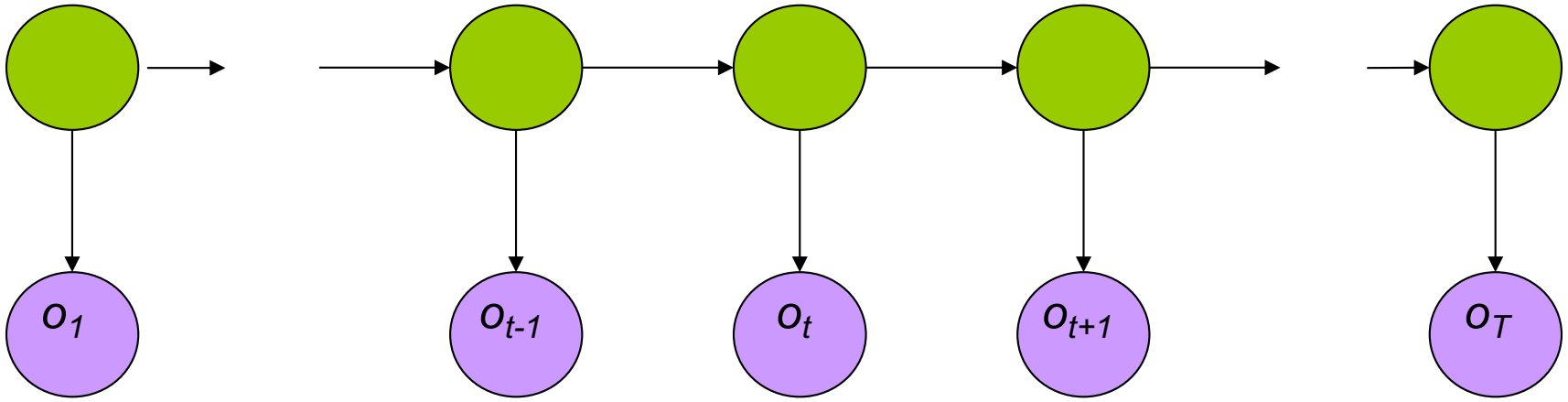
Backward Procedure

$$P(O \mid \mu) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t)$$

Combination

Important for (2) and (3)

(2) Best State Sequence



- **Find the state sequence that best explains the observations**

- **Two approaches**
  - **Individually most likely states**
  - **Most likely sequence (Viterbi)**

$$\arg\max_X P(X \mid O)$$

(2) Best State Sequence

$$\gamma_i(t) = P(X_t = i \mid O, \mu)$$ target

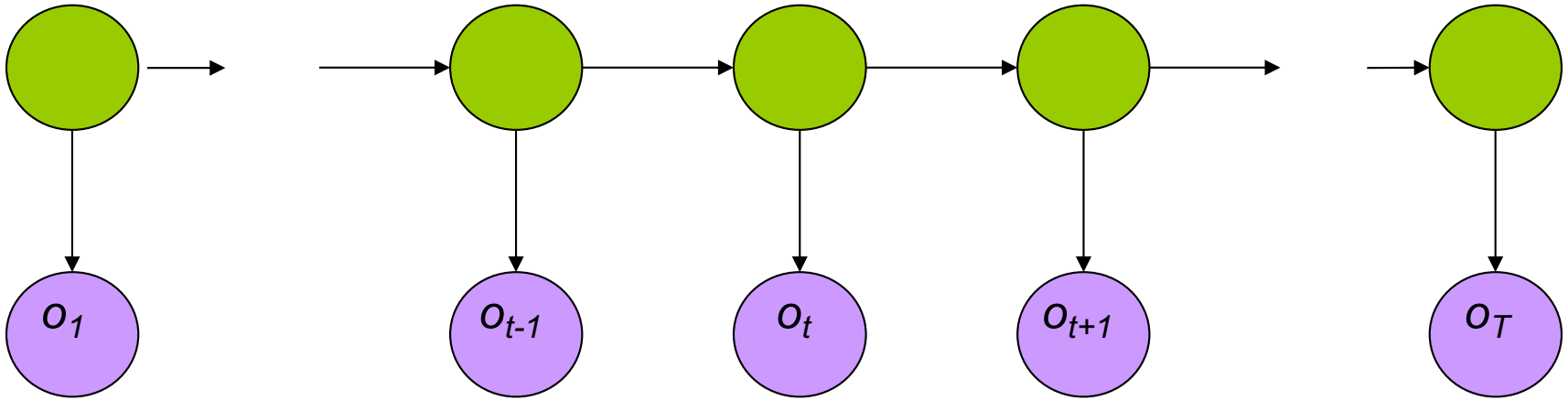
$$= \frac{P(X_t = i, O \mid \mu)}{P(O \mid \mu)}$$ definition of conditional prob

$$= \frac{\alpha_i(t).\beta_i(t)}{\sum_{j=1}^{n} \alpha_j(t).\beta_j(t)}$$ use what we know already

Most likely state at each point in time

$$\hat{X}_t = \arg\max \gamma_i(t)$$

(2) Best State Sequence

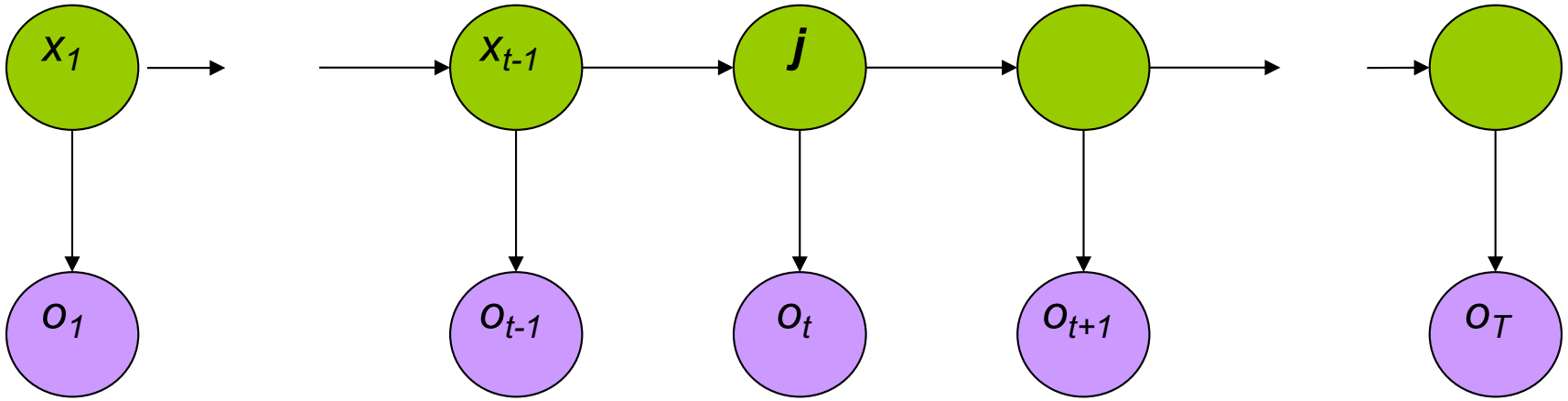


- Find the state sequence that best explains the observations, i.e., MPE

- **Viterbi algorithm**

$$\arg\max_X P(X \mid O)$$

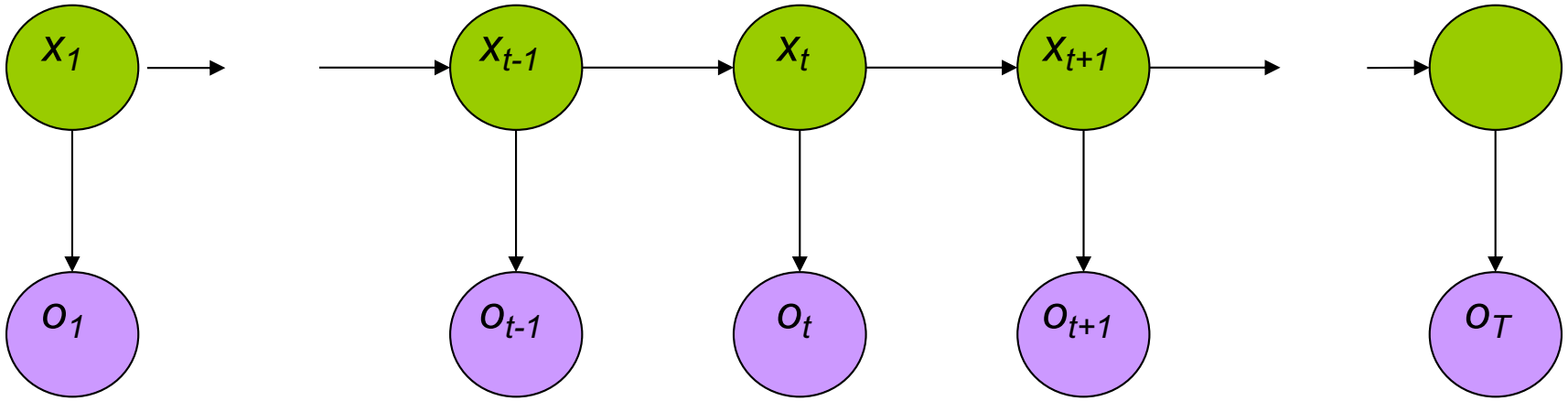
Viterbi Algorithm



$$\delta_j(t) = \max_{x_1 \ldots x_{t-1}} P(x_1 \ldots x_{t-1}, o_1 \ldots o_{t-1}, x_t = j, o_t)$$

The state sequence which maximizes the probability of seeing the observations to time t-1, landing in state j, and seeing the observation at time t

Viterbi Algorithm



$$\delta_j(t) = \max_{x_1 \ldots x_{t-1}} P(x_1 \ldots x_{t-1}, o_1 \ldots o_{t-1}, x_t = j, o_t)$$

$$\delta_j(t+1) = \max_i \delta_i(t) a_{ij} b_{jo_{t+1}}$$

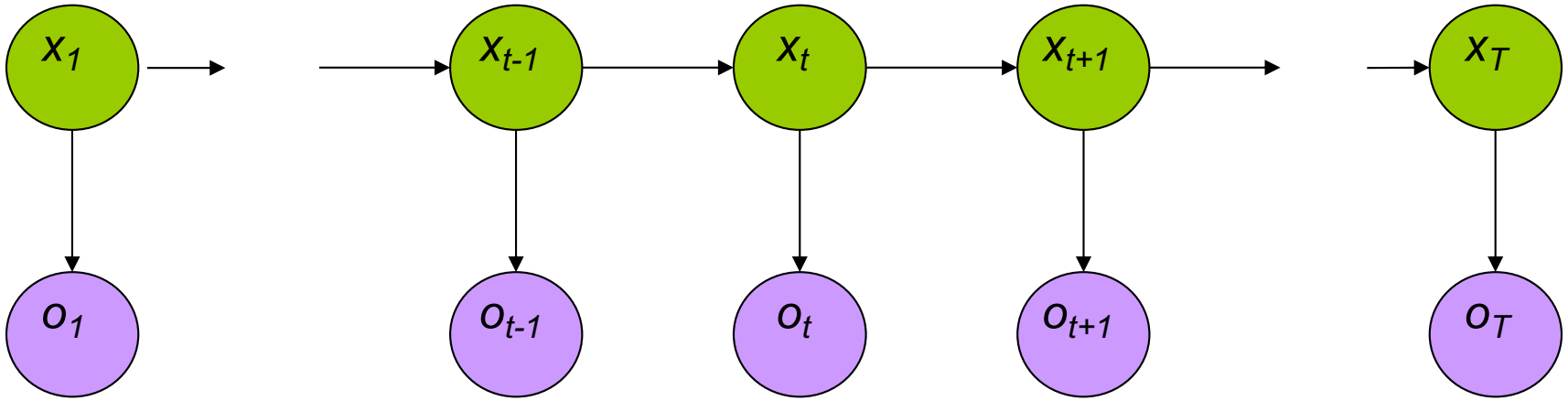
$$\psi_j(t+1) = \arg\max_i \delta_i(t) a_{ij} b_{jo_{t+1}}$$

Initialization

$$\delta_1(i) = \pi_i b_{io_1}$$

$$\psi_1(i) = 0$$

Viterbi Algorithm



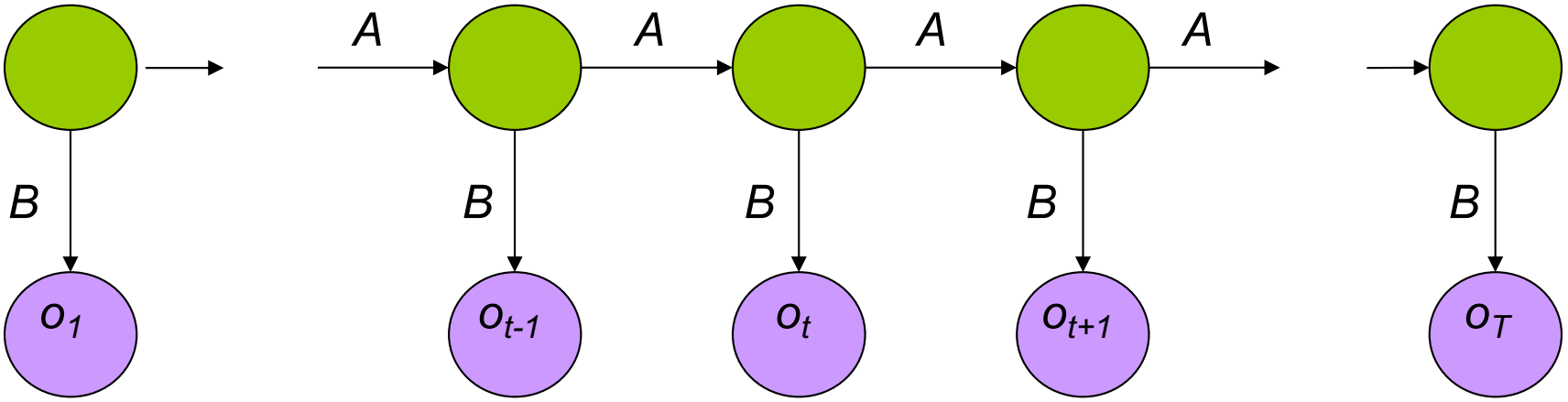
$$\hat{X}_T = \arg\max_i \delta_i(T)$$

$$\hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1)$$

$$P(\hat{X}) = \arg\max_i \delta_i(T)$$

Compute the most likely state sequence by working backwards

Parameter Estimation using EM



- Given an observation sequence, find the model that is most likely to produce that sequence.
- No analytic method $\arg\max_{\mu} P(O_{training} \mid \mu)$
- Given a model and observation sequence, update the model parameters to better fit the observations.

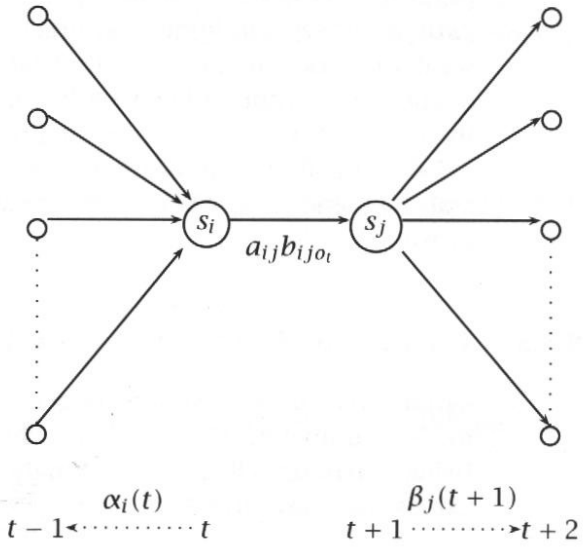


Figure 9.7 The probability of traversing an arc. Given an observation sequence and a model, we can work out the probability that the Markov process went from state s_i to s_j at time t.

$$\alpha_i(t) = P(o_1 ... o_t, x_t = i \mid \mu)$$

$$\beta_i(t) = P(o_{t+1} ... o_T \mid x_t = i)$$

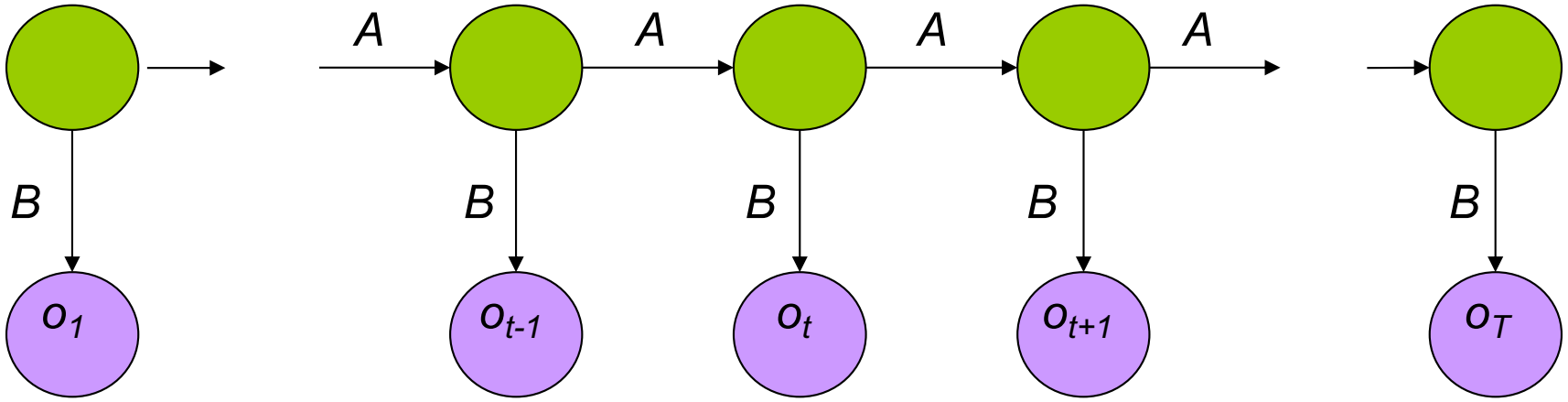
$$P(O \mid \mu) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t)$$

Probability of making a transition from i to j at time t

$$p_t(i,j) = P(X_t = i, X_{t+1} = j \mid O, \mu) \quad \text{target}$$

$$= \frac{P(X_t = i, X_{t+1} = j, O \mid \mu)}{P(O \mid \mu)} \quad \text{def of cond prob}$$

Parameter Estimation



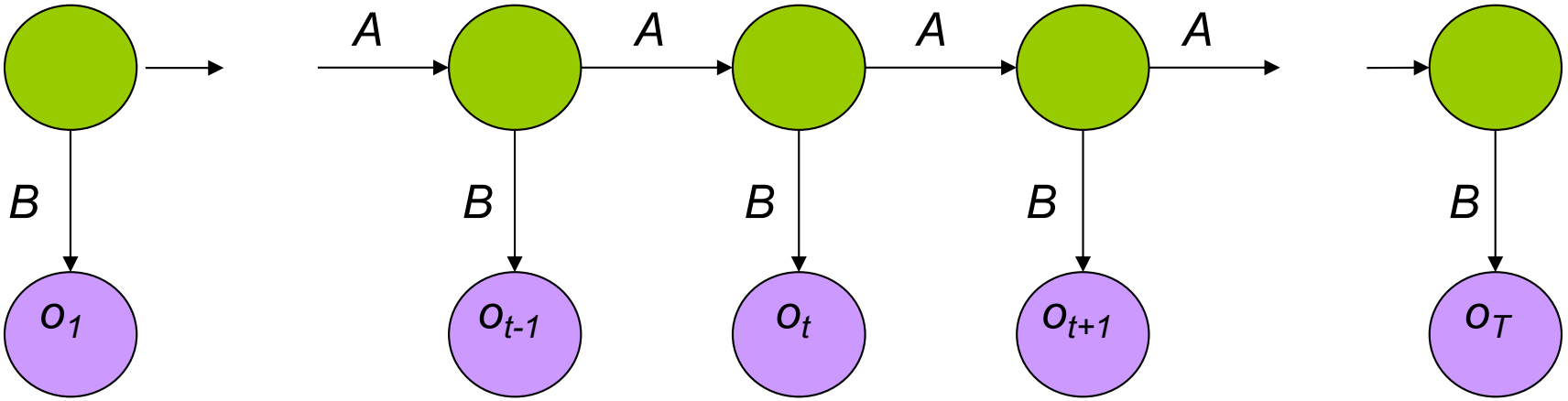
$$p_t(i,j) = \frac{\alpha_i(t) a_{ij} b_{jo_{t+1}} \beta_j(t+1)}{\sum_{m=1\ldots N} \alpha_m(t)\beta_m(t)}$$

Probability of traversing an arc

$$\gamma_i(t) = \sum_{j=1\ldots N} p_t(i,j)$$

Probability of being in state *i*

Parameter Estimation



expected counts

$$\hat{\pi}_i = \gamma_i(1)$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} p_t(i,j)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$\hat{b}_{ik} = \frac{\sum_{\{t:o_t=k\}} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

Now we can compute the new estimates of the model parameters.

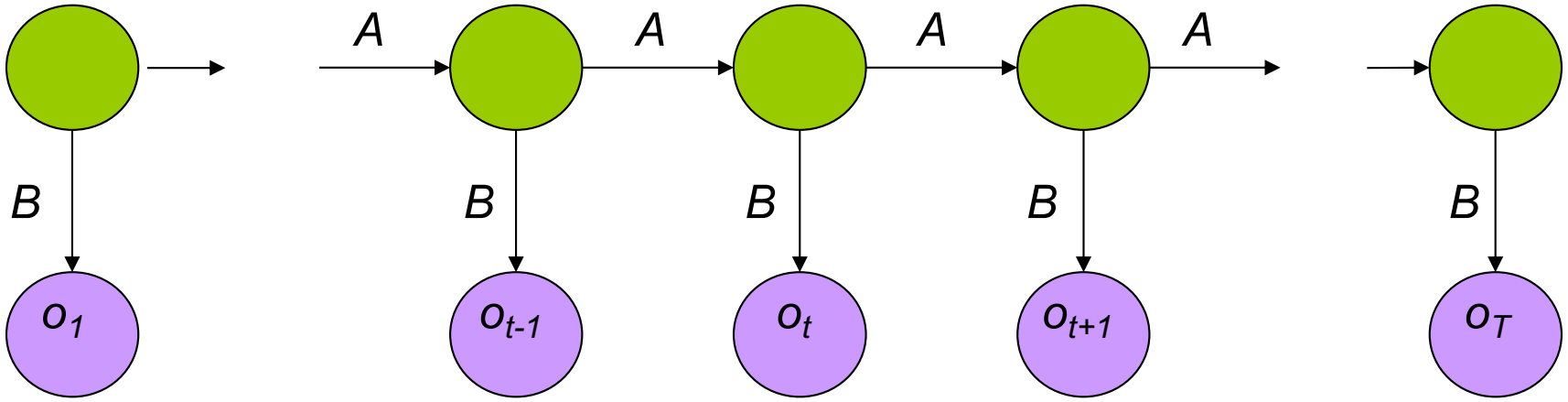
Instance of EM

- We have that

$$P(O \mid \hat{\mu}) \geq P(O \mid \mu)$$

- We may get stuck in local maximum (or even saddle point)
- Nevertheless, Baum-Welch usually effective

The Most Important Thing



We can use the special structure of this model to do a lot of neat math and solve problems that are otherwise not solvable.

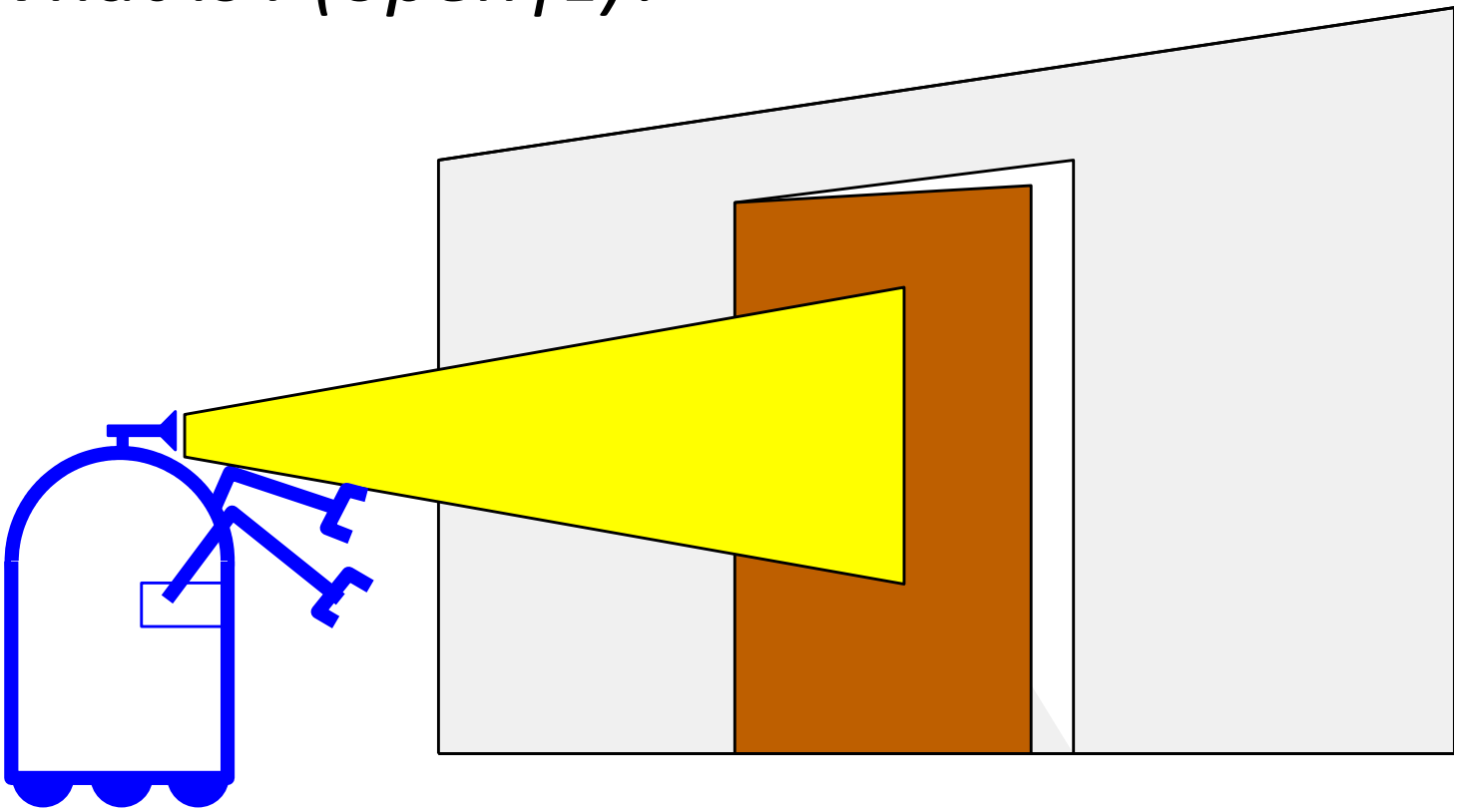
Kalman Filers

- What about continuous states as e.g. encountered in robotics?

- The Kalman filter is a recursive filter that estimates the state of a linear dynamic system from a series of noisy measurements

Simple Example of State Estimation

- Suppose a robot obtains measurement z
- What is *P(open|z)?*



Example

- $P(z|open) = 0.6$ $\quad P(z|\neg open) = 0.3$
- $P(open) = P(\neg open) = 0.5$

$$P(open \mid z) = \frac{P(z \mid open)P(open)}{P(z \mid open)p(open) + P(z \mid \neg open)p(\neg open)}$$

$$P(open \mid z) = \frac{0.6 \cdot 0.5}{0.6 \cdot 0.5 + 0.3 \cdot 0.5} = \frac{2}{3} = 0.67$$

***Observing z* raises the probability that the door is open.**

Combining Evidence

- Suppose our robot obtains another observation z_2.
- How can we integrate this new information?
- More generally, how can we estimate $P(x| z_1...z_n)$?

Recursive Bayesian Updating

$$P(x \mid z_1, \ldots, z_n) = \frac{P(z_n \mid x, z_1, \ldots, z_{n-1})\, P(x \mid z_1, \ldots, z_{n-1})}{P(z_n \mid z_1, \ldots, z_{n-1})}$$

Markov assumption: z_n is independent of $z_1,\ldots,z_{n-1}$ if we know x

$$P(x \mid z_1, \ldots, z_n) = \frac{P(z_n \mid x)\, P(x \mid z_1, \ldots, z_{n-1})}{P(z_n \mid z_1, \ldots, z_{n-1})}$$

$$= \eta\, P(z_n \mid x)\, P(x \mid z_1, \ldots, z_{n-1})$$

$$= \eta_{1\ldots n} \prod_{i=1\ldots n} P(z_i \mid x)\, P(x)$$

Example: Second Measurement

TECHNISCHE UNIVERSITÄT DARMSTADT

- $P(z_2|open) = 0.5$ $\quad\quad$ $P(z_2|\neg open) = 0.6$
- $P(open|z_1)=2/3$

$$P(open \mid z_2, z_1) = \frac{P(z_2 \mid open)\, P(open \mid z_1)}{P(z_2 \mid open)\, P(open \mid z_1) + P(z_2 \mid \neg open)\, P(\neg open \mid z_1)}$$

$$= \frac{\frac{1}{2} \cdot \frac{2}{3}}{\frac{1}{2} \cdot \frac{2}{3} + \frac{3}{5} \cdot \frac{1}{3}} = \frac{5}{8} = 0.625$$

z_2 lowers the probability that the door is open.

Actions

- As you know already, often the world is **dynamic** since
  - **actions carried out by the robot**,
  - **actions carried out by other agents**,
  - or just the **time** is passing by

  change the world.

- How can we **incorporate** such **actions**?

Typical Actions for a Robot

- The robot **turns its wheels** to move
- The robot **uses its manipulator** to grasp an object
- Plants grow over **time**…

- Actions are **never carried out with absolute certainty**.
- In contrast to measurements, **actions generally increase the uncertainty**.

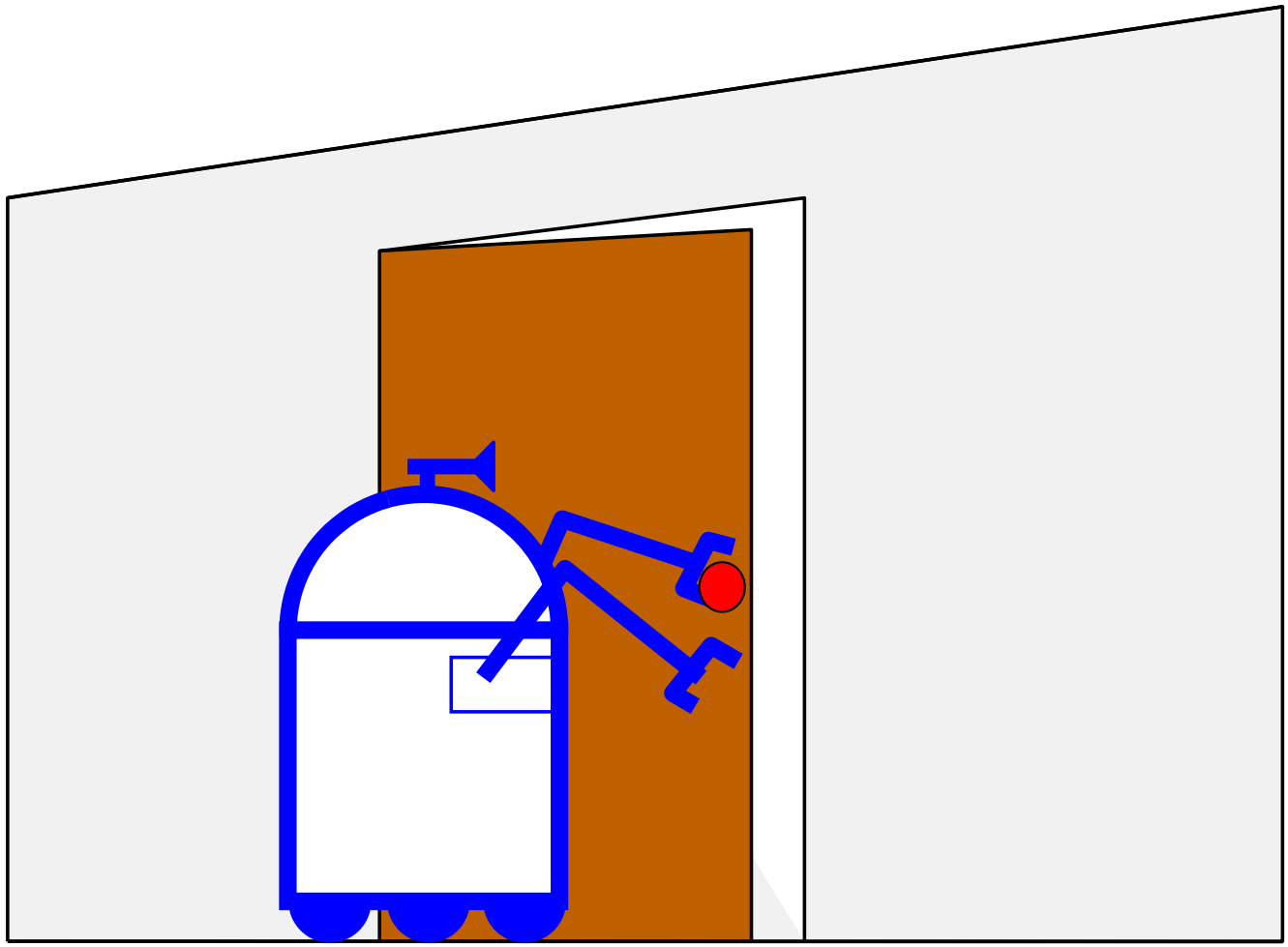
Modeling Action

- To incorporate the outcome of an action *u* into the current "belief", we use the conditional pdf

$$P(x|u,x')$$

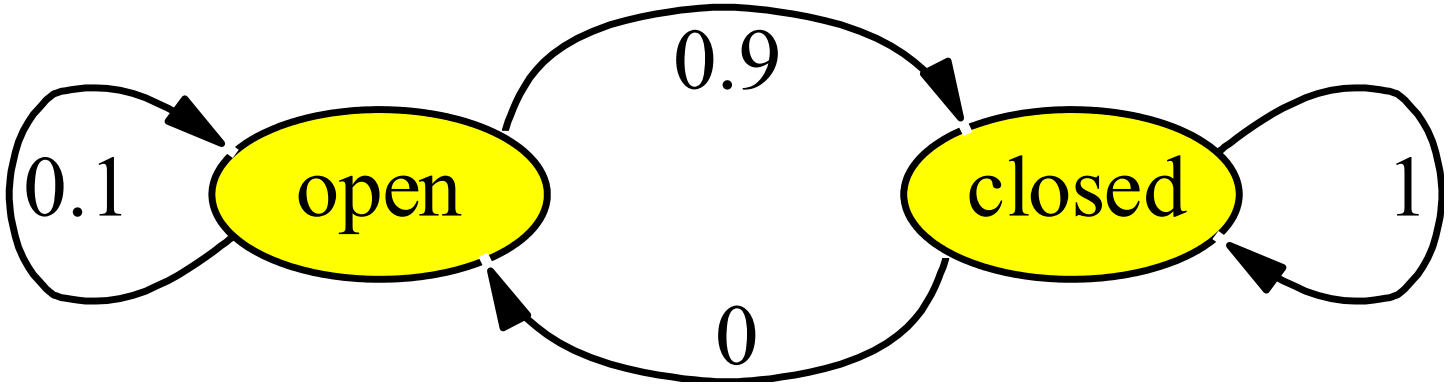
- This term specifies the pdf that **executing *u* changes the state from *x' to x***.

Example: Closing the door



State Transitions

P(x|u,x') for *u* = "close door":



If the door is open, the action "close door" succeeds in 90% of all cases.

Integrating the Outcome of Actions

Continuous case:

$$P(x \mid u) = \int P(x \mid u, x') P(x') dx'$$

Discrete case:

$$P(x \mid u) = \sum_{x'} P(x \mid u, x') P(x')$$

Example: The Resulting Belief

$$
\begin{aligned}
P(closed \mid u) &= \sum P(closed \mid u, x')P(x') \\
&= P(closed \mid u, open)P(open) \\
&\quad + P(closed \mid u, closed)P(closed) \\
&= \frac{9}{10} * \frac{5}{8} + \frac{1}{1} * \frac{3}{8} = \frac{15}{16} \\
P(open \mid u) &= \sum P(open \mid u, x')P(x') \\
&= P(open \mid u, open)P(open) \\
&\quad + P(open \mid u, closed)P(closed) \\
&= \frac{1}{10} * \frac{5}{8} + \frac{0}{1} * \frac{3}{8} = \frac{1}{16} \\
&= 1 - P(closed \mid u)
\end{aligned}
$$

Bayes Filters: Framework

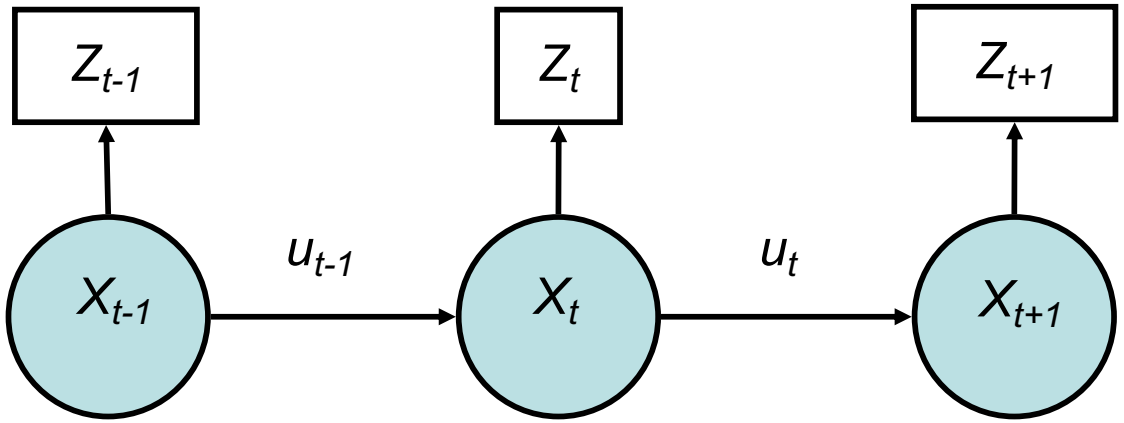
- **Given:**
  - Stream of observations *z* and action data *u:*

$$d_t = \{u_1, z_2 \ldots, u_{t-1}, z_t\}$$

  - Sensor model *P(z|x).*
  - Action model *P(x|u,x').*
  - Prior probability of the system state *P(x).*
- **Wanted:**
  - Estimate of the state *X* of a dynamical system.
  - The posterior of the state is also called **Belief**:

$$Bel(x_t) = P(x_t \mid u_1, z_2 \ldots, u_{t-1}, z_t)$$

Markov Assumption



$$p(d_t, d_{t-1}, \ldots, d_0 \mid x_t, d_{t+1}, d_{t+2}, \ldots) = p(d_t, d_{t-1}, \ldots, d_0 \mid x_t)$$

$$p(d_t, d_{t+1}, \ldots \mid x_t, d_1, d_2, \ldots, d_{t-1}) = p(d_t, d_{t+1}, \ldots \mid x_t)$$

$$p(x_t \mid u_{t-1}, x_{t-1}, d_{t-2}, \ldots, d_0) = p(x_t \mid u_{t-1}, x_{t-1})$$

Underlying Assumptions

- Static world; Independent noise; Perfect model, no approximation errors

Bayes Filters

z = observation
u = action
x = state

$$Bel(x_t) = P(x_t \mid u_1, z_2 \ldots, u_{t-1}, z_t)$$

Bayes
$$= \eta\ P(z_t \mid x_t, u_1, z_2, \ldots, u_{t-1})\ P(x_t \mid u_1, z_2, \ldots, u_{t-1})$$

Markov
$$= \eta\ P(z_t \mid x_t)\ P(x_t \mid u_1, z_2, \ldots, u_{t-1})$$

Total prob.
$$= \eta\ P(z_t \mid x_t) \int P(x_t \mid u_1, z_2, \ldots, u_{t-1}, x_{t-1})$$
$$P(x_{t-1} \mid u_1, z_2, \ldots, u_{t-1})\ dx_{t-1}$$

Markov
$$= \eta\ P(z_t \mid x_t) \int P(x_t \mid u_{t-1}, x_{t-1})\ P(x_{t-1} \mid u_1, z_2, \ldots, u_{t-1})\ dx_{t-1}$$

$$= \eta\ P(z_t \mid x_t) \int P(x_t \mid u_{t-1}, x_{t-1})\ Bel(x_{t-1})\ dx_{t-1}$$

$$Bel(x_t) = \eta\, P(z_t \mid x_t) \int P(x_t \mid u_{t-1}, x_{t-1})\, Bel(x_{t-1})\, dx_{t-1}$$

1. Algorithm **Bayes_filter**(*Bel(x),d*):
2. η=0
3. if *d* is a perceptual data item *z* then
4. For all *x* do
5. $Bel'(x) = P(z \mid x) Bel(x)$
6. $\eta = \eta + Bel'(x)$
7. For all *x* do
8. $Bel'(x) = \eta^{-1} Bel'(x)$
9. else if *d* is an action data item *u* then
10. For all *x* do
11. $Bel'(x) = \int P(x \mid u, x')\, Bel(x')\, dx'$
12. return *Bel'(x)*

Bayes Filters are Familiar!

$$Bel(x_t) = \eta\, P(z_t \mid x_t) \int P(x_t \mid u_{t-1}, x_{t-1})\, Bel(x_{t-1})\, dx_{t-1}$$

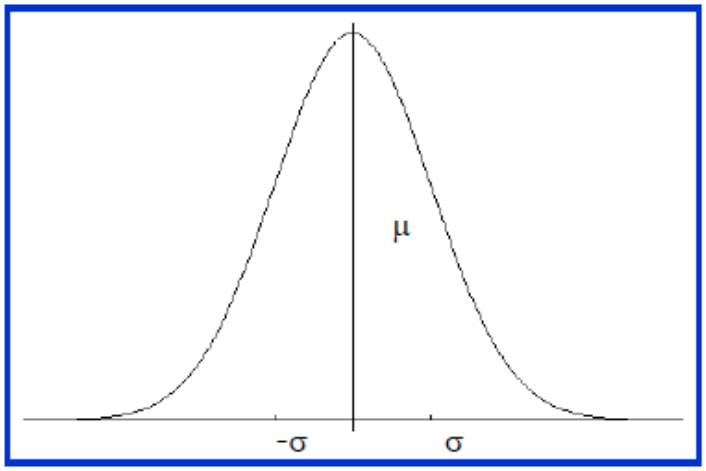
- Kalman filters
- Particle filters
- Hidden Markov models
- Dynamic Bayes networks (later this class)
- Partially Observable Markov Decision Processes (POMDPs; maybe we touch them later this class)

Gaussian

$$p(x) \sim N(\mu, \sigma^2):$$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

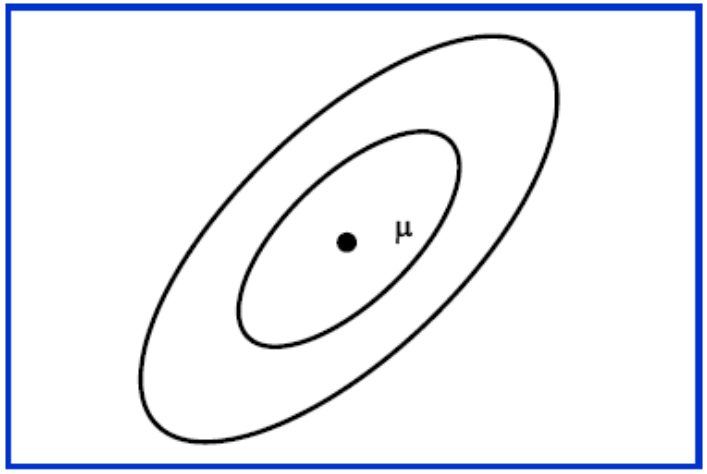
Univariate



$$p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}):$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

Multivariate



Propertis of Gaussians

$$\left.\begin{array}{l} X \sim N(\mu, \Sigma) \\ Y = AX + B \end{array}\right\} \Rightarrow \quad Y \sim N(A\mu + B, A\Sigma A^T)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \Sigma_1) \\ X_2 \sim N(\mu_2, \Sigma_2) \end{array}\right\} \Rightarrow p(X_1) \cdot p(X_2) \sim N\left(\frac{\Sigma_2}{\Sigma_1 + \Sigma_2}\mu_1 + \frac{\Sigma_1}{\Sigma_1 + \Sigma_2}\mu_2, \quad \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}}\right)$$

- We stay in the "Gaussian world" as long as we start with Gaussians and perform only linear transformations.

Discrete Time Kalman Filter

Estimates the state x of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

with a measurement

$$z_t = C_t x_t + \delta_t$$

Components of a Kalman Filter

A_t Matrix (nxn) that describes how the state evolves from t to t-1 without controls or noise.

B_t Matrix (nxl) that describes how the control u_t changes the state from t to t-1.

C_t Matrix (kxn) that describes how to map the state x_t to an observation z_t.

ε_t δ_t Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance R_t and Q_t respectively.

Initialization

- Initial belief is normally distributed:

$$bel(x_0) = N(x_0; \mu_0, \Sigma_0)$$

Dynamics

- Dynamics are linear function of state and control plus additive noise:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

$$p(x_t \mid u_t, x_{t-1}) = N\left(x_t; A_t x_{t-1} + B_t u_t, R_t\right)$$

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}) \; bel(x_{t-1})\, dx_{t-1}$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$\sim N\left(x_t; A_t x_{t-1} + B_t u_t, R_t\right) \quad \sim N\left(x_{t-1}; \mu_{t-1}, \Sigma_{t-1}\right)$$

Belief Update: Prediction

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}) \quad bel(x_{t-1})\, dx_{t-1}$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$\sim N(x_t; A_t x_{t-1} + B_t u_t, R_t) \quad \sim N(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})$$

$$\Downarrow$$

$$\overline{bel}(x_t) = \eta \int \exp\left\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1} (x_t - A_t x_{t-1} - B_t u_t)\right\}$$

$$\exp\left\{-\frac{1}{2}(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1} (x_{t-1} - \mu_{t-1})\right\} dx_{t-1}$$

$$\overline{bel}(x_t) = \begin{cases} \overline{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \end{cases}$$

Observations

- Observations are linear function of state plus additive noise:

$$z_t = C_t x_t + \delta_t$$

$$p(z_t \mid x_t) = N(z_t; C_t x_t, Q_t)$$

$$bel(x_t) = \eta \; p(z_t \mid x_t) \; \overline{bel}(x_t)$$

$$\Downarrow \qquad \Downarrow$$

$$\sim N(z_t; C_t x_t, Q_t) \qquad \sim N(x_t; \overline{\mu}_t, \overline{\Sigma}_t)$$

Belief Update: Correction

$$bel(x_t) = \eta \; p(z_t \mid x_t) \; \overline{bel}(x_t)$$

$$\Downarrow \qquad \Downarrow$$

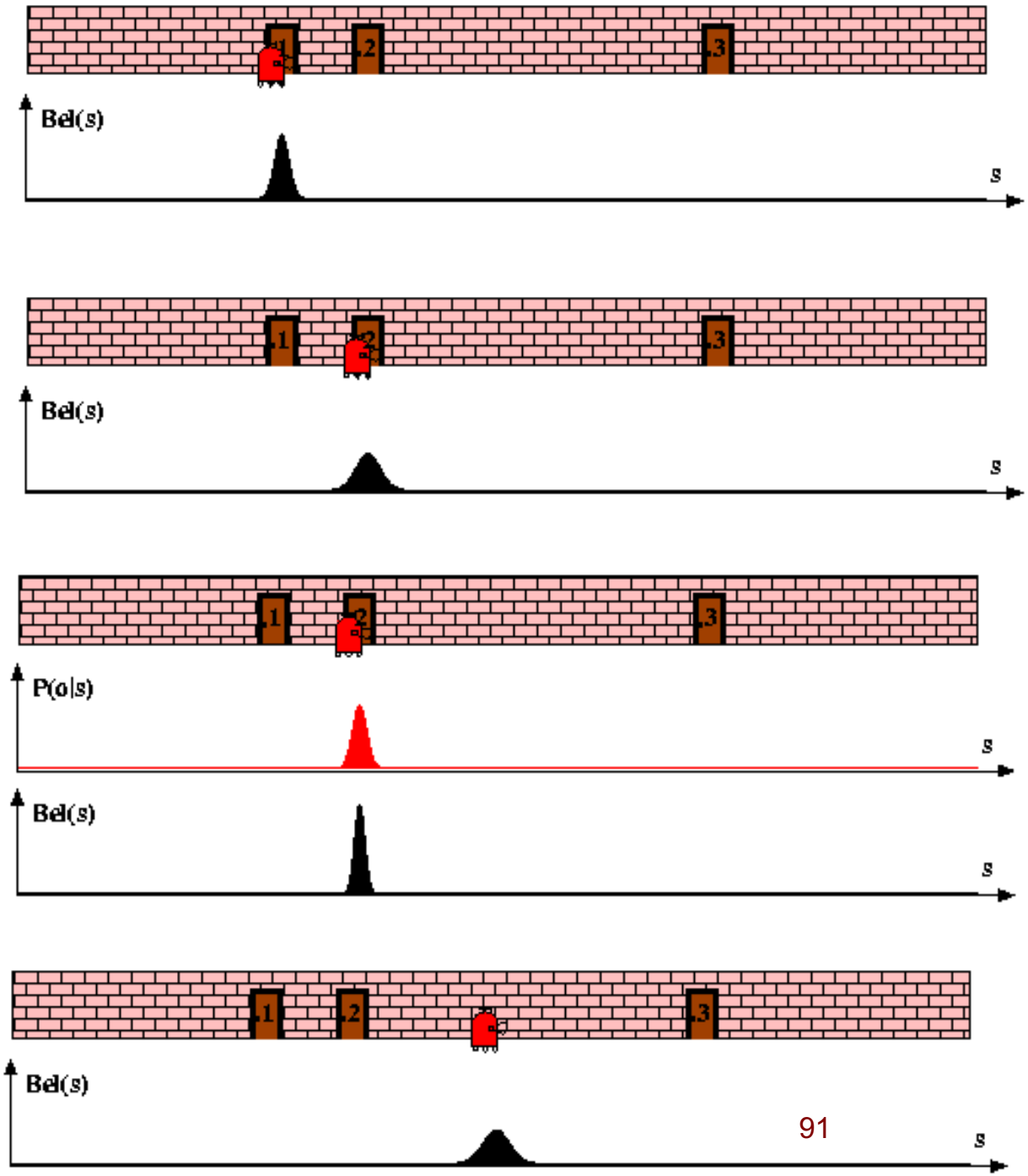
$$\sim N(z_t; C_t x_t, Q_t) \qquad \sim N(x_t; \overline{\mu}_t, \overline{\Sigma}_t)$$

$$\Downarrow$$

$$bel(x_t) = \eta \exp\left\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)\right\} \exp\left\{-\frac{1}{2}(x_t - \overline{\mu}_t)^T \overline{\Sigma}_t^{-1}(x_t - \overline{\mu}_t)\right\}$$

$$bel(x_t) = \begin{cases} \mu_t = \overline{\mu}_t + K_t(z_t - C_t\overline{\mu}_t) \\ \Sigma_t = (I - K_t C_t)\overline{\Sigma}_t \end{cases} \quad \text{with} \quad K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}$$

Kalman Filters

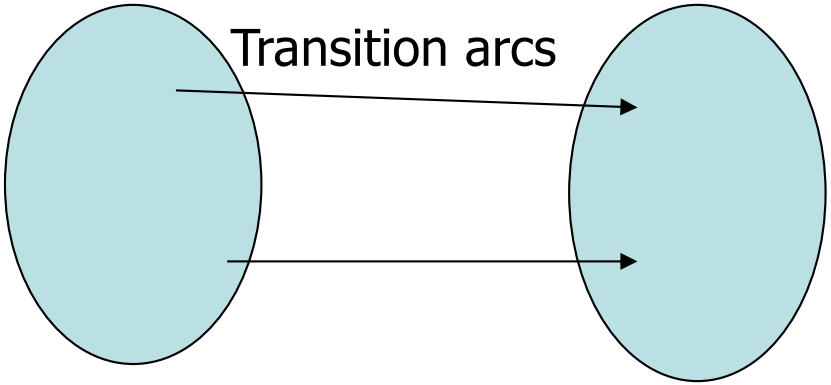


So far

- Sometimes, belief states admit compact representations & manipulations:
  - E.g. Kalman filters (assuming Gaussian processes)

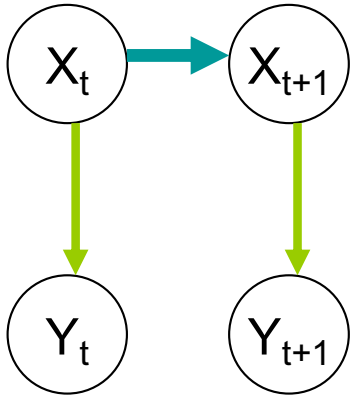
- What about Dynamic Bayesian Networks, i.e., the general case ?

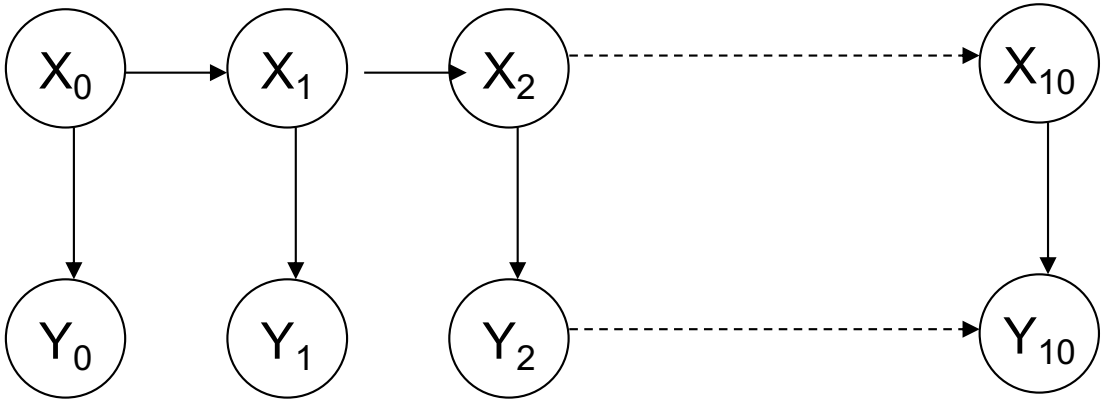
Dynamic Bayesian Networks (DBNs)



Arbitrary Bayesian Network at time t

The same Bayesian Network at time t+1





Unrolled DBN for t=0 to t=10

Types of Inference Tasks

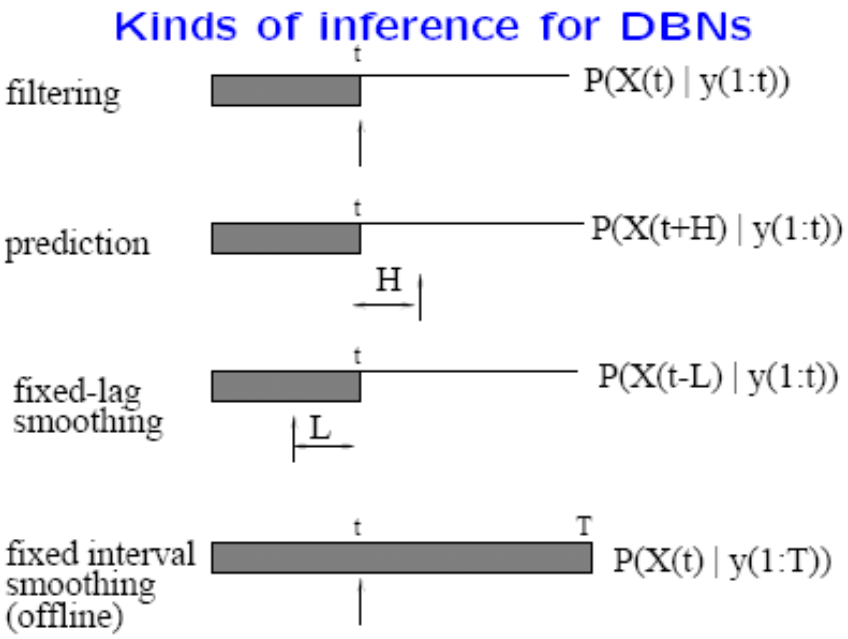
TECHNISCHE UNIVERSITÄT DARMSTADT

Filtering: $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$
belief state—input to the decision process of a rational agent

Prediction: $\mathbf{P}(\mathbf{X}_{t+k}|\mathbf{e}_{1:t})$ for $k > 0$
evaluation of possible action sequences;
like filtering without the evidence

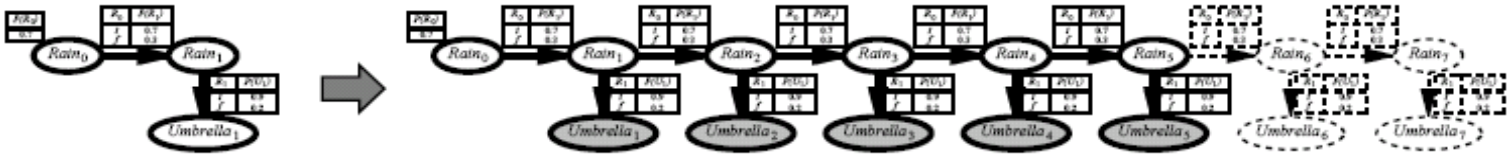
Smoothing: $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ for $0 \leq k < t$
better estimate of past states, essential for learning

Most likely explanation: $\arg\max_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t}|\mathbf{e}_{1:t})$
speech recognition, decoding with a noisy channel



Exact inference in DBNs

Naive method: unroll the network and run any exact algorithm



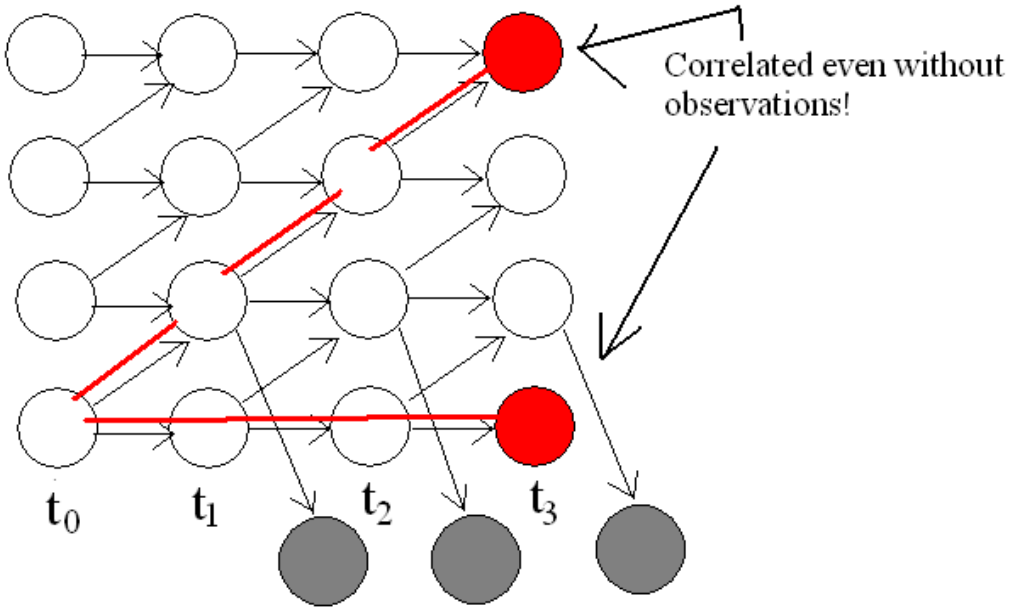
Problem: inference cost for each update grows with t

Rollup filtering: add slice $t+1$, "sum out" slice t using variable elimination

Largest factor is $O(d^{n+1})$, update cost $O(d^{n+2})$
(cf. HMM update cost $O(d^{2n})$)

Problem

- Bayesian Network: a decomposed structure to represent the full joint distribution and to do efficient inference
- Does it imply easy decomposition for the *belief state*?



- No!

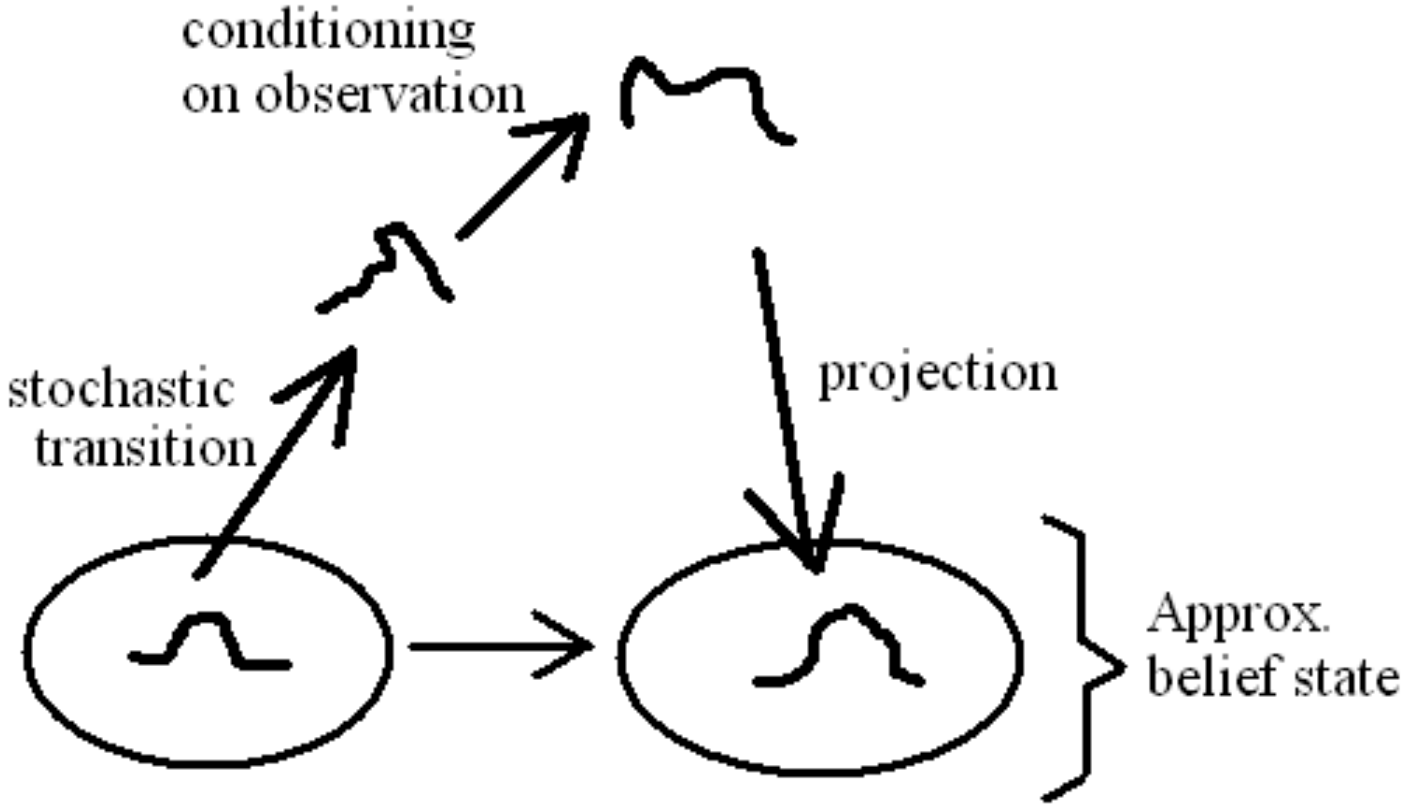
Tractable, approximate representation

- Exact inference in DBN is intractable
- Need approximation
  - Maintain an *approximate belief state*
- E.g. Factored belief state

Assumed Density Filtering: Idea

TECHNISCHE UNIVERSITÄT DARMSTADT

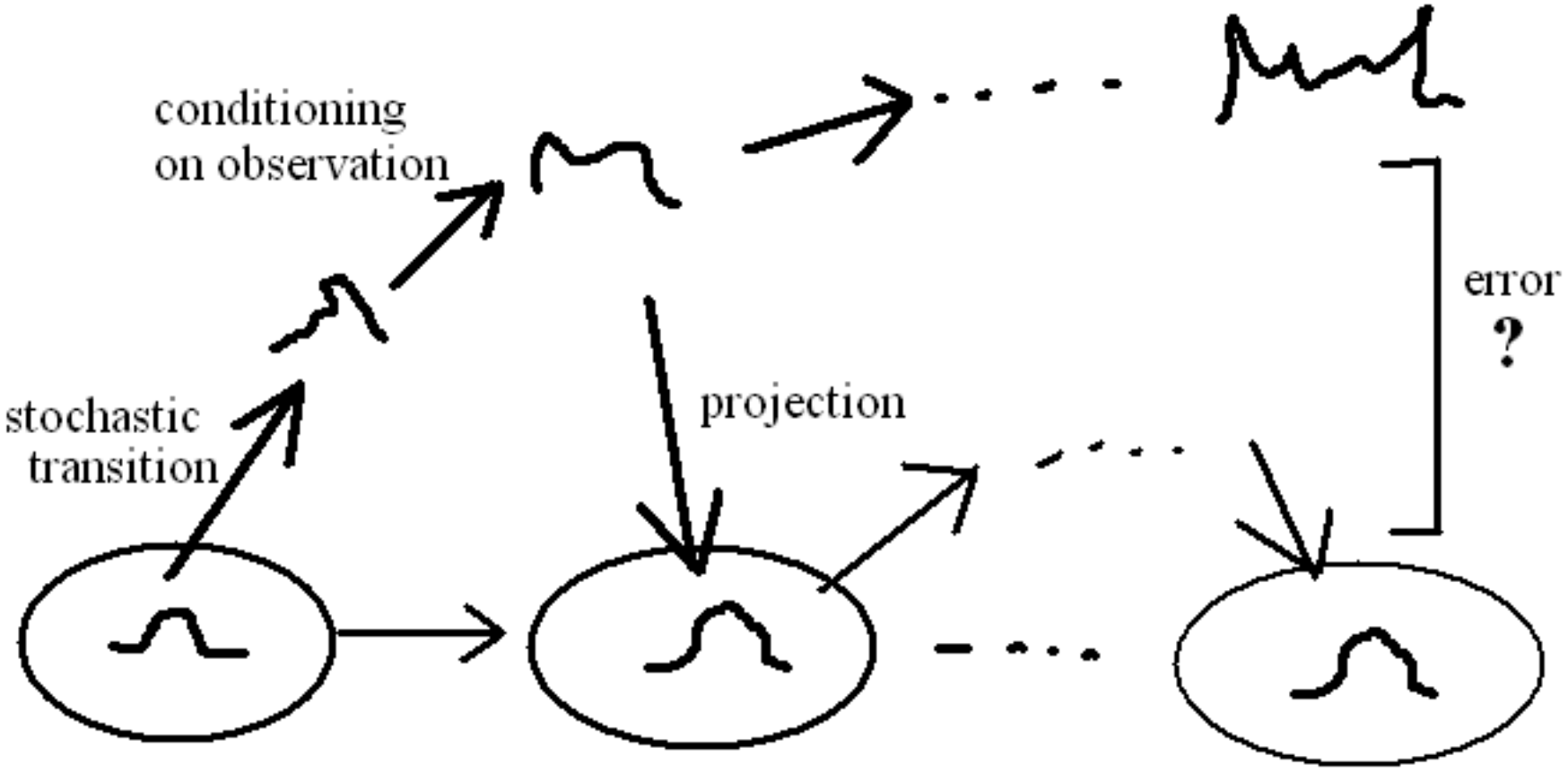
Use a decomposable representation for the belief state (pre-assume some independency)



Is there a Problem?

TECHNISCHE UNIVERSITÄT DARMSTADT

- What about the approximation errors?
  - It might accumulate and grow unbounded…



Contraction property

- Boyen and Koller have proven that
  - Under reasonable assumptions about the stochasticity of the process, every state transition results in a *contraction* of the distance between the two distributions by a constant factor
  - Since approximation errors from previous steps decrease exponentially, the overall error remains bounded *indefinitely*