Announcements

Quiz today!

General questions?

Demo of gather.town

Final coursework/project: submit your pdf/png format of your poster to a shared one drive folder (link will be provided on moodle) and also update the spreadsheet (link will be provided on moodle)

More info coming soon on moodle

Attendance

INM431: Machine Learning

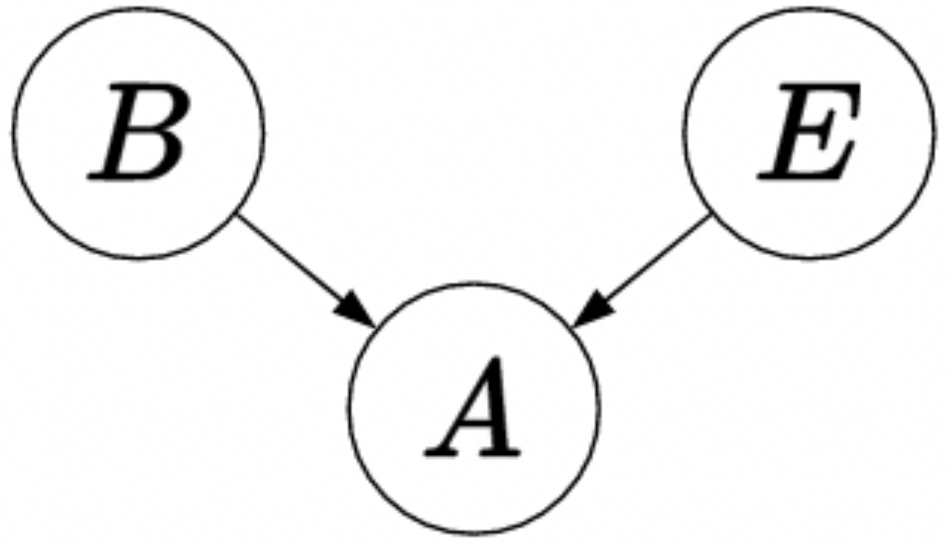
Sequence models

Pranava Madhyastha (pranava.madhyastha@city.ac.uk)

Based on materials by Artur S. d'Avila Garcez + Bishop's book + Murphy's book + Materials from Simone Teufel and Percy Liang

Quick recap

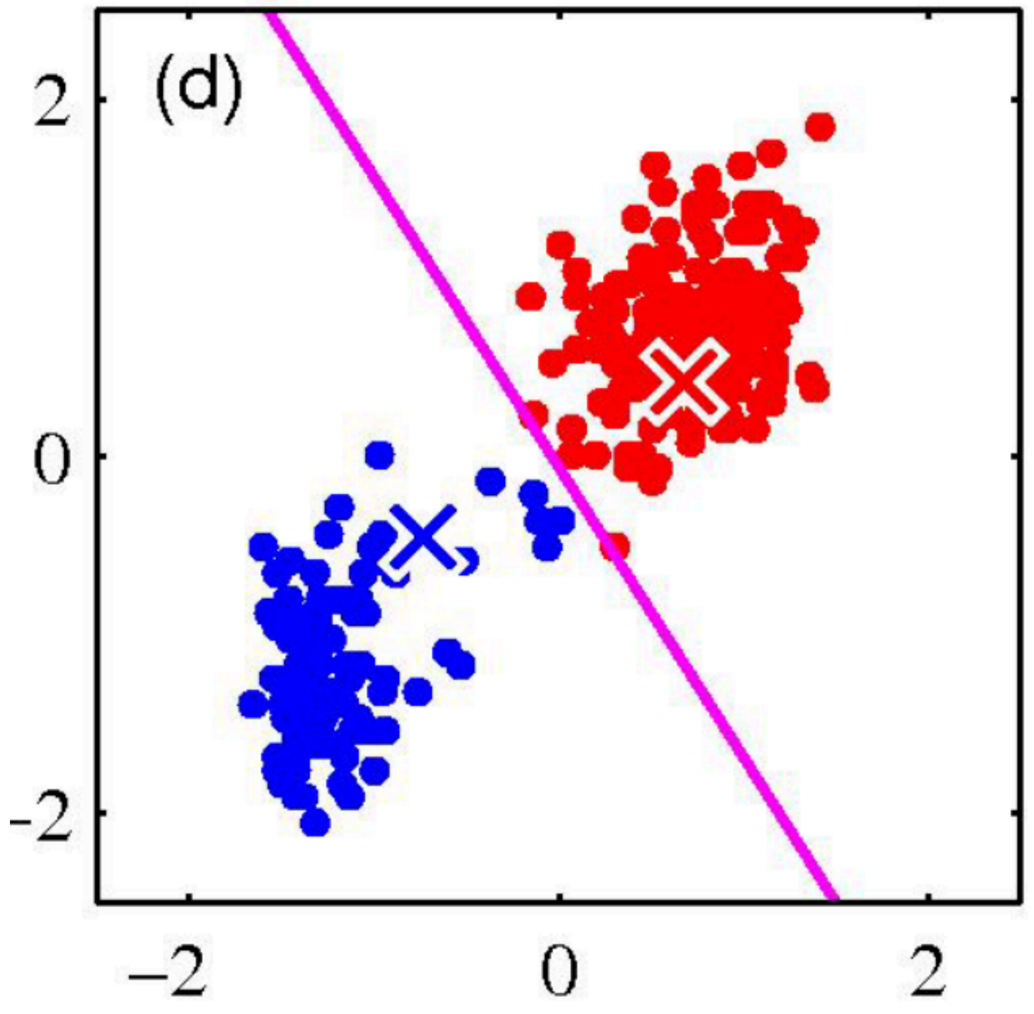
We have seen probabilistic graphical models with bayesian networks



We talked about probabilistic inference

We have seen k-Means and Gaussian Mixture models

- where we had a ‘latent/hidden variable’
- we developed intuitions for latent variable modelling



Quick recap

Probabilistic graphical models:

$$P(B, E, A) = P(B)P(E)P(A \mid B, E)$$

Bayes net

Naïve Bayes

Independence assumptions

On independence assumptions

i.i.d data

Complete joint distributions

Degree of independence

Computationally efficient

Theoretical support

Neat approximations

Expensive!

Sequential data

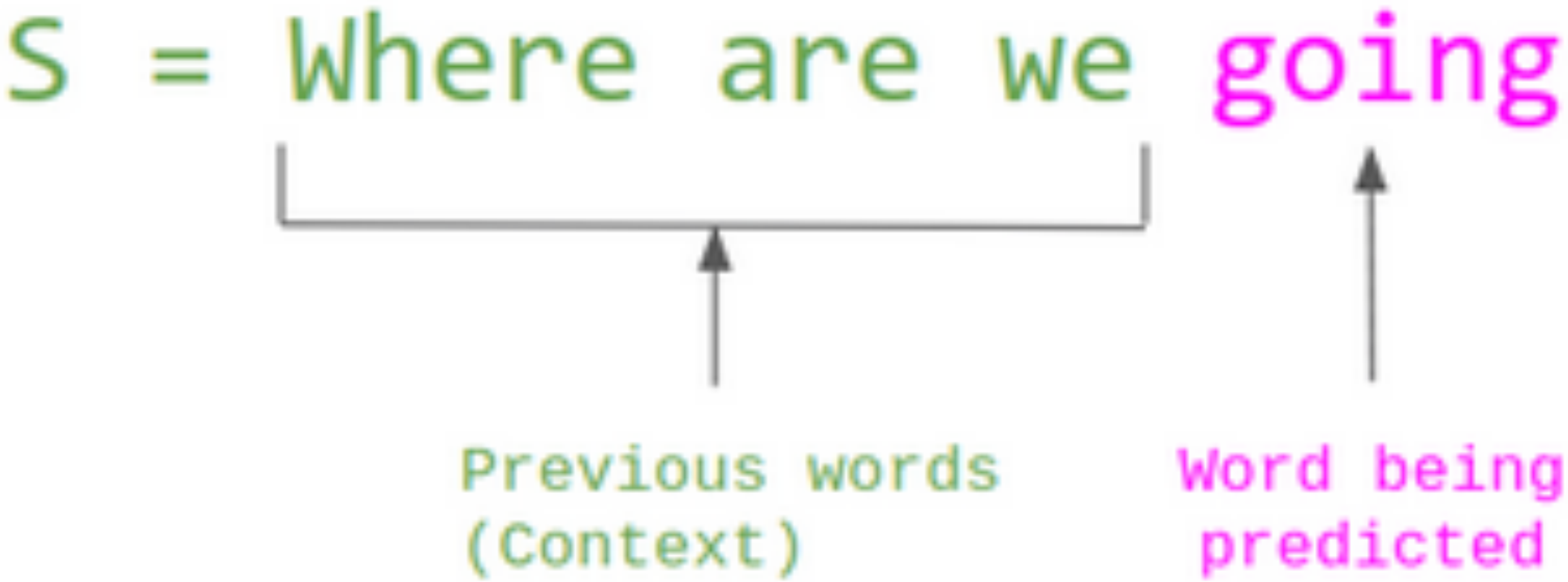
So far we have focused on sets of data points that were assumed to be independent and identically distributed (i.i.d.)

However, the i.i.d. assumption may not be the best assumption when modelling sequential data

Today we will focus on sequential data: such text, time-series data, DNA sequences, financial data, etc.,

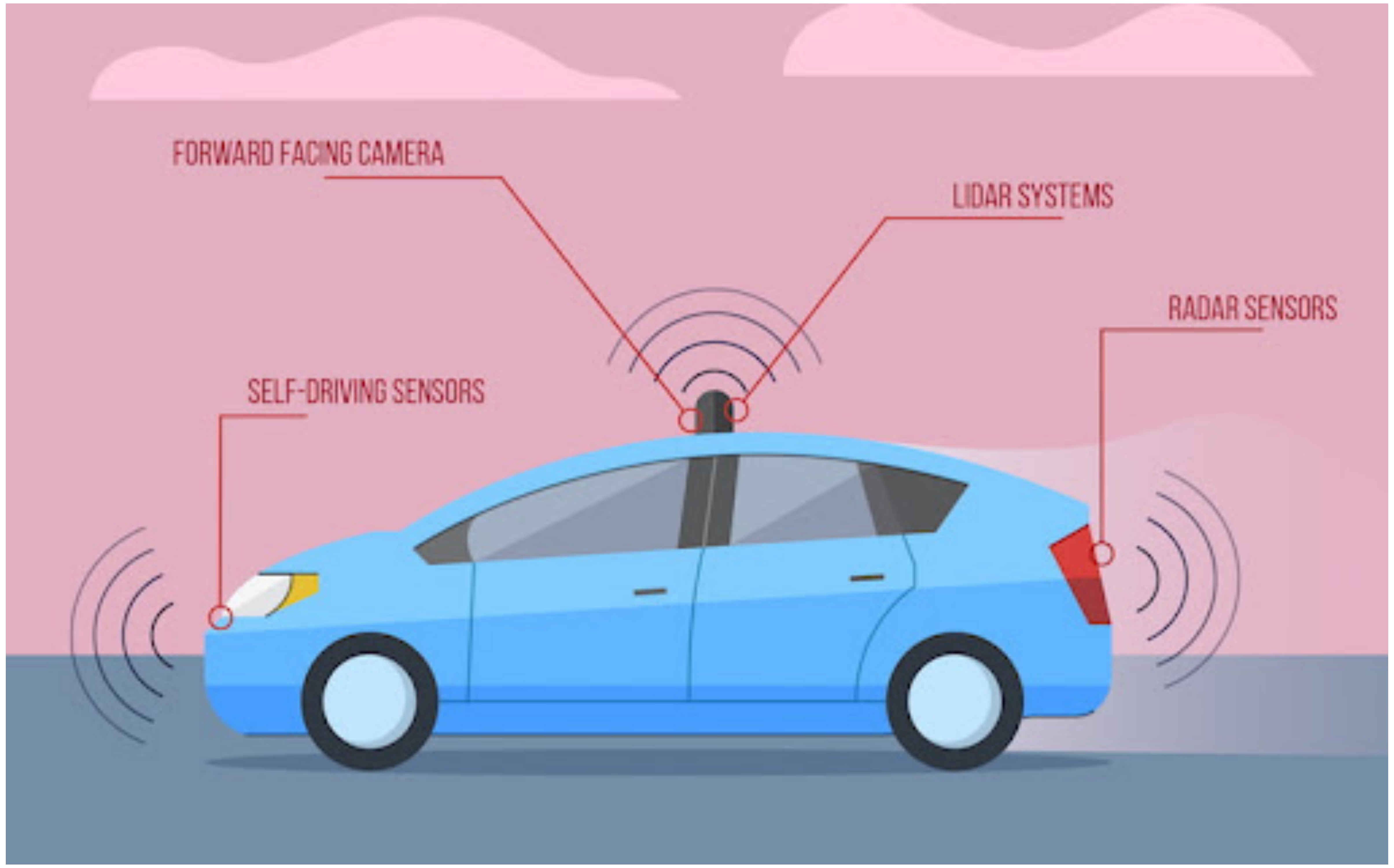
Sequential data

Language modelling:



Sequential data

Autonomous driving:



Sequential data

Return prediction (temporal observations - at some time state/time step):

Markov models

Basic assumption: the past is independent of the future given the present

Imagine this: $P(\text{high winds tonight}) > P(\text{large winds tonight})$

There are 3 words for each sentence here: w_1, w_2, w_3 and w_4, w_2, w_3, hence we are asking for:

$$P(w_1, w_2, w_3) > P(w4,w2,w3)$$

One possible way

$$P(w_1, w_2, w_3) > P(w4,w2,w3)$$

One simple way with massive independence assumption:

$$P(w_1) \times P(w_2) \times P(w_3) > P(w_4) \times P(w_2) \times P(w_3)??$$

This is too simplistic?

$$P(\text{high winds tonight}) > P(\text{large winds tonight})$$

Problems? Order doesn't matter, the assumption is too strong!

1st order Markov model

Basic assumption: the past is independent of the future given the present

$$p\left(w_i, w_k \mid w_j\right) = p\left(w_i \mid w_j\right) p\left(w_k \mid w_j\right) \quad i < j < k$$

In general:

$$p\left(w_1, \ldots, w_T\right) = p\left(w_1\right) \prod_{t=1}^{T-1} p\left(w_{t+1} \mid w_t\right)$$

use chain rule of probability

2nd order Markov model

Basic assumption: the past is independent of the future given the present

$$P(w_1, w_2, w_3) > P(w4, w2, w3)$$

With second order Markov assumption:

$$P(w_1) \times P(w_2 | w_1) \times P(w_3 | w_2, w_1) > P(w_4) \times P(w_2 | w_4) \times P(w_3 | w_2, w_4)$$

How about this?

Markov models in this course

We will focus only on 1st order Markov models

Generally to compute: $p(S) = p\left(w_1\right)\prod_{t=1}^{T-1} p\left(w_{t+1} \mid w_t\right)$

We would perform **Forward message passing**:

for t from 1 to $(T-1)$ do:

$$p\left(w_{t+1}\right) = \sum_{w_t} p\left(w_t\right) p\left(w_{t+1} \mid w_t\right)$$

Predicting weather

Two types of weather: rainy and cloudy

The weather doesn't change within the day

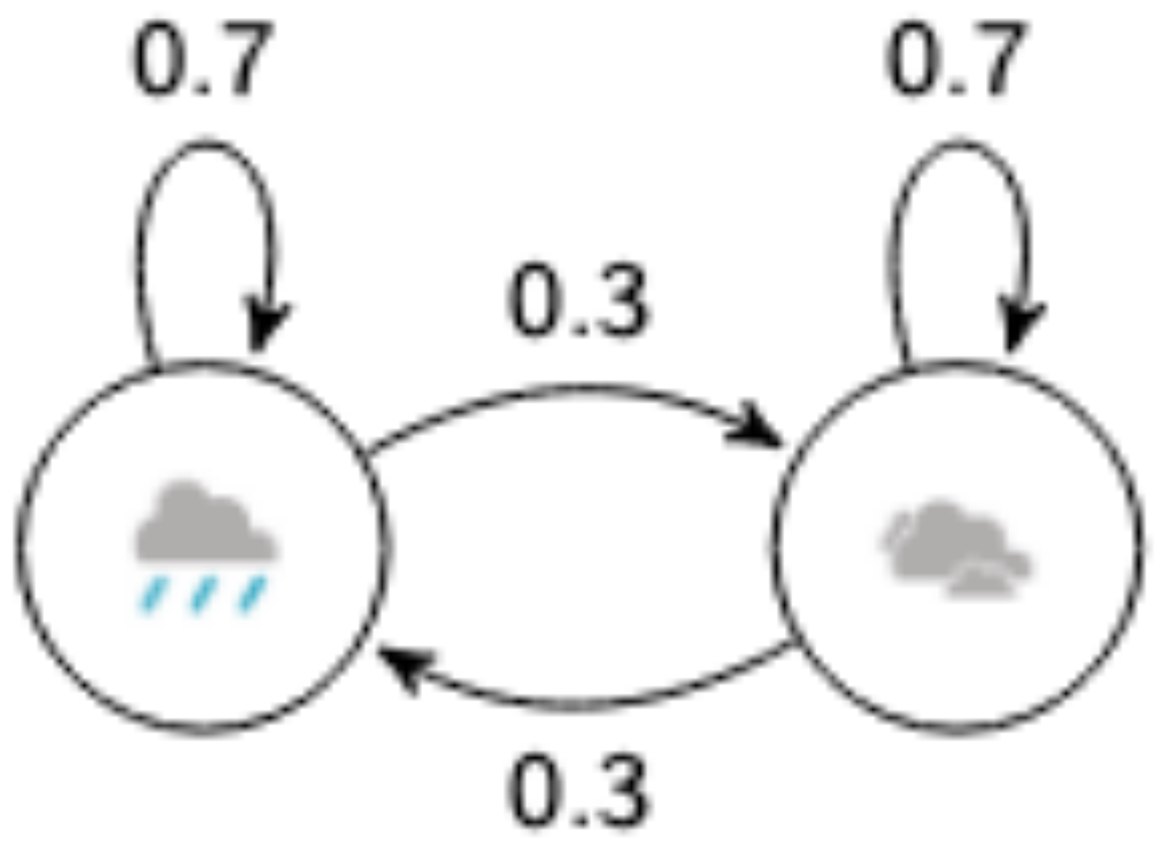
Can we guess what the weather will be like tomorrow?

We can use a history of weather observations:

$$P\left(w_t = \text{Rainy} \mid w_{t-1} = \text{Rainy}, w_{t-2} = \text{Cloudy}, w_{t-3} = \text{Cloudy}, w_{t-4} = \text{Rainy}\right)$$

Predicting weather

		Tomorrow	
		Rainy	Cloudy
Today	Rainy	0.7	0.3
	Cloudy	0.3	0.7



Transition probability matrix and the state diagram

Predicting weather: Markov chains

		Tomorrow	
		Rainy	Cloudy
Today	Rainy	0.7	0.3
	Cloudy	0.3	0.7

A Markov chain is a probabilistic process that assumes a Markovian Assumption.

In both of our examples - language model and Markov chains, we see that the states are observable, finite and discrete

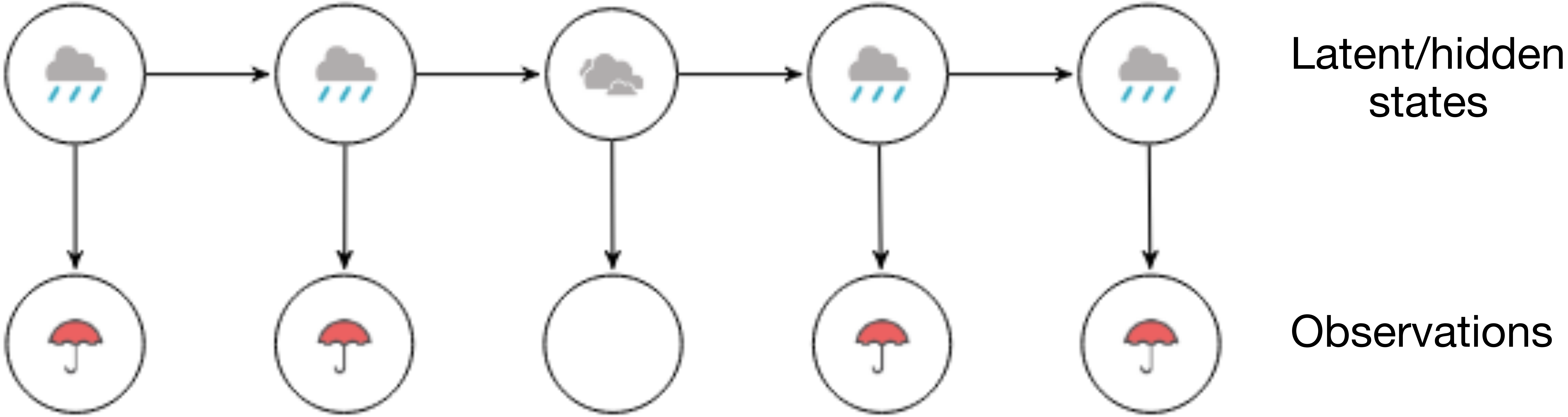
Famous Markov chain example

T9 or predictive texting

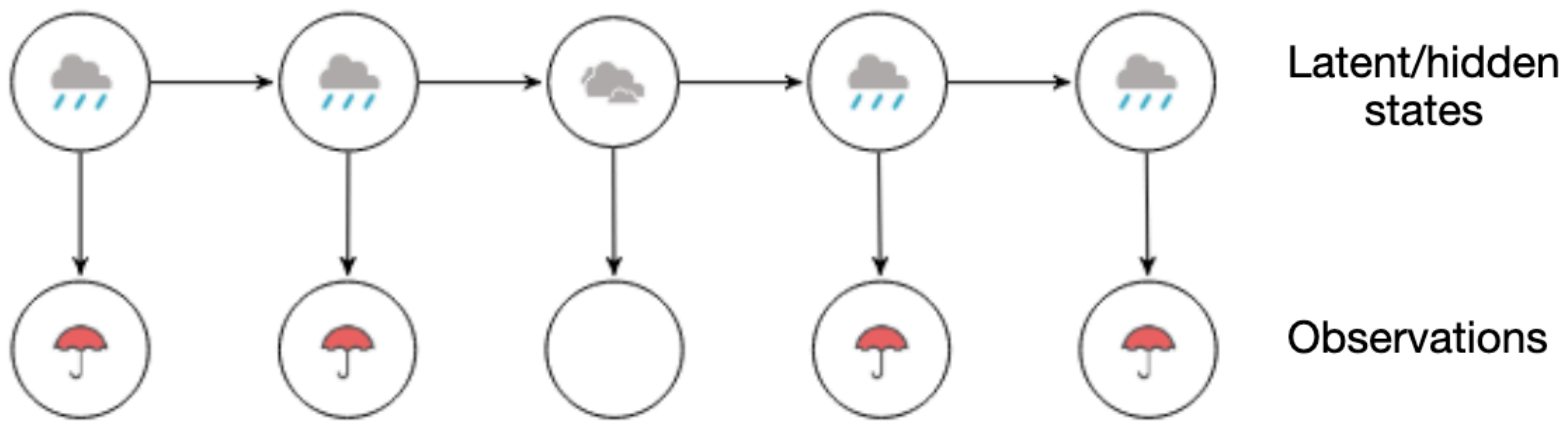
Most probable phoneme sequences for speech recognition

Dialating markov chains

Hidden Markov model



Hidden Markov Models



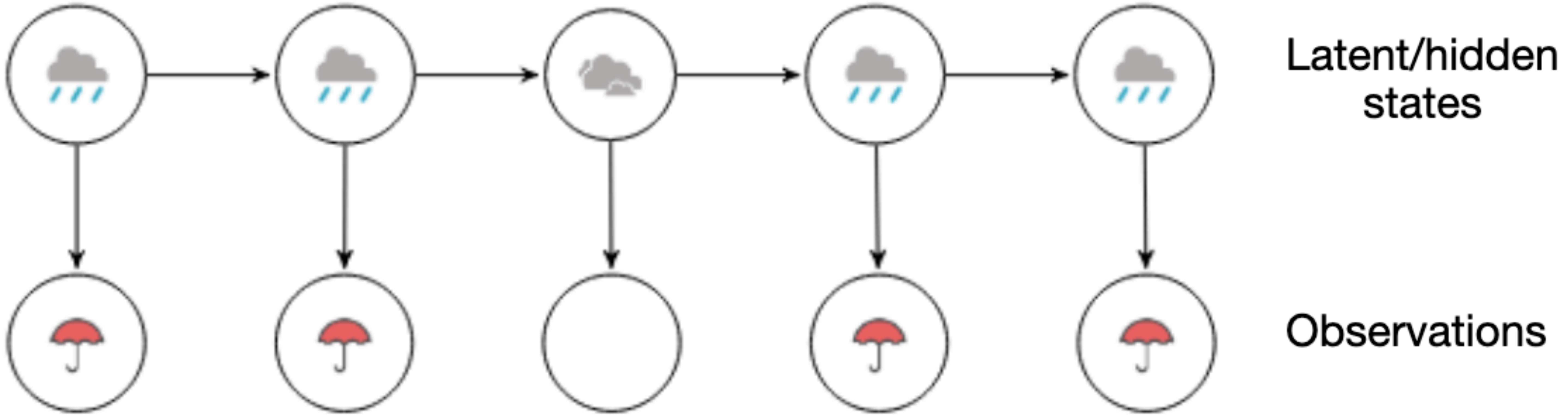
Here, we make a Markov assumption over the latent states

At each time step, we only have access to the ‘observations’

There need not be one to one mapping between the latent states & observations

Goal: Infer the sequence of hidden states given a sequence of observations!

Hidden Markov Models: tagging



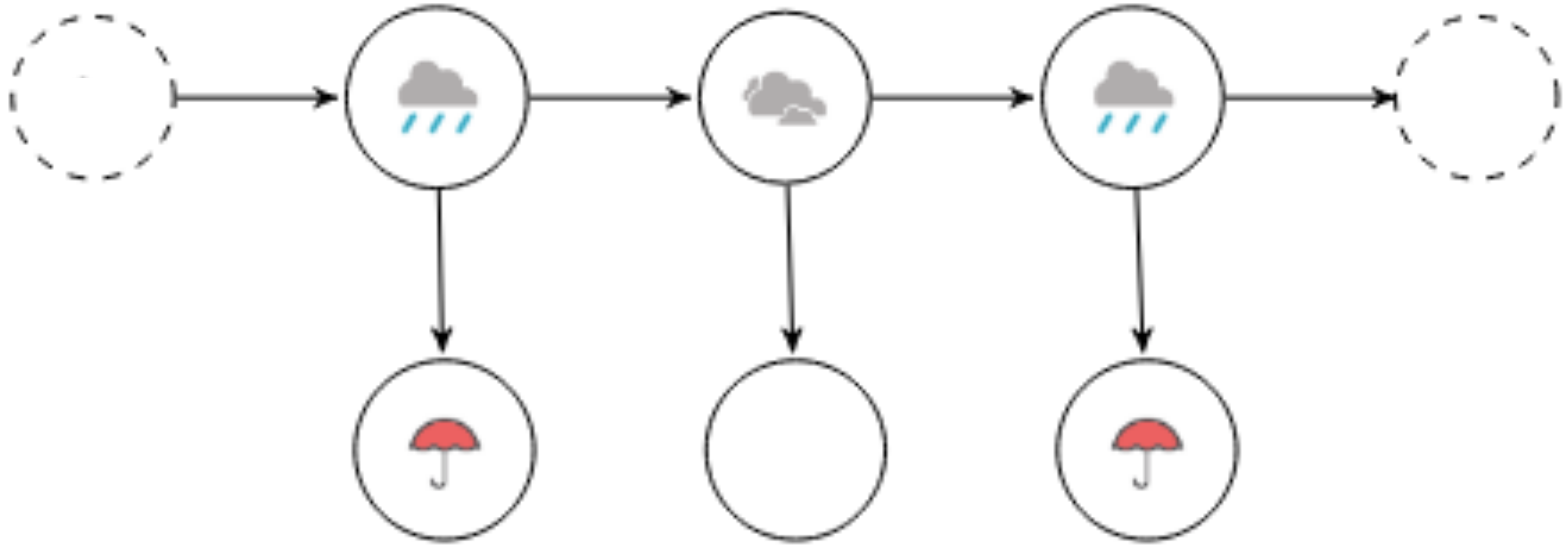
	Rainy	Cloudy
Rainy	0.7	0.3
Cloudy	0.3	0.7

State transition probability ($P(x_t | x_{t-1})$)

	Umbrella	No Umbrella
Rainy	0.9	0.1
Cloudy	0.2	0.8

State emission probabilities/observation likelihoods ($P(y_t | x_t)$)

Hidden Markov model



$Z_i \in \{1,\cdots,K\}$: latent state or unobservable label (tagging/object tracking) at time step i

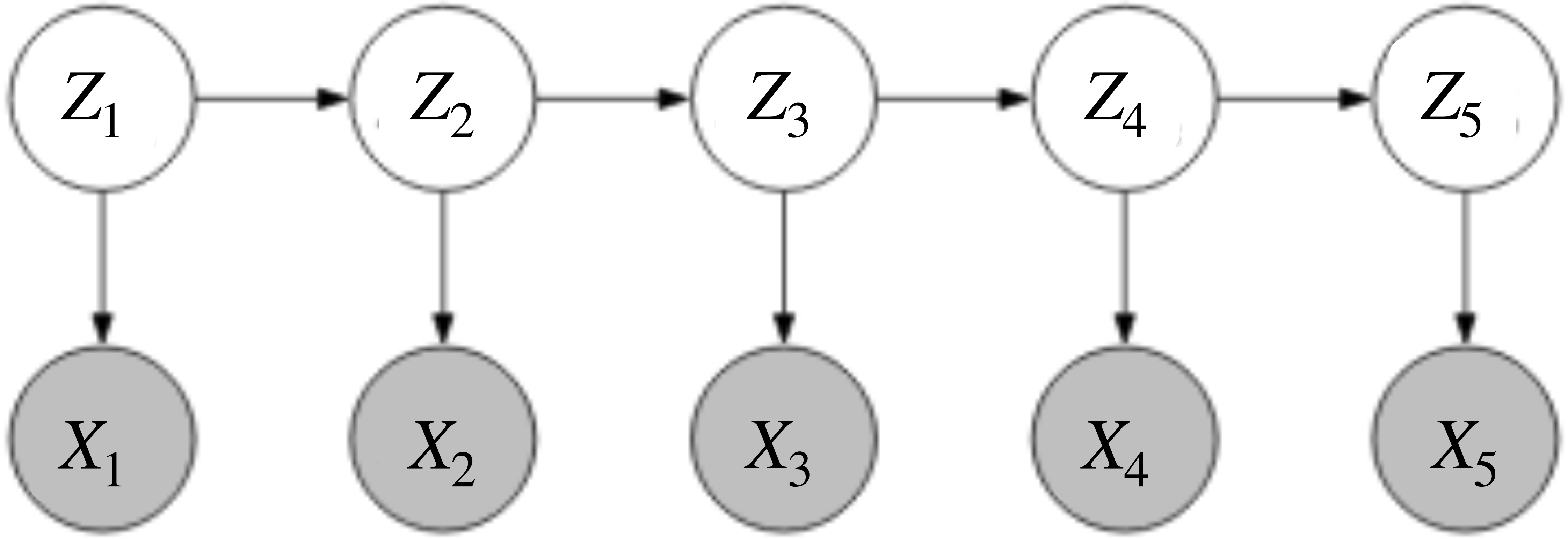
$X_i \in \{1,\cdots,K\}$: Observation, or the word, at time step i

Starting/special probability: $p(z_1)$

Transition probabilities: $p(z_i \mid z_{i-1})$

Emission probabilities: $p(x_i \mid z_i)$

Formal model



$$P(Z = z, X = x) = \underbrace{p\left(z_1\right)}_{\text{start}} \prod_{i=2}^{n} \underbrace{p\left(z_i \mid z_{i-1}\right)}_{\text{transition}} \prod_{i=1}^{n} \underbrace{p\left(x_i \mid z_i\right)}_{\text{emission}}$$

The continuous analogue

If the observations are continuous $p(x_i \mid z_i)$ can be modelled by say a Gaussian:

$$p\left(\mathbf{x}_t \mid z_t = k, \boldsymbol{\theta}\right) = \mathcal{N}\left(\mathbf{x}_t \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$$

where k is state value index.

Instead of having output probabilities, we have a function mapping from the value of the latent states to the mean (and potentially covariance) of a Gaussian distribution

Stock market (bullish/bearish) predictions

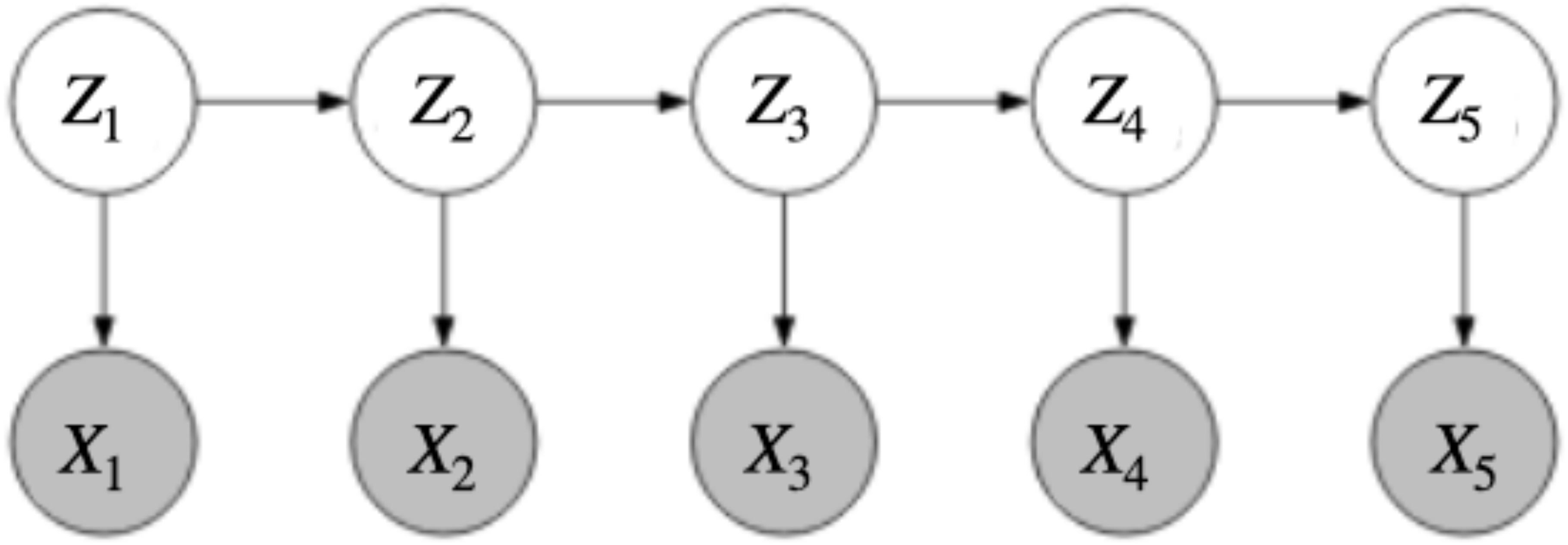
Observations (bull/bear) are independent, discrete

Observations depends on the internal state of the market

The historical data captures any significant breaks

The latent behaviour of the stock market behaviour can be quantised into a finite number of states

Two popular formulations

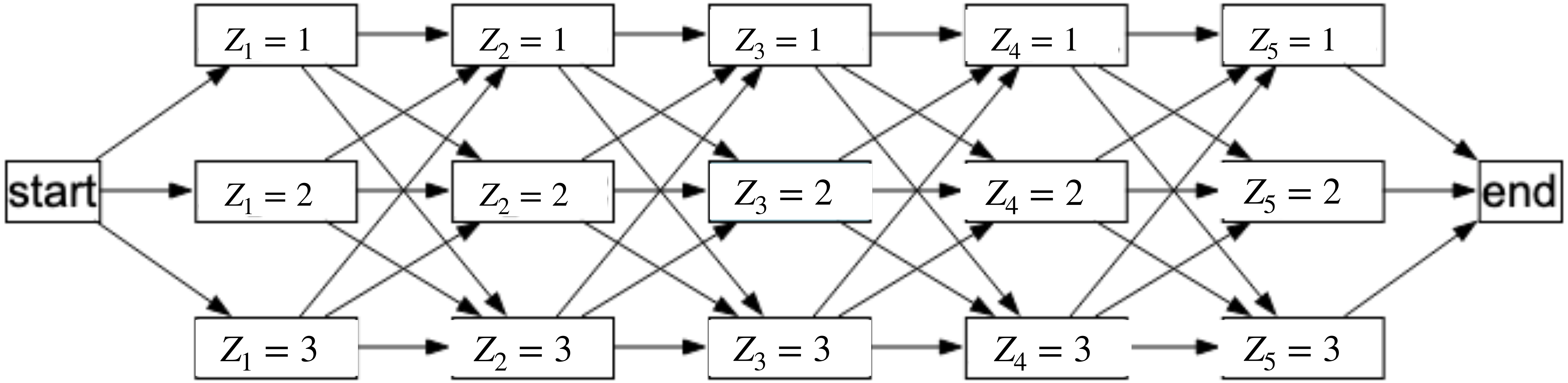


$P\left(z_3 \mid x_1, x_2, x_3\right)$: the distribution of a particular hidden variable conditioned on only the observation up until that point. This is useful when you're doing real-time predictions where you cannot see the future.

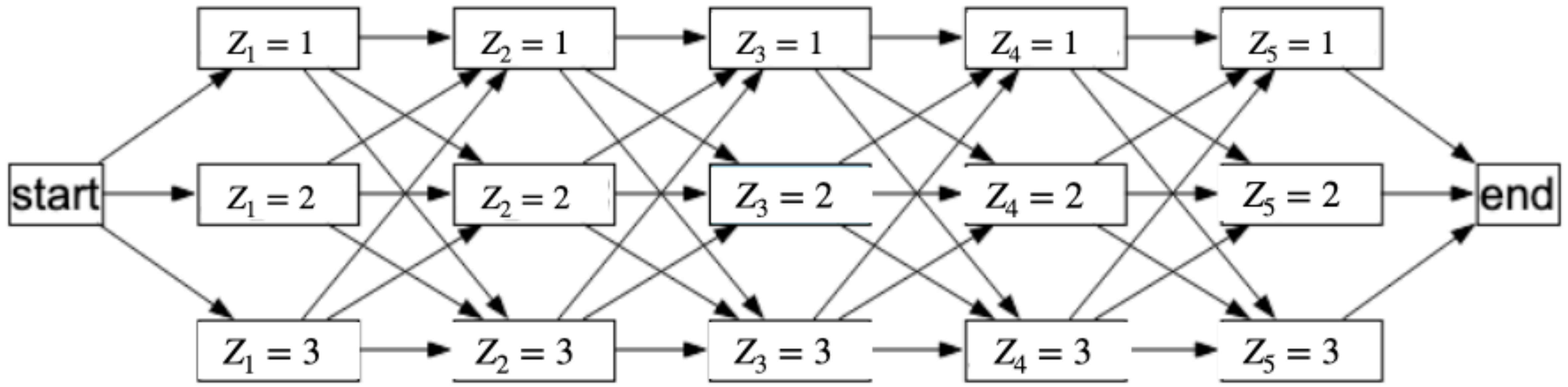
$P\left(z_3 \mid x_1, x_2, x_3, x_4, x_5\right)$: the distribution of a particular hidden variable conditioned on all the observations including the future. This is useful when you have collected all the data and then work towards having the model that can describe the data.

First one is a special case of the second - we can marginalise out the future observations

The lattice representation



Lattice representation



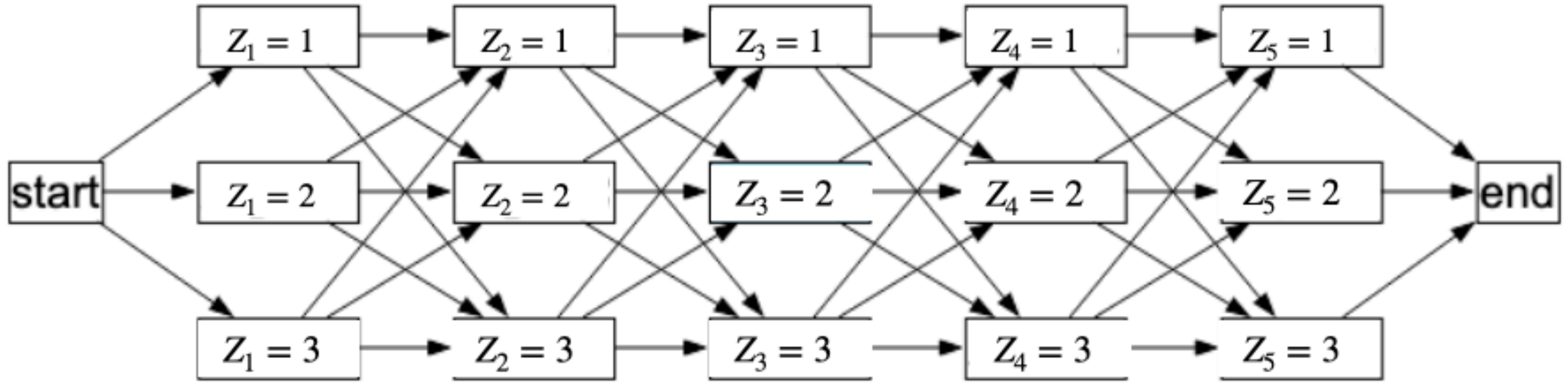
Imagine that we are working with a graph type representation

It has a start node, an end node, and a node for each assignment of a value to a variable $Z_i =$ (state assignment).

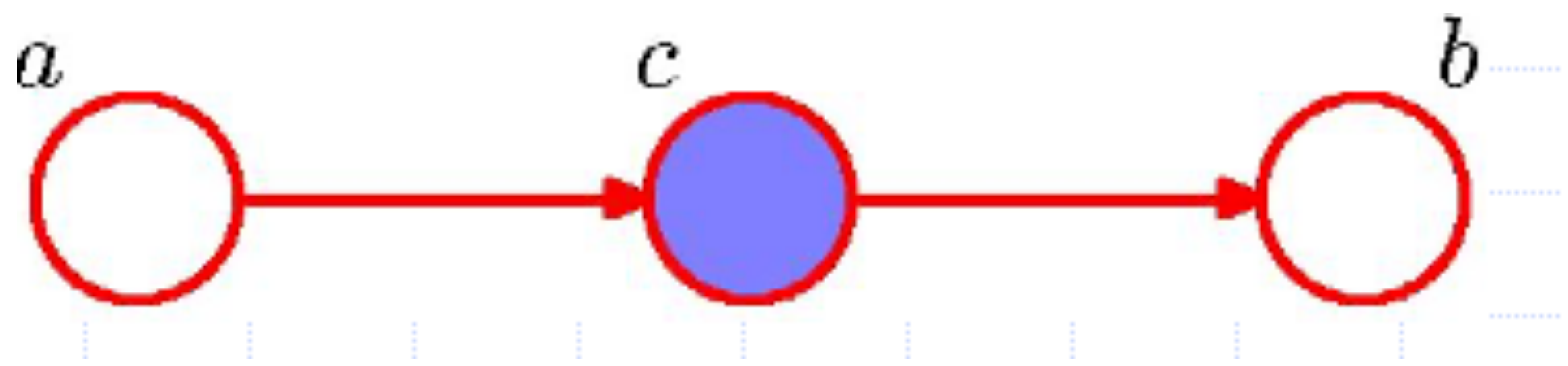
The nodes are arranged in a lattice, where each column corresponds to one variable Z_i; and each row a corresponds to a particular state assignment.

Every path from the start to the end corresponds exactly to a complete assignment to the nodes.

Lattice representation



Recall from Bayes Net lecture:

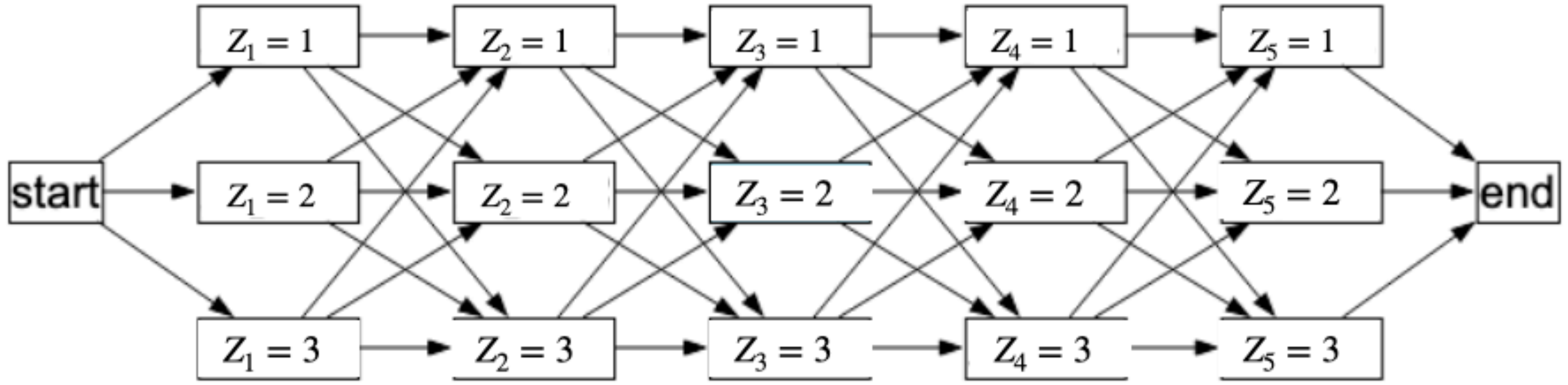


$$P(a, b \mid c) = \frac{P(a, b, c)}{P(c)}$$

$$= \frac{P(a)P(c \mid a)P(b \mid c)}{P(c)}$$

$$= P(a \mid c)P(b \mid c)$$

Lattice representation

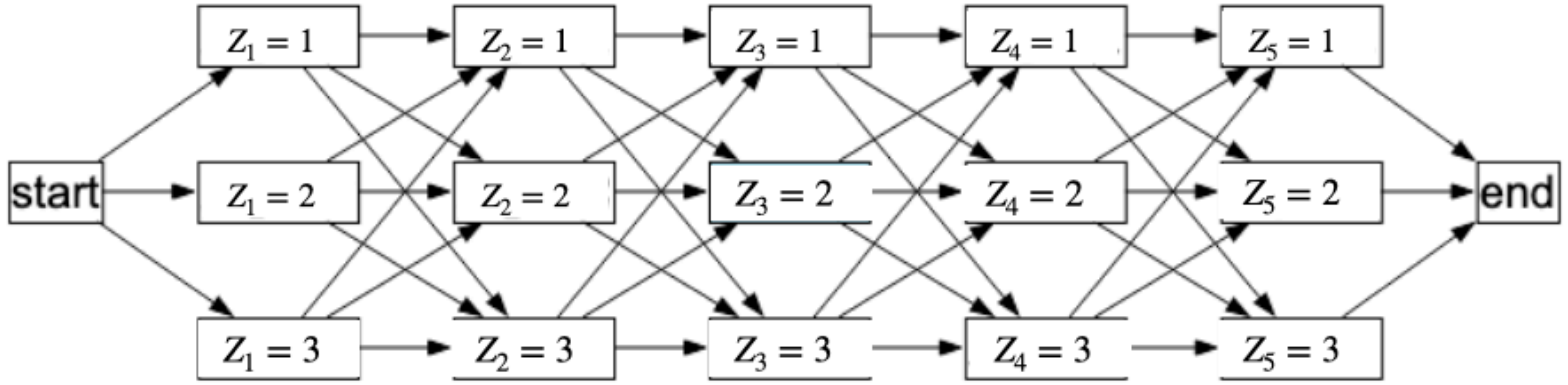


Edge start $\rightarrow Z_1 = z_1$ has a weight $p(z_1)p(e_1 \mid h_1)$

Edge $Z_{i-1} = z_{i-1} \rightarrow Z_i = z_i$ has weight $p(z_i \mid z_{i-1})p(x_i \mid z_i)$

For each edge we multiply by the transition probability into z_i and its emission probability $p(x_i \mid z_i)$. This defines a weight for each path (assignment) in the graph equal to the joint probability distribution $P(Z, X)$

Forward



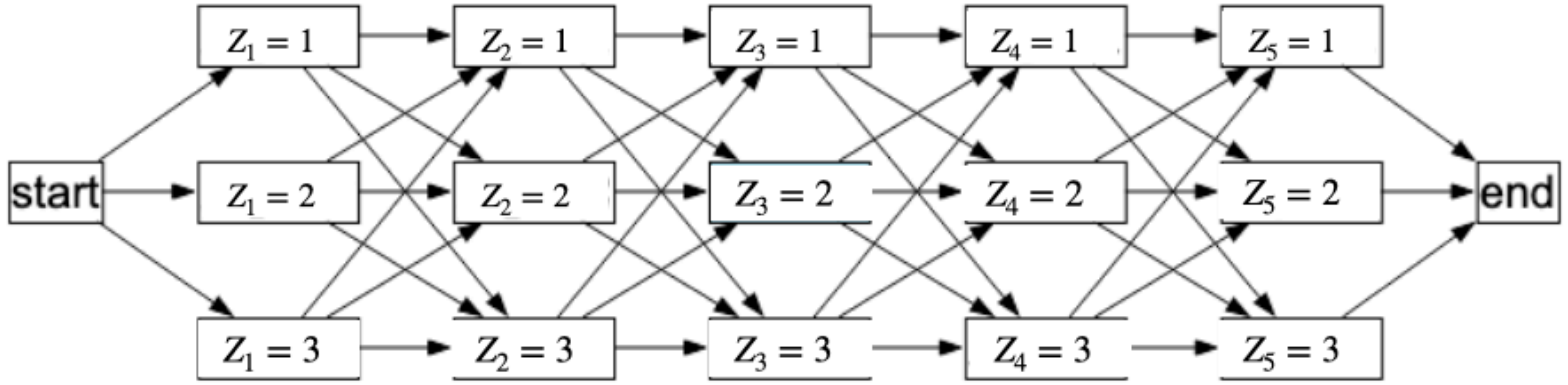
We will now use the recursive formalism using dynamic programming to compute effectively any query for a particular node.

To do this, we will define: Forward and Backward probabilities (in Bishop's book this is referred to as α and β):

Forward: $F_i\left(z_i\right) = \sum_{z_{i-1}} F_{i-1}\left(z_{i-1}\right) w\left(z_{i-1}, z_i\right)$

Which is the sum of weights of paths from 'start' $\rightarrow Z_i = z_i$

Backward



Backward: $B_i\left(z_i\right) = \sum_{z_{i+1}} B_{i+1}\left(z_{i+1}\right) w\left(z_i, z_{i+1}\right)$

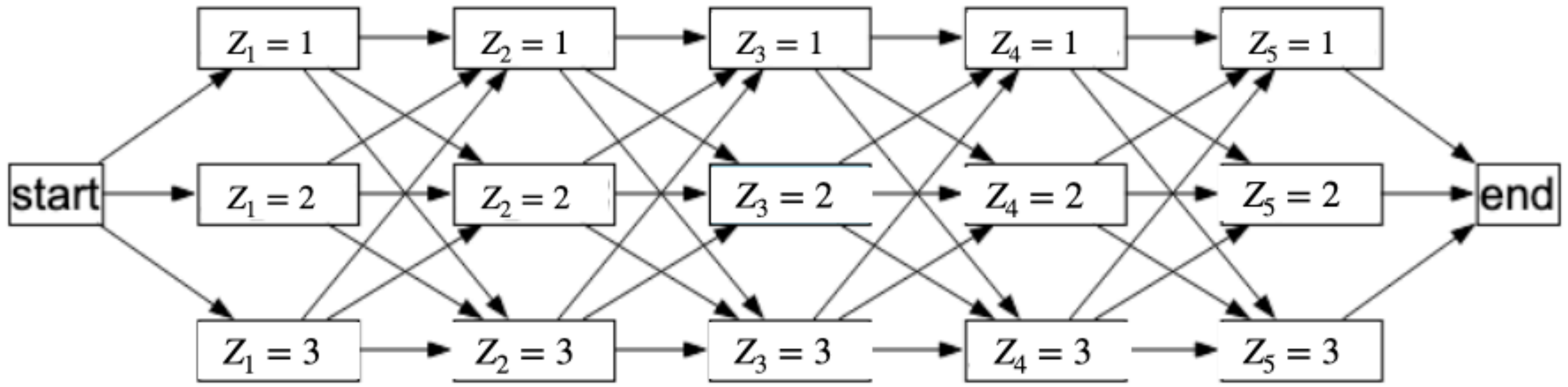
Which is the sum of weights of paths from 'end' $\rightarrow Z_i = z_i$

Let us then define sum of weights = $S_i(z_i) = F_i(z_i)B_i(z_i)$

- sum of the weights over all paths from the start node to the end node that pass through the particular intermediate node: which is the product of the weights of paths going through z_i and leaving it!

Normalise this to get the probability distribution!

Forward/Backward



Computationally, this costs $O(nK^2)$

The algorithm is:

Compute forward

Compute backward

Compute sum of weights of the paths and normalise!

Another form of inference

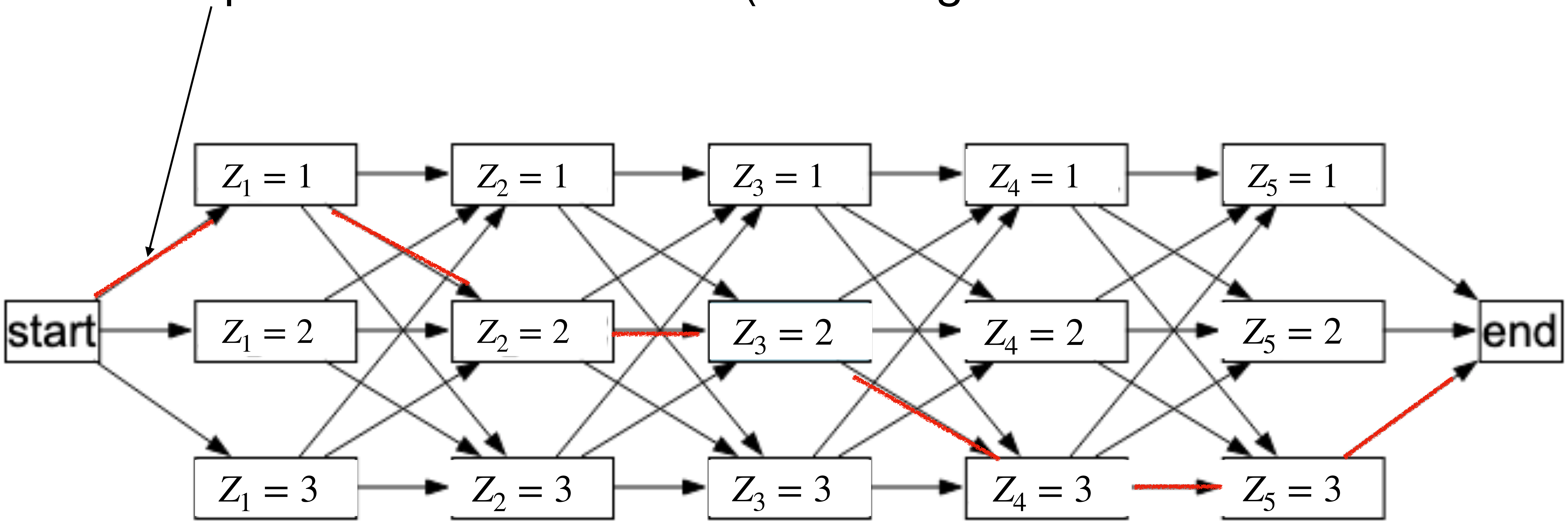
Let us assume that we have observed data: $X = \{x_1, \cdots, x_n\}$ and we want to estimate the most probable sequence of states:

$$\mathbf{z}^* = \arg\max_{\mathbf{z}_{1:T}} p\left(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}\right)$$

This is a very common form of inference, tagging words with their parts of speech (noun, verb, etc.,), handwriting recognition, speech recognition, etc.,

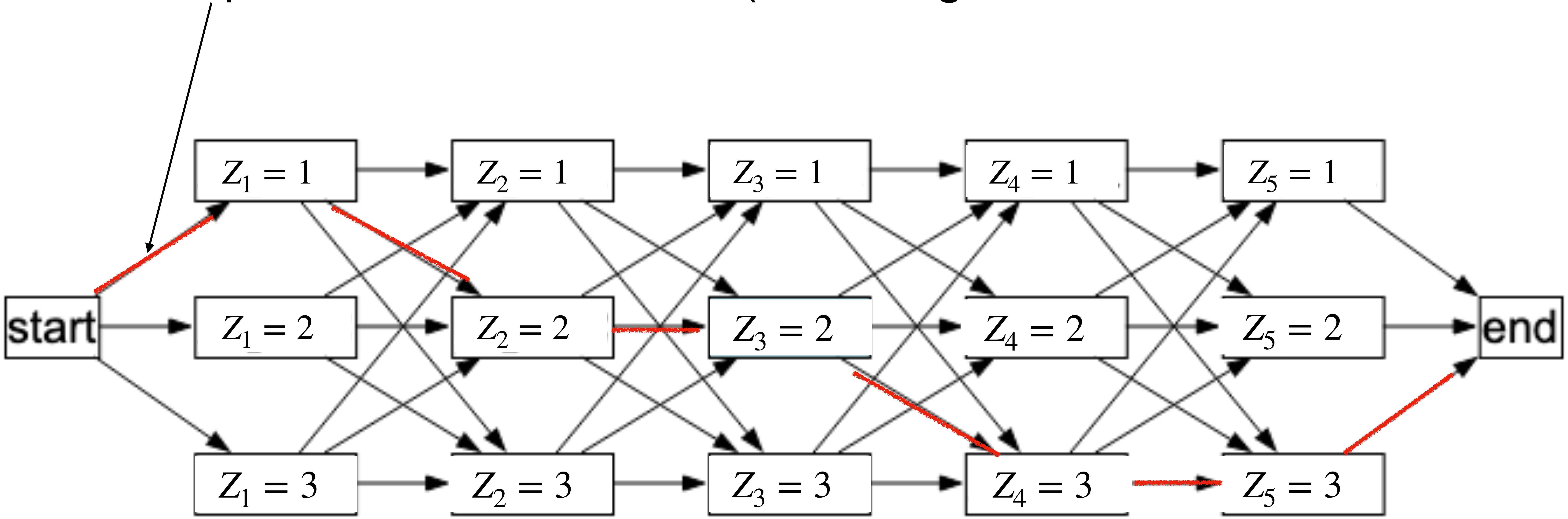
Enter the Viterbi algorithm

The best sequence of hidden states (assuming we have access to the observations)



Enter the Viterbi algorithm

The best sequence of hidden states (assuming we have access to the observations)



Dynamic Programming algorithm

Solve an optimisation problem by breaking it down into simpler subproblems.

Optimal solution to the overall problem depends upon the optimal solution to its subproblems.

(There is a small thing here - backtracking to get the best path, but that’s a technicality)

Viterbi setting

Goal: Compute $z^* = \operatorname*{argmax}_z p(z \mid x)$

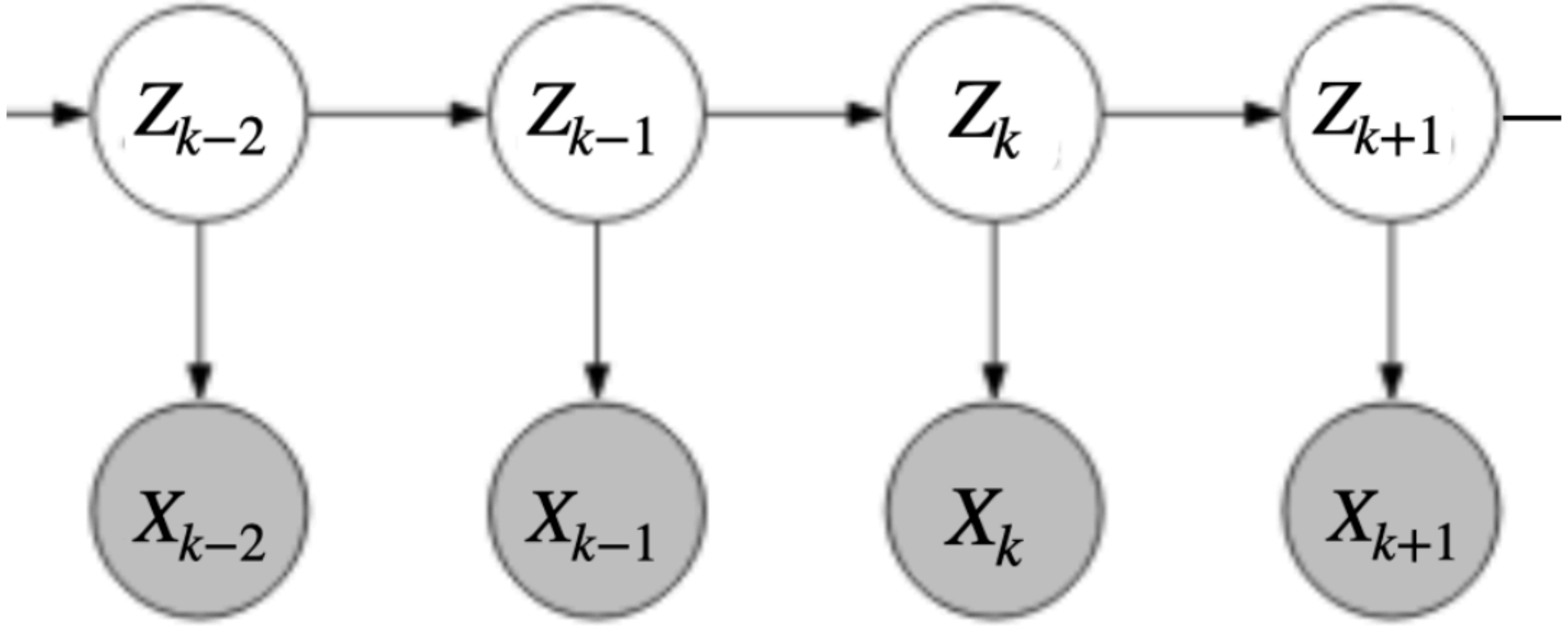
We will make use of this assumption:

$$\operatorname*{argmax}_{z_{1:n}} p(z \,|\, x) = \text{argmax}_{z_{1:n}}\, p(z, x)$$

And a lemma: If $f(a) \geq 0\, \forall a$ and $g(a, b) \geq 0\, \forall (a, b)$; then:

$$\max_{a,b} f(a) g(a, b) = \max_a \left(f(a) \max_b g(a, b) \right)$$

Viterbi intuition



$$r_k(z_k) = \max_{z_{1:k-1}} p(z_{1:k}, x_{1:k})$$

Using the local probability distribution using the definition of Bayes net:

$$= \max_{z_{1:k-1}} p(x_k \mid z_k) p(z_k \mid z_{k-1}) p(z_{1:k-1}, x_{1:k-1})$$

Viterbi intuition

$$= \max_{z_{1:k-1}} p(x_k \mid z_k) p(z_k \mid z_{k-1}) p(z_{1:k-1}, x_{1:k-1})$$

Let's use the lemma, by setting $a = z_{k-1}$ and $b = z_{1:k-2}$

$$= \max_{z_{1:k-1}} \Big(\underbrace{p(x_k \mid z_k) p(z_k \mid z_{k-1})}_{\text{this can be computed}} \underbrace{\boxed{\max_{z_{1:k-2}} p(z_{1:k-1}, x_{1:k-1}) \Big)}}_{r_{k-1}(z_{k-1})}$$

Viterbi intuition

Rewriting this:

$$r_k(z_k) = \max_{z_{1:k-1}} \left(p(x_k \mid z_k) p(z_k \mid z_{k-1}) r_{k-1}(z_{k-1}) \right)$$

One small weirdness: what happens when k=1?

$r_1(z_1) = p(z_1, x_1)$ - special condition!

$$r_1(z_1) = p(z_1)p(x_1 \mid z_1)$$

Viterbi intuition

In general:

Best path = $\text{argmax}_{z_{1:k-1}} \left(p(x_k \mid z_k) p(z_k \mid z_{k-1}) r_{k-1}(z_{k-1}) \right)$

And the special case: $r_1(z_1) = p(z_1) p(x_1 \mid z_1)$

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

Generalisation: State Space Models

A state space model (SSM) is just like an HMM, except the hidden states are now continuous.

An SSM can be written in the following generic form:

$$\mathbf{z}_t = g\left(\mathbf{u}_t, \mathbf{z}_{t-1}, \boldsymbol{\epsilon}_t\right)$$

$$\mathbf{y}_t = h\left(\mathbf{z}_t, \mathbf{u}_t, \boldsymbol{\delta}_t\right)$$

where, $\mathbf{z}_t$ is a hidden state

$\mathbf{u}_t$ is an optional input or control signal

$\mathbf{y}_t$ is the observation

g is the transition model

h is the observation/emission model

$\boldsymbol{\epsilon}_t$ is the system noise

$\boldsymbol{\delta}_t$ is the observation noise

State Space Models: use case

An important special case of an SSM is where all the conditional probability distributions are Gaussian distributed and the transition/observation models are linear functions. This is also called a linear dynamical system (LDS).

Applications of SSMs:

- Object tracking
- Simultaneous localisation and mapping (SLAM) robotics

Summary

Models for sequential data

Markov models, Markov assumptions

Hidden Markov models - Learning using Forward/Backward, Viterbi decoding

Structural properties of HMM make is feasible with neat algorithms!