Markov Chains

OVERVIEW

We now random sequences in which, roughly speaking, each entry can depend on the previous one.

Motivation

- The simplest type of random sequences are those with independent entries.
- In general, joint distributions with dependence are much more complicated objects.
- In Markov chains, dependence can be present, but is tightly restricted.
- This results in a class of distributions that can represent (some) dependence, but are still mathematically and computationally tractable.

Uses

In this class, Markov chains will appear predominantly in two contexts:

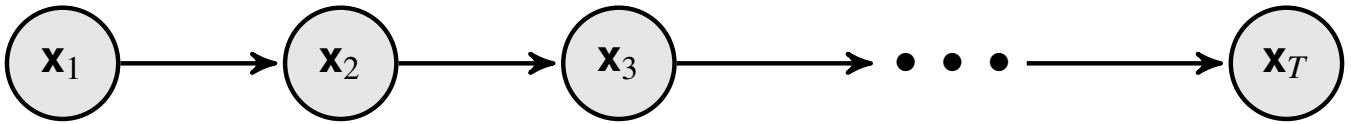
- As latent variables in so-called latent chain models.
- As an inference in tool in sampling algorithms.

A general joint distribution of a sequence:

$$P(\mathbf{x}_1, \ldots, \mathbf{x}_t) = P(\mathbf{x}_1)P(\mathbf{x}_2|\mathbf{x}_1)P(\mathbf{x}_3|\mathbf{x}_1, \mathbf{x}_2) \cdots P(\mathbf{x}_t|\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_{t-1})$$

First-order Markov model:

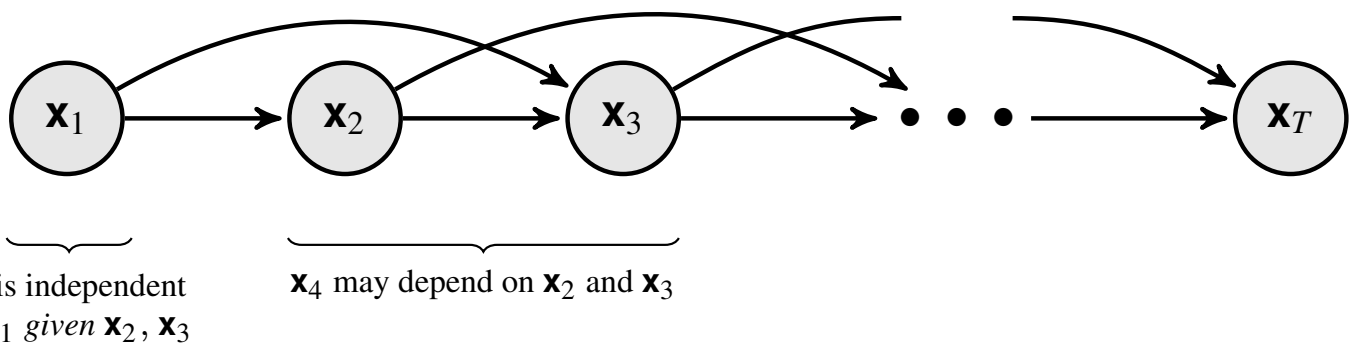
$$P(\mathbf{x}_1, \ldots, \mathbf{x}_t) = P(\mathbf{x}_1)P(\mathbf{x}_2|\mathbf{x}_1) \cdots P(\mathbf{x}_t|\mathbf{x}_{t-1})$$



The term *Markov* refers to a conditional independence relationship. In this case, the Markov property is that, given the present observation ($\mathbf{x}_t$), the future ($\mathbf{x}_{t+1}, \ldots$) is independent of the past ($\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}$).

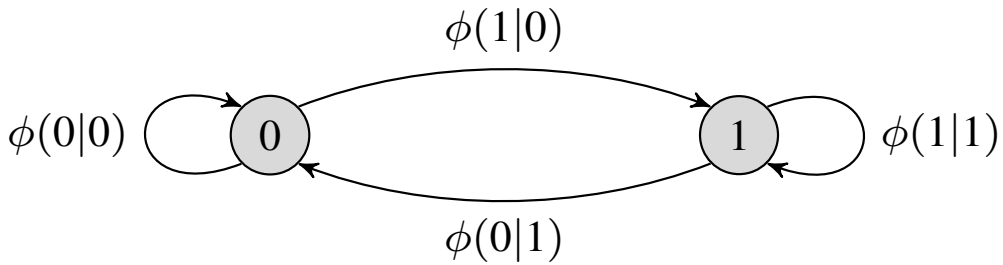
Second-order Markov model:

$$P(\mathbf{x}_1, \ldots, \mathbf{x}_t) = P(\mathbf{x}_1)P(\mathbf{x}_2|\mathbf{x}_1) \cdots P(\mathbf{x}_{t-1}|\mathbf{x}_{t-3}, \mathbf{x}_{t-2})P(\mathbf{x}_t|\mathbf{x}_{t-2}, \mathbf{x}_{t-1})$$



A simple binary chain

We consider a Markov chain $X_0, X_1, X_2, \ldots$ with state space $\mathcal{X}$. Suppose $\mathcal{X} = \{0, 1\}$.



- We regard 0 and 1 as possible "states" of X, represented as vertices in a graph.
- Each pair $X_{t-1} = i, X_t = j$ in the sequence is regarded as a "transition" from i to j and represented as an edge in the graph.
- Each edge $i \to j$ is weighted by the probability

$$\phi(j|i) := P(X_t = j | X_{t-1} = i) \ .$$

Caution: The graph is *not* a graphical model. (For computer scientists: It is a probabilistic finite automaton.)

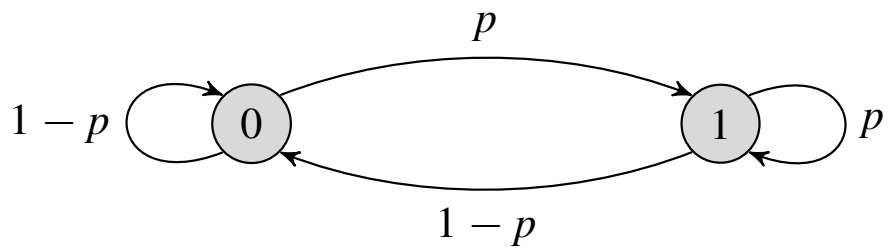
Stationarity

The graph representation is only possible if $\phi(j|i)$ is independent of t. Otherwise we would have to draw a different graph for each t.

If $\phi(j|i)$ does not depend on t, the Markov chain is called **stationary**.

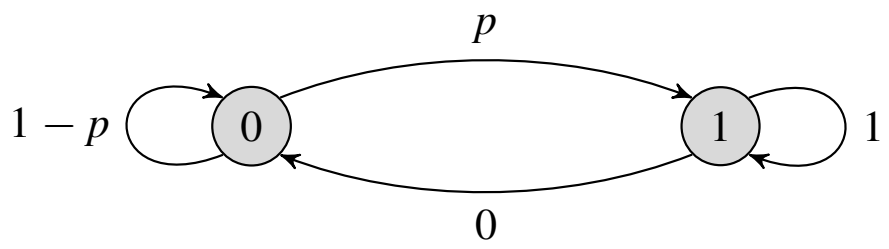
First example: Independent coin flips

Suppose X is a biased coin with $P(X_t = 1) = p$ independently of X_{t-1}. In other words, the sequence (X_t) is iid Bernoulli with parameter p.



Breaking independence

Here is a simple modification to the chain above; only $\phi(0|1)$ and $\phi(1|1)$ have changed:



This is still a valid Markov chain, but the elements of the sequence are no longer independent.

Transition matrix

The probabilities $\phi(j|i)$ are called the **transition probabilities** of the Markov chain. If $|\mathcal{X}| = K$, the $K \times K$-matrix

$$\Phi := (\phi(j|i))_{i,j \leq K} = \begin{pmatrix} \phi(1|1) & \dots & \phi(K|1) \\ \vdots & & \vdots \\ \phi(1|K) & \dots & \phi(K|K) \end{pmatrix}$$

is called the **transition matrix** of the chain. This is precisely the adjacency matrix of the graph representing the chain. Each row is a probability distribution on K events.

Initial distribution

The distribution of the first state, i.e. the vector

$$\mathbf{P}_{\text{init}} := (P(X_0 = 1), \dots, P(X_0 = K)) \ ,$$

is called the **initial distribution** of the Markov chain.

The distribution of a stationary Markov chain with finite state space is completely determined by the pair $(\Phi, \mathbf{P}_{\text{init}})$.

Distribution after the first step

- If we *know* the initial state, then

$$P(X_1 = j \,|\, X_0 = i) = \phi(j|i) \ .$$

- Now consider the marginal distribution $\mathbf{P}_1$ of X_1. This is the distribution X_1 if we do *not* know the initial state (e.g. before we start the chain):

$$\mathbf{P}_1(j) = P(X_1 = j) = \sum_{i \in \mathcal{X}} P(X_1 = j \,|\, X_0 = i)\mathbf{P}_{\text{init}}(i) = \sum_{i \in \mathcal{X}} \phi(j|i)\mathbf{P}_{\text{init}}(i) \ .$$

- Since Φ is a $K \times K$-matrix and $\mathbf{P}_{\text{init}}$ a vector of length K, we can write that as

$$\mathbf{P}_1 = \mathbf{P}_{\text{init}} \cdot \Phi \ .$$

- The same argument shows that $\mathbf{P}_2$ is given by $\mathbf{P}_2 = \mathbf{P}_1 \cdot \Phi = \mathbf{P}_{\text{init}} \cdot \Phi \cdot \Phi$ et cetera.

The marginal distribution of the chain's state after t steps is

$$\mathbf{P}_t = \mathbf{P}_{\text{init}}\Phi^t$$

Limiting distribution

Instead of considering $\mathbf{P}_t$ for a specific, large t, we take the limit

$$\mathbf{P}_\infty := \lim_{t\to\infty} \mathbf{P}_t = \lim_{t\to\infty} \mathbf{P}_{\text{init}}\Phi^t \ ,$$

provided that the limit exists.

Observation

If the limit $\mathbf{P}_\infty$ exists, then

$$\mathbf{P}_\infty \cdot \Phi = \left(\lim_{t\to\infty} \mathbf{P}_{\text{init}}\Phi^t\right) \cdot \Phi = \lim_{t\to\infty} \mathbf{P}_{\text{init}}\Phi^t = \mathbf{P}_\infty \ ,$$

which motivates the next definition.

Let Φ be the transition matrix of a Markov chain. A distribution $\mathbf{P}$ on $\mathcal{X}$ which is invariant under Φ in the sense that

$$\mathbf{P} \cdot \Phi = \mathbf{P}$$

is called an **equilibrium distribution** or **invariant distribution** of the chain.

Does a chain have an equilibrium?

A stationary Markov chain is **aperiodic** if the transition matrix has non-zero diagonal,

$$P(X_t = k \,|\, X_{t-1} = k) = \phi(k|k) > 0 \qquad \text{for all } k \in \mathcal{X} \ .$$

It is **irreducible** if there is a path (with non-zero probability) from each state to every other state in the transition graph.

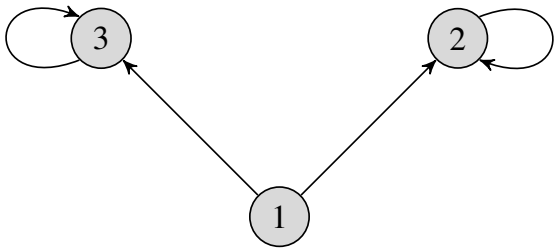
The actual definition of aperiodic is a bit more complicated. The property above is a sufficient condition for aperiodicity that is sufficient for our purposes.

Theorem

If a first-order, stationary Markov chain with $K < \infty$ states is aperiodic and irreducible, then:

- The limit distribution $\mathbf{P}_\infty$ exists.
- The limit distribution is also the equlibrium distribution.
- The equilibrium distribution is unique.
- The equilibrium does not depend on the initial distribution.

Why an equilibrium distribution may not be unique



- For this chain, both $\mathbf{P} = (0, 1, 0)$ and $\mathbf{P}' = (0, 0, 1)$ are valid equilibria.
- Which one emerges depends on the initial state and (if we start in state 1) on the first transition.

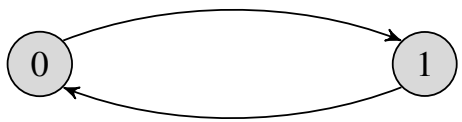
Remedy: Irreducibility. An irreducible chain could move between states 2 and 3.

Why the limit may not exist

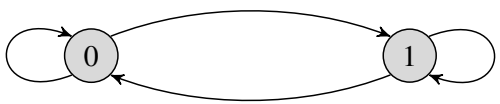
Recall that a sequence in $\mathbb{R}$ does not have a limit if it "oscillates". For example, $\lim_n 1^n = 1$, but $\lim_n (-1)^n$ does not exist.

- The chain on the right has no limit distribution.
- If we start e.g. in state 0, then 0 can only be reached in even steps, 1 only in odd steps.
- The distribution $\mathbf{P}_t$ oscillates between

$$\mathbf{P}_{\text{even}} = (1, 0) \qquad \text{and} \qquad \mathbf{P}_{\text{odd}} = (0, 1) \ .$$



Remedy: Require the chain to be aperiodic. That would add two edges to the graph:



The theorem shows that these simple fixes to the two most obvious problems suffice to ensure existence of a unique equilibrium.

THE PERRON–FROBENIUS THEOREM

Theorem Let A be a $K \times K$ matrix with entries $A_{ij} \geq 0$, and let $\rho(A) := \max_j |\lambda_j|$ be its largest absolute eigenvalue. If A^m has strictly positive entries for some $m \in \mathbb{N}$, then:

- $\rho(A)$ is itself an eigenvalue of A, and all other eigenvalues satisfy $|\lambda| < \rho(A)$.
- Its left and right eigenvectors are unique up to scaling, and can be chosen with all entries positive.

When do eigenvectors form a basis?

- If all K eigenvalues are distinct, a basis consisting of eigenvectors exists.
- This is typically the case for the transition matrices we work with. A chain satisfying detailed balance (discussed later), always has an eigenvector basis.

Otherwise, one can still draw similar conclusions as below using the Jordan normal form, but we will not be concerned with this.

If $A = \Phi$ is a transition matrix

- The rows sum to 1, so $\Phi\mathbf{1} = \mathbf{1}$, and no eigenvalue exceeds 1 in absolute value, so $\rho(\Phi) = 1$.
- If the chain is irreducible and aperiodic, some Φ^m is strictly positive.

Perron–Frobenius theorem for Markov chains. If the transition matrix Φ of a chain on K states is irreducible and aperiodic, then:

- 1 is an eigenvalue of Φ, and all other eigenvalues satisfy $|\lambda| < 1$.
- Its left eigenvector is unique up to scaling and has positive entries. Normalized to sum 1, it is the equilibrium $\mathbf{P}_{\text{inv}}$.

Why $\mathbf{P}_t$ converges

Expand $\mathbf{P}_{\text{init}} = \sum_j c_j \xi_j$ in left eigenvectors $\xi_1 = \mathbf{P}_{\text{inv}}, \xi_2, \ldots, \xi_K$ of Φ (assuming they form a basis). Then

$$\mathbf{P}_t = \mathbf{P}_{\text{init}} \Phi^t = \sum_{j=1}^{K} c_j \lambda_j^t \xi_j = \mathbf{P}_{\text{inv}} + \sum_{j=2}^{K} c_j \lambda_j^t \xi_j \xrightarrow{t \to \infty} \mathbf{P}_{\text{inv}} \ .$$

The error shrinks like $|\lambda_2|^t$.

Recall: $A^n \mathbf{v}$ for a symmetric matrix

- As in the symmetric case, the eigenvalue of largest absolute value dominates.
- For $A^n \mathbf{v}$ only the *direction* converges. Here, $\mathbf{P}_t$ itself converges since $\lambda_1 = 1$.
- Without the conditions: periodic chains have eigenvalue -1 (oscillation), reducible ones can have several equilibria.

Power method

If the the transition matrix Φ makes the chain irreducible and aperiodic, we know that

$$\text{equilibrium distribution} = \text{limit distribution} \quad .$$

This means we can approximate the equilibrium $\mathbf{P}_\infty$ by $\mathbf{P}_t$:

- Initialize with any distribution $\mathbf{P}_{\text{init}}$ (e.g. uniform).
- Repeat $\mathbf{P}_{t+1} = \mathbf{P}_t \cdot \Phi$.
- Terminate once $\|\mathbf{P}_{t+1} - \mathbf{P}_t\| < \tau$ for some small τ.

The link structure of the web is represented by the **web graph**, defined by the adjacency matrix

$$A = (A_{ij})_{i,\, j \leq \#\text{pages}} \qquad \text{where} \qquad A_{ij} = \begin{cases} 1 & \text{page } i \text{ links to page } j \\ 0 & \text{otherwise} \end{cases}$$

Vertices represent pages, edges represent links. The graph is directed.

Simple random walk

Let G be a directed graph with K vertices. Generate a sequence $X_0, X_1, \ldots$ of vertices:

- Select a vertex X_0 in G uniformly at random.
- For $t = 1, 2, \ldots$, select X_t uniformly at random from the children of X_{t-1} in the graph.

That defines a Markov chain whose state space is the vertex set of the graph, with

$$\mathbf{P}_{\text{init}} = \left(\tfrac{1}{K}, \ldots, \tfrac{1}{K}\right) \qquad \text{and} \qquad \phi(j|i) = \begin{cases} \frac{1}{\#\text{ edges out of } i} & \text{if } i \text{ links to } j \\ 0 & \text{otherwise} \end{cases}$$

This Markov chain is called **simple random walk** on G.

The PageRank Algorithm

(Approximately) compute $\mathbf{P}_\infty$ for simple random walk on the web graph.

Problem: Ranking queries

The first step in internet search is query matching:

- The user enters a search query (a string of words).
- The search engine determines all web pages in its database which match the query.
- This is a large set (typically tens or hundreds of millions). The results are only useful if the "best" links can be filtered out by the engine.

Available data

- Using a web crawler, we can (approximately) determine the link structure of the internet.
- We can determine which pages there are, and which pages they link to.
- We cannot determine how often a link is followed, or how often a page is visited.

PageRank solution

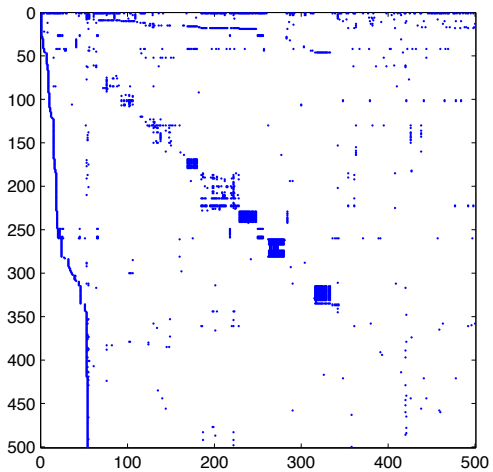
- PageRank uses $\mathbf{P}_\infty(v)$ as the score of web page v. It ranks by decreasing score.
- This uses (i) popularity as a proxy for quality/usefulness, and (ii) the amount of incoming links as a proxy for popularity.
- $\mathbf{P}_1$ would measure how often a page is linked. $\mathbf{P}_2$ weights this by how often the linking page is linked, etc.
- The PageRank paper interpreted $\mathbf{P}_\infty(v)$ as the probability that "random web surfer" would end up on page v after randomly following a large number of links. Note the start page does not matter for $t \to \infty$.

- Simple random walk on the web graph has transition matrix T with

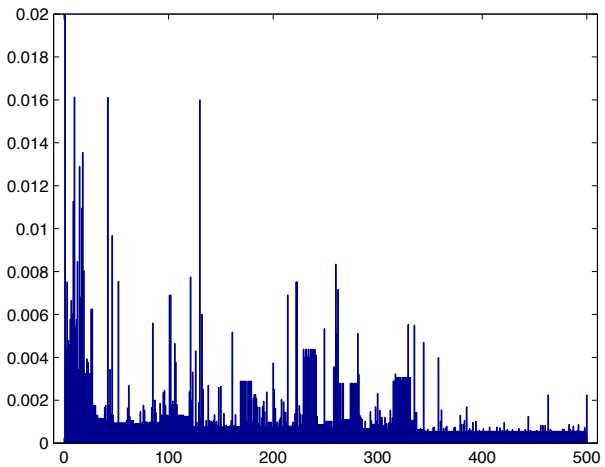
$$T_{ij} := \begin{cases} \frac{1}{\#\text{ edges out of } i} & \text{if } i \text{ links to } j \\ 0 & \text{otherwise} \end{cases}$$

  This is typically not irreducible (think of web pages which do not link anywhere).
- PageRank therefore uses the regularized transition matrix $\Phi_{ij} := (1-\alpha)T_{ij} + \frac{\alpha}{K}$, for some small $\alpha \in (0,1)$. Clearly, this makes Φ both irreducible and aperiodic.
- The equilibrium can be approximated by the power method. Since the web changes, it can be re-run every few days with the previous equilibrium as initial distribution.

Example



Adjacence matrix of the web graph of 500 web pages. The root (index 0) is www.harvard.edu.



Equilibrium distribution computed by PageRank.

Illustration: See K. Murphy, "Machine Learning", MIT Press 2012.

LATENT VARIABLE MODELS FOR TIME SERIES

MODELLING TIME SERIES

We will now consider a class of models for sequential observations

$$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_t$$

that are *not* well-represented by independent variables.

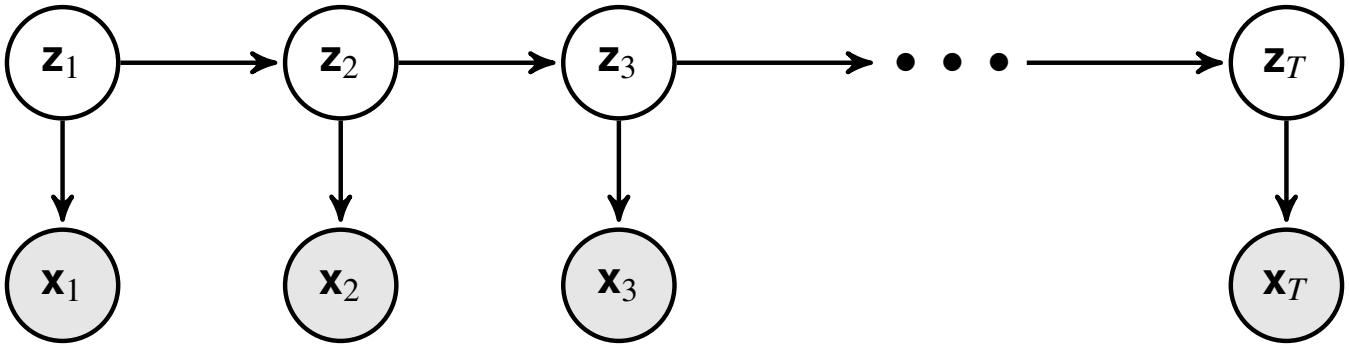
Examples:

- Sequence of images
- Stock prices
- Recorded speech or music, English sentences
- Kinematic variables in a robot
- Sensor readings from an industrial process
- Amino acids, DNA, etc. . .

Goal: To build a joint probabilistic model of the data $p(\mathbf{x}_1, \ldots, \mathbf{x}_t)$.

- Predict $p(\mathbf{x}_t | \mathbf{x}_1, \ldots, \mathbf{x}_{t-1})$
- Detect abnormal/changed behaviour (if $p(\mathbf{x}_t, \mathbf{x}_{t+1}, \ldots | \mathbf{x}_1, \ldots, \mathbf{x}_{t-1})$ small)
- Recover underlying/latent/hidden causes linking entire sequence

CAUSAL STRUCTURE AND LATENT VARIABLES



Temporal dependence captured by latents, with observations conditionally independent.
Speech recognition:

- $\mathbf{z}$ - underlying phonemes or words
- $\mathbf{x}$ - acoustic waveform

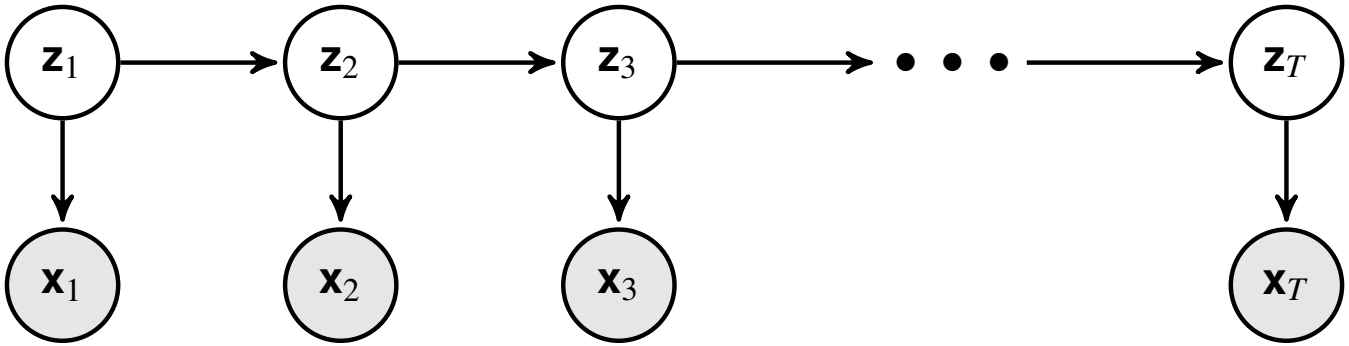
Vision:

- $\mathbf{z}$ - object identities, poses, illumination
- $\mathbf{x}$ - image pixel values

Industrial Monitoring:

- $\mathbf{z}$ - current state of molten steel in caster
- $\mathbf{x}$ - temperature and pressure sensor readings

LATENT-CHAIN MODELS



Joint probability factorizes:

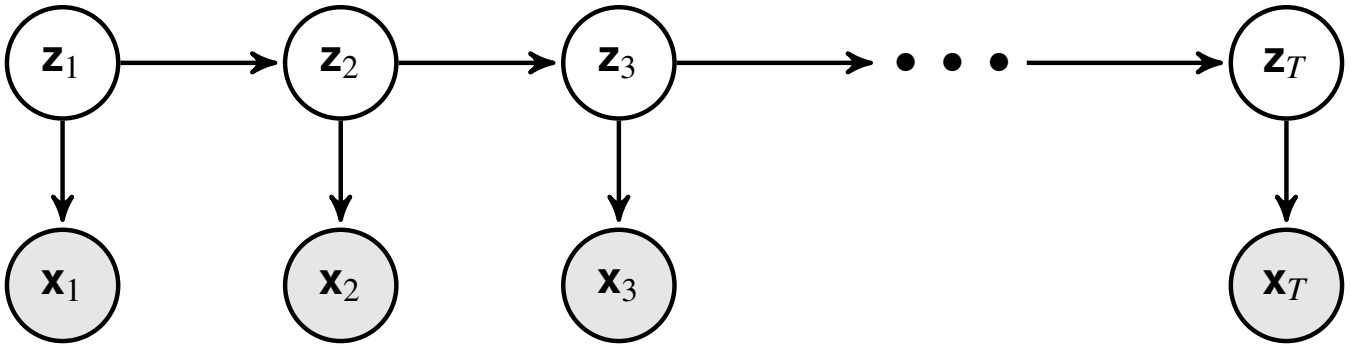
$$P(\mathbf{z}_{1:T}, \mathbf{x}_{1:T}) = P(\mathbf{z}_1)P(\mathbf{x}_1|\mathbf{z}_1)\prod_{t=2}^{T} P(\mathbf{z}_t|\mathbf{z}_{t-1})P(\mathbf{x}_t|\mathbf{z}_t)$$

where $\mathbf{z}_t$ and $\mathbf{x}_t$ are both real-valued vectors. We use the notation $\square_{1:T} \equiv \square_1, \ldots, \square_T$.

Two frequently-used tractable models:

- Linear-Gaussian state-space models (latents are Gaussian)
- Hidden Markov models (latents are discrete)

LINEAR-GAUSSIAN STATE-SPACE MODELS (SSMS)



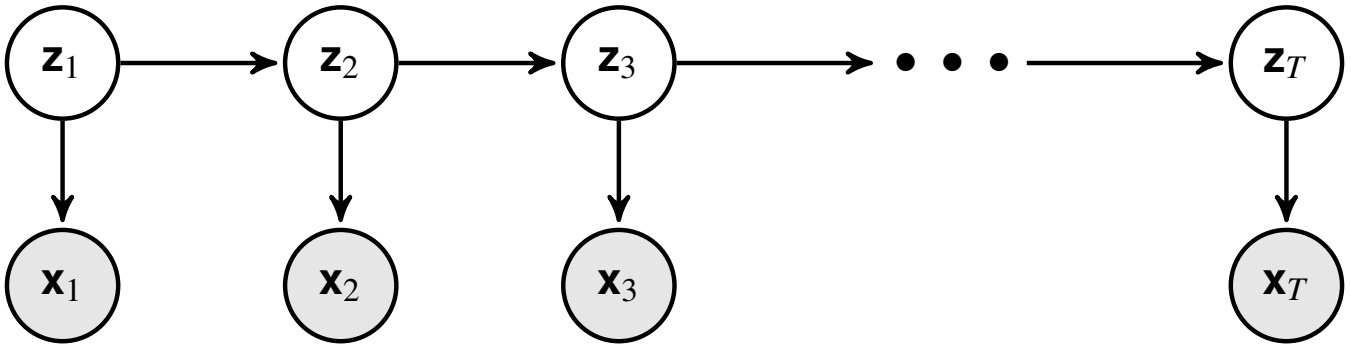
In a linear Gaussian SSM all conditional distributions are linear and Gaussian:

Output equation:	$\mathbf{x}_t = C\mathbf{z}_t + \mathbf{v}_t$
State dynamics equation:	$\mathbf{z}_t = A\mathbf{z}_{t-1} + \mathbf{w}_t$

where $\mathbf{v}_t$ and $\mathbf{w}_t$ are uncorrelated zero-mean multivariate Gaussian noise vectors.

Also assume $\mathbf{z}_1$ is multivariate Gaussian. The joint distribution over all variables $\mathbf{x}_{1:T}, \mathbf{z}_{1:T}$ is (one big) multivariate Gaussian.

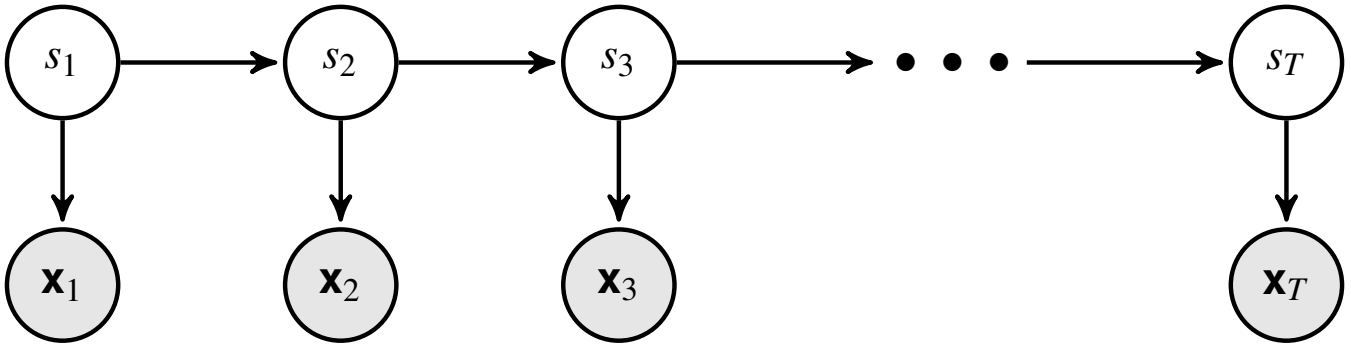
These models are also known as stochastic linear dynamical systems, Kalman filter models.



Interpretation 2:

- Markov chain with linear dynamics $\mathbf{z}_t = A\mathbf{z}_{t-1} \ldots$
- $\ldots$ perturbed by Gaussian innovations noise – may describe stochasticity, unknown control, or model mismatch.
- Observations are a linear projection of the dynamical state, with additive iid Gaussian noise.
- Note:
  - Dynamical process ($\mathbf{z}_t$) may be higher dimensional than the observations ($\mathbf{x}_t$).
  - Observations do not form a Markov chain – longer-scale dependence reflects/reveals latent dynamics.

HIDDEN MARKOV MODELS



Discrete hidden states $s_t \in \{1 \ldots, K\}$; outputs $\mathbf{x}_t$ can be discrete or continuous.
Joint probability factorizes:

$$P(s_{1:T}, \mathbf{x}_{1:T}) = P(s_1)P(\mathbf{x}_1|s_1) \prod_{t=2}^{T} P(s_t|s_{t-1})P(\mathbf{x}_t|s_t)$$

Generative process:

- A first-order Markov chain generates the hidden state sequence (path):

  initial state probs: $\pi_j = P(s_1 = j)$ transition matrix: $\Phi_{ij} = P(s_{t+1} = j|s_t = i)$

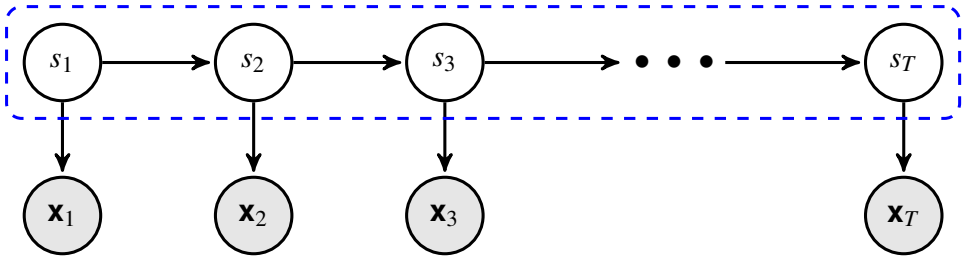
- A set of emission (output) distributions $A_j(\cdot)$ (one per state) converts state path to a sequence of observations $\mathbf{x}_t$.

$$A_j(\mathbf{x}) = P(\mathbf{x}_t = \mathbf{x}|s_t = j) \qquad \text{(for continuous } \mathbf{x}_t\text{)}$$
$$A_{jk} = P(\mathbf{x}_t = k|s_t = j) \qquad \text{(for discrete } \mathbf{x}_t\text{)}$$

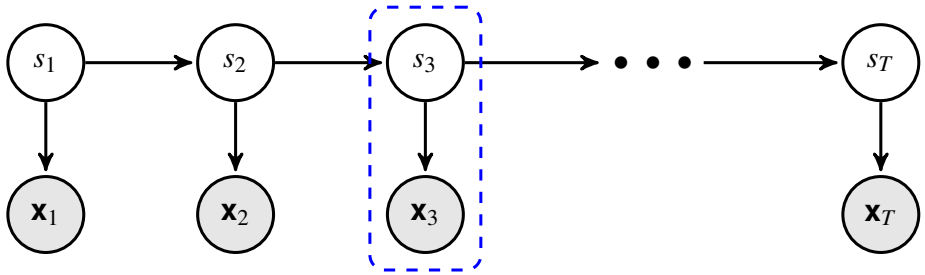
HIDDEN MARKOV MODELS

Two interpretations:

- a Markov chain with stochastic measurements:



- or a mixture model with states coupled across time:



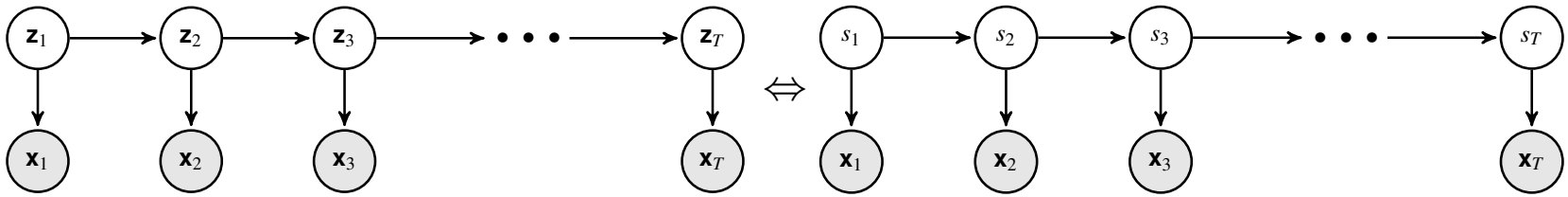
Even though hidden state sequence is first-order Markov, the output process may not be Markov of any order (for example: 1111121111311121111131 . . .).

Discrete state, discrete output models can approximate any continuous dynamics and observation mapping even if nonlinear; however this is usually not practical.

HMMs are related to stochastic finite state machines/automata.

HMMs and SSMs

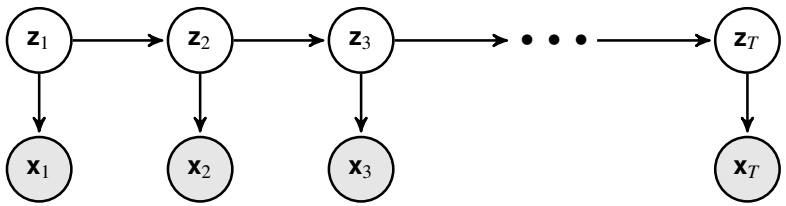
(Linear Gaussian) State space models are the continuous state analogue of hidden Markov models.



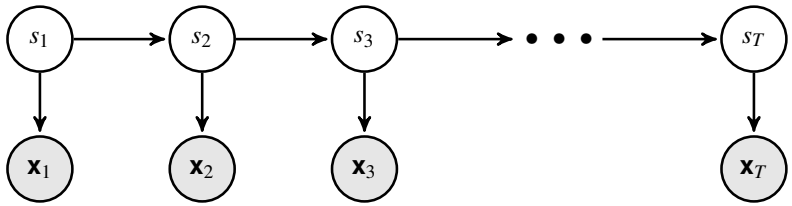
- A continuous vector state is a very powerful representation.
  For an HMM to communicate N bits of information about the past, it needs 2^N states! But a real-valued state vector can store an arbitrary number of bits in principle.

- Linear-Gaussian output/dynamics are very weak.
  The types of dynamics linear SSMs can capture is very limited. HMMs can in principle represent arbitrary stochastic dynamics and output mappings.

CHAIN MODELS: ML LEARNING WITH EM



$$\begin{aligned}\mathbf{z}_1 &\sim \mathcal{N}(\mu_0, Q_0)\\ \mathbf{z}_t|\mathbf{z}_{t-1} &\sim \mathcal{N}(A\mathbf{z}_{t-1}, Q)\\ \mathbf{x}_t|\mathbf{z}_t &\sim \mathcal{N}(C\mathbf{z}_t, R)\end{aligned}$$



$$\begin{aligned}s_1 &\sim \pi\\ s_t|s_{t-1} &\sim \Phi_{s_{t-1},\cdot}\\ \mathbf{x}_t|s_t &\sim \mathsf{A}_{s_t}\end{aligned}$$

The structure of learning and inference for both models is dictated by the factored structure.

$$P(\mathbf{x}_1, \ldots, \mathbf{x}_T, \mathbf{z}_1, \ldots, \mathbf{z}_T) = P(\mathbf{z}_1) \prod_{t=2}^{T} P(\mathbf{z}_t|\mathbf{z}_{t-1}) \prod_{t=1}^{T} P(\mathbf{x}_t|\mathbf{z}_t)$$

Learning (M-step):

$$\arg\max \langle \log P(\mathbf{x}_1, \ldots, \mathbf{x}_T, \mathbf{z}_1, \ldots, \mathbf{z}_T) \rangle_{q(\mathbf{z}_1, \ldots, \mathbf{z}_T)} =$$

$$\arg\max \left[\langle \log P(\mathbf{z}_1) \rangle_{q(\mathbf{z}_1)} + \sum_{t=2}^{T} \langle \log P(\mathbf{z}_t|\mathbf{z}_{t-1}) \rangle_{q(\mathbf{z}_t, \mathbf{z}_{t-1})} + \sum_{t=1}^{T} \langle \log P(\mathbf{x}_t|\mathbf{z}_t) \rangle_{q(\mathbf{z}_t)} \right]$$

So the expectations needed (in E-step) are derived from singleton and pairwise marginals.

CHAIN MODELS: INFERENCE

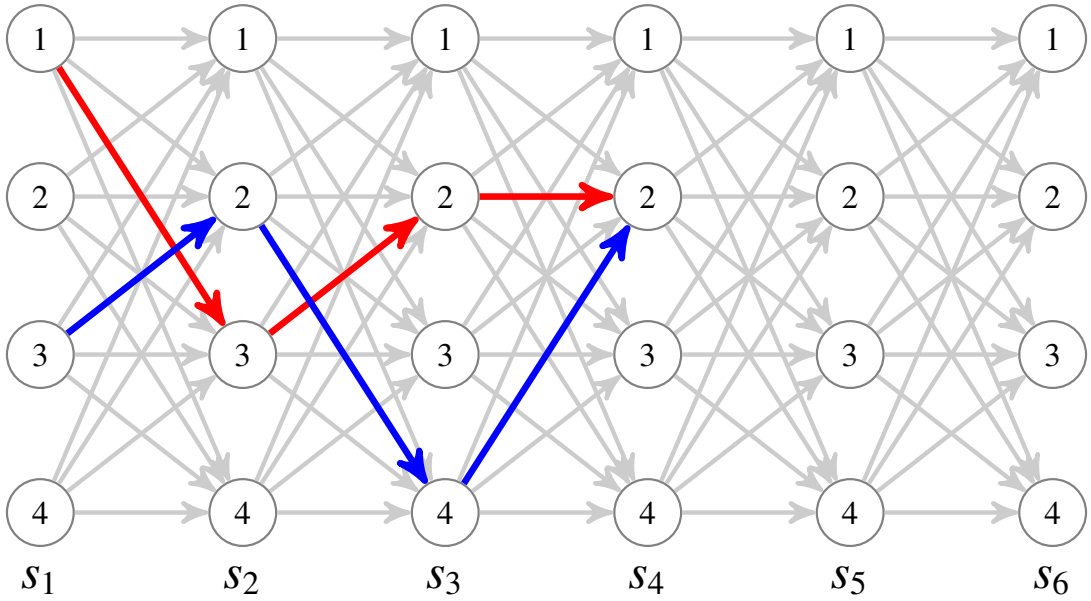
Three general inference problems:

Filtering:	$P(\mathbf{z}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_t)$	
Smoothing:	$P(\mathbf{z}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_T)$	(also $P(\mathbf{z}_t, \mathbf{z}_{t-1} \mid \mathbf{x}_1, \ldots, \mathbf{x}_T)$ for learning)
Prediction:	$P(\mathbf{z}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_{t-\triangle t})$	

Naively, these marginal posteriors seem to require very large integrals (or sums)

$$P(\mathbf{z}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_t) = \int \cdots \int d\mathbf{z}_1 \ldots d\mathbf{z}_{t-1} \; P(\mathbf{z}_1, \ldots, \mathbf{z}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_t)$$

but again the factored structure of the distributions will help us. The algorithms rely on a form of temporal updating or message passing.



Consider an HMM, where we want to find $P(s_t = k|\mathbf{x}_1 \dots \mathbf{x}_t) =$

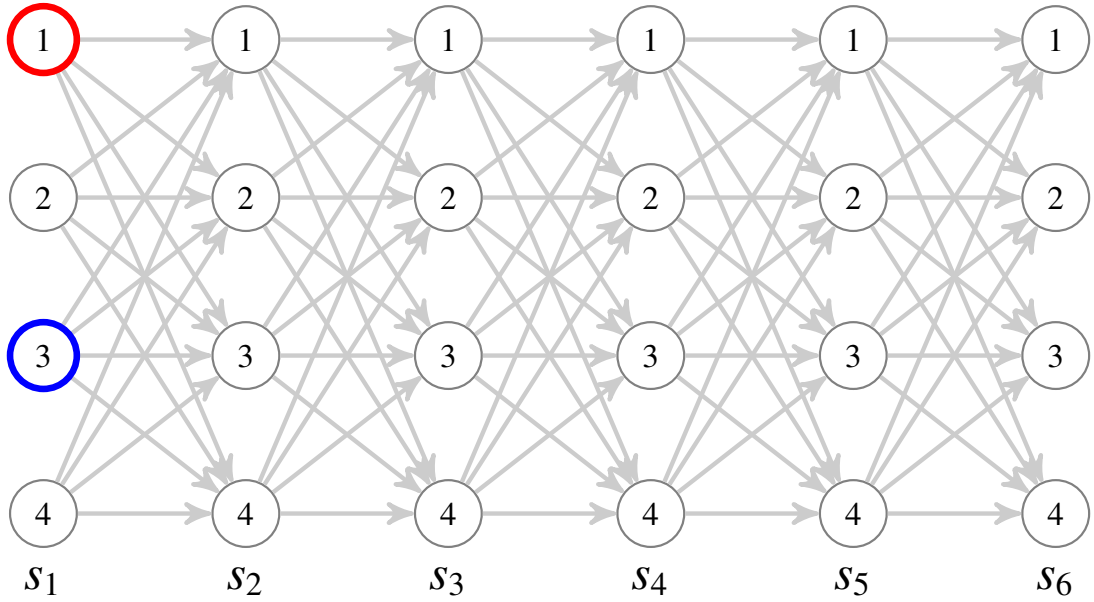
$$\sum_{k_1,\dots,k_{t-1}} P(s_1 = k_1, \dots, s_t = k|\mathbf{x}_1 \dots \mathbf{x}_t) \propto \sum_{k_1,\dots,k_{t-1}} \pi_{k_1} A_{k_1}(\mathbf{x}_1) \Phi_{k_1,k_2} A_{k_2}(\mathbf{x}_2) \dots \Phi_{k_{t-1},k} A_k(\mathbf{x}_t)$$

Naïve algorithm:

- place a pebble at each of the $k_1 = 1 \dots K$ states at $t = 1$ holding value $\pi_{k_1} A_{k_1}(\mathbf{x}_1)$
- move each pebble forward in time: make copies of each pebble to each subsequent state and multiply the value of each copy by transition prob. $\times$ output emission prob.
- repeat until all pebbles have reached time t
- sum up values on all K^{t-1} pebbles that reach state $s_t = k$ (one pebble per state path)

Clever recursion:

- at every step, replace pebbles at each node with a single pebble carrying sum of values



Consider an HMM, where we want to find $P(s_t = k|\mathbf{x}_1 \dots \mathbf{x}_t) =$

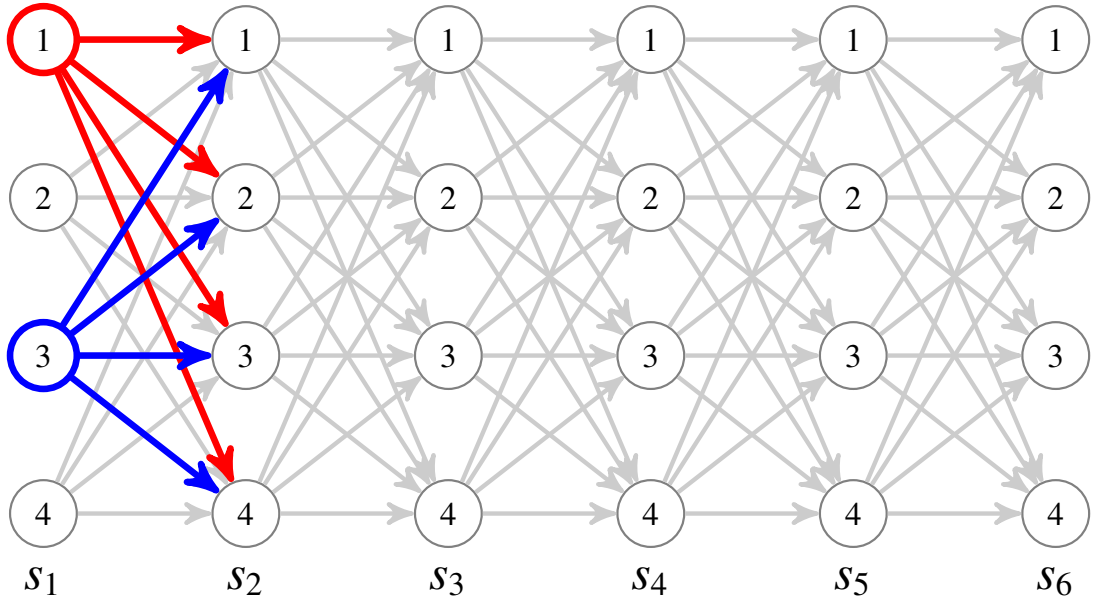
$$\sum_{k_1,\dots,k_{t-1}} P(s_1 = k_1, \dots, s_t = k|\mathbf{x}_1 \dots \mathbf{x}_t) \propto \sum_{k_1,\dots,k_{t-1}} \pi_{k_1} A_{k_1}(\mathbf{x}_1) \Phi_{k_1,k_2} A_{k_2}(\mathbf{x}_2) \dots \Phi_{k_{t-1},k} A_k(\mathbf{x}_t)$$

Naïve algorithm:

- place a pebble at each of the $k_1 = 1 \dots K$ states at $t = 1$ holding value $\pi_{k_1} A_{k_1}(\mathbf{x}_1)$
- move each pebble forward in time: make copies of each pebble to each subsequent state and multiply the value of each copy by transition prob. $\times$ output emission prob.
- repeat until all pebbles have reached time t
- sum up values on all K^{t-1} pebbles that reach state $s_t = k$ (one pebble per state path)

Clever recursion:

- at every step, replace pebbles at each node with a single pebble carrying sum of values



Consider an HMM, where we want to find $P(s_t = k|\mathbf{x}_1 \dots \mathbf{x}_t) =$

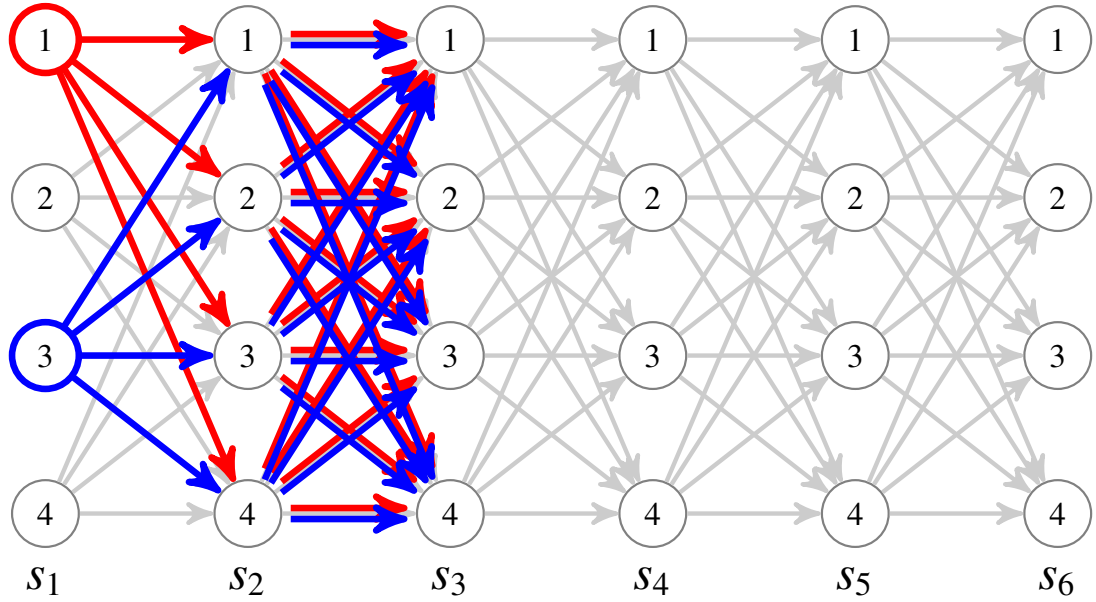
$$\sum_{k_1,\dots,k_{t-1}} P(s_1 = k_1, \dots, s_t = k|\mathbf{x}_1 \dots \mathbf{x}_t) \propto \sum_{k_1,\dots,k_{t-1}} \pi_{k_1} A_{k_1}(\mathbf{x}_1) \Phi_{k_1,k_2} A_{k_2}(\mathbf{x}_2) \dots \Phi_{k_{t-1},k} A_k(\mathbf{x}_t)$$

Naïve algorithm:

- place a pebble at each of the $k_1 = 1 \dots K$ states at $t = 1$ holding value $\pi_{k_1} A_{k_1}(\mathbf{x}_1)$
- move each pebble forward in time: make copies of each pebble to each subsequent state and multiply the value of each copy by transition prob. $\times$ output emission prob.
- repeat until all pebbles have reached time t
- sum up values on all K^{t-1} pebbles that reach state $s_t = k$ (one pebble per state path)

Clever recursion:

- at every step, replace pebbles at each node with a single pebble carrying sum of values



Consider an HMM, where we want to find $P(s_t = k|\mathbf{x}_1 \dots \mathbf{x}_t) =$

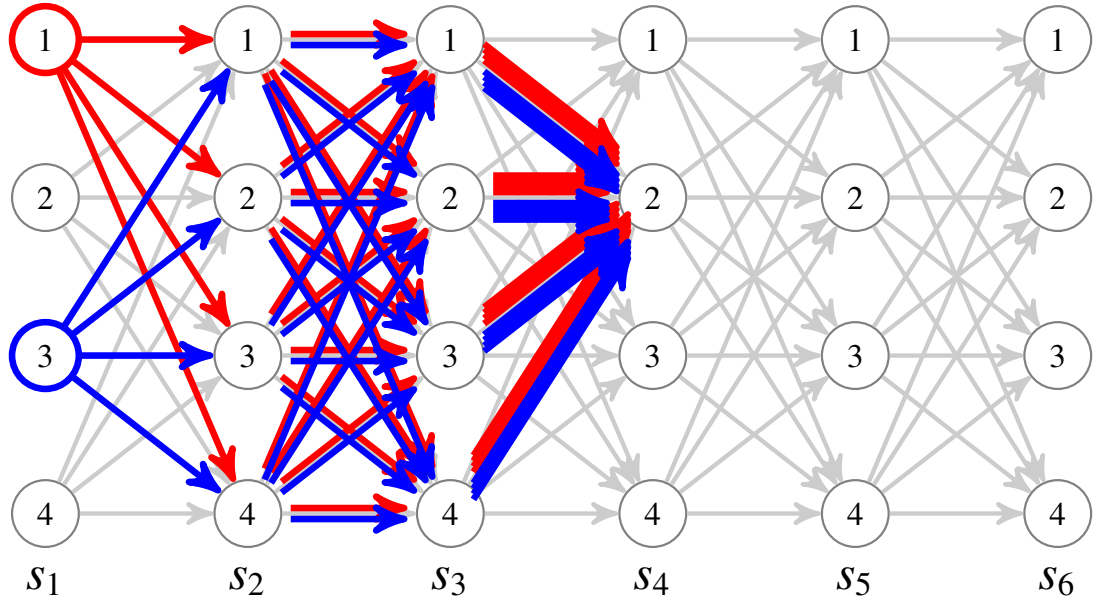
$$\sum_{k_1,\dots,k_{t-1}} P(s_1 = k_1, \dots, s_t = k|\mathbf{x}_1 \dots \mathbf{x}_t) \propto \sum_{k_1,\dots,k_{t-1}} \pi_{k_1} A_{k_1}(\mathbf{x}_1) \Phi_{k_1,k_2} A_{k_2}(\mathbf{x}_2) \dots \Phi_{k_{t-1},k} A_k(\mathbf{x}_t)$$

Naïve algorithm:

- place a pebble at each of the $k_1 = 1 \dots K$ states at $t = 1$ holding value $\pi_{k_1} A_{k_1}(\mathbf{x}_1)$
- move each pebble forward in time: make copies of each pebble to each subsequent state and multiply the value of each copy by transition prob. $\times$ output emission prob.
- repeat until all pebbles have reached time t
- sum up values on all K^{t-1} pebbles that reach state $s_t = k$ (one pebble per state path)

Clever recursion:

- at every step, replace pebbles at each node with a single pebble carrying sum of values



Consider an HMM, where we want to find $P(s_t = k|\mathbf{x}_1 \dots \mathbf{x}_t) =$

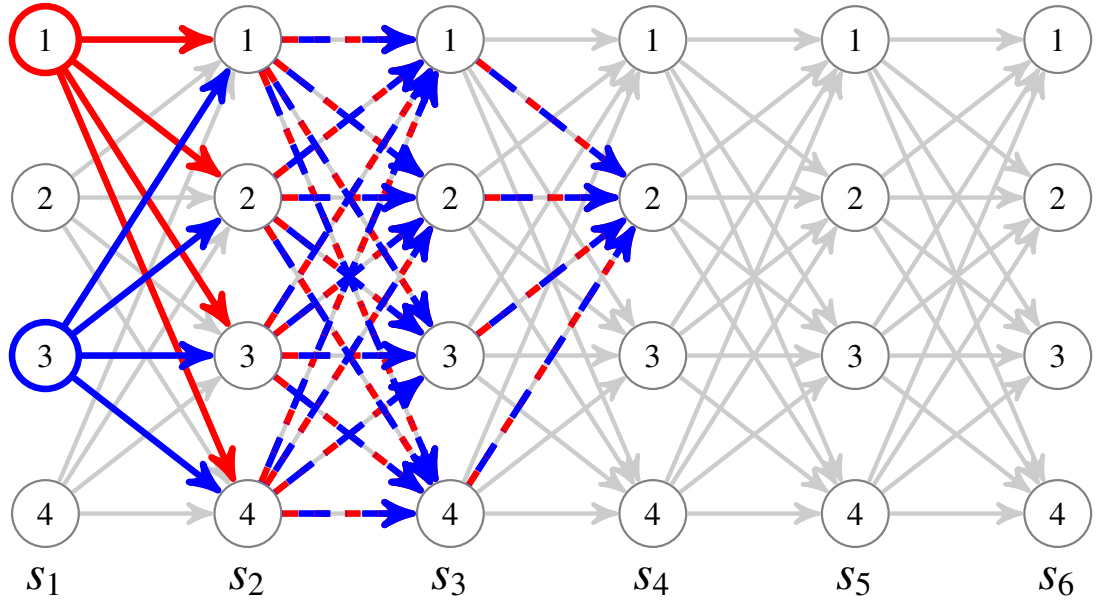
$$\sum_{k_1,\dots,k_{t-1}} P(s_1 = k_1, \dots, s_t = k|\mathbf{x}_1 \dots \mathbf{x}_t) \propto \sum_{k_1,\dots,k_{t-1}} \pi_{k_1} A_{k_1}(\mathbf{x}_1) \Phi_{k_1,k_2} A_{k_2}(\mathbf{x}_2) \dots \Phi_{k_{t-1},k} A_k(\mathbf{x}_t)$$

Naïve algorithm:

- place a pebble at each of the $k_1 = 1 \dots K$ states at $t = 1$ holding value $\pi_{k_1} A_{k_1}(\mathbf{x}_1)$
- move each pebble forward in time: make copies of each pebble to each subsequent state and multiply the value of each copy by transition prob. $\times$ output emission prob.
- repeat until all pebbles have reached time t
- sum up values on all K^{t-1} pebbles that reach state $s_t = k$ (one pebble per state path)

Clever recursion:

- at every step, replace pebbles at each node with a single pebble carrying sum of values



Consider an HMM, where we want to find $P(s_t = k|\mathbf{x}_1 \dots \mathbf{x}_t) =$

$$\sum_{k_1,\dots,k_{t-1}} P(s_1 = k_1, \dots, s_t = k|\mathbf{x}_1 \dots \mathbf{x}_t) \propto \sum_{k_1,\dots,k_{t-1}} \pi_{k_1} A_{k_1}(\mathbf{x}_1) \Phi_{k_1,k_2} A_{k_2}(\mathbf{x}_2) \dots \Phi_{k_{t-1},k} A_k(\mathbf{x}_t)$$

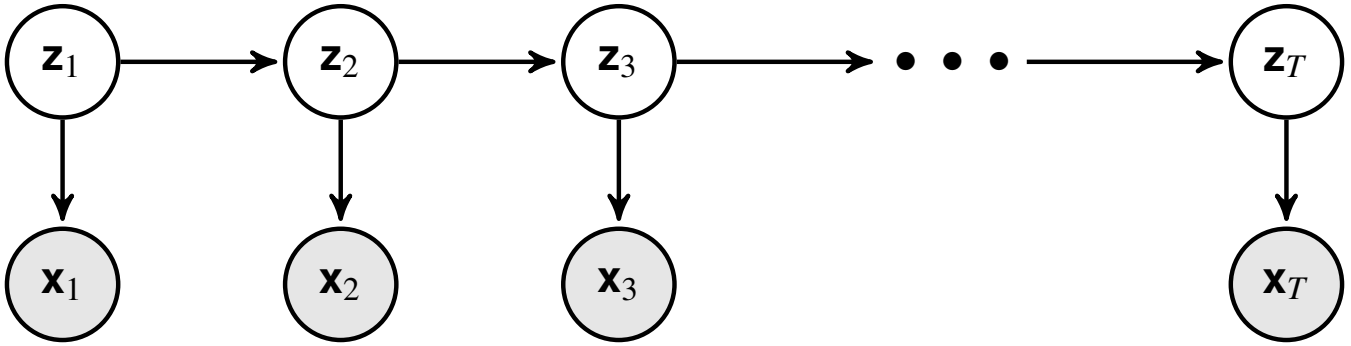
Naïve algorithm:

- place a pebble at each of the $k_1 = 1 \dots K$ states at $t = 1$ holding value $\pi_{k_1} A_{k_1}(\mathbf{x}_1)$
- move each pebble forward in time: make copies of each pebble to each subsequent state and multiply the value of each copy by transition prob. $\times$ output emission prob.
- repeat until all pebbles have reached time t
- sum up values on all K^{t-1} pebbles that reach state $s_t = k$ (one pebble per state path)

Clever recursion:

- at every step, replace pebbles at each node with a single pebble carrying sum of values

PROBABILITY UPDATING: "BAYESIAN FILTERING"



$$
\begin{aligned}
P(\mathbf{z}_t|\mathbf{x}_{1:t}) &= \int P(\mathbf{z}_t, \mathbf{z}_{t-1}|\mathbf{x}_t, \mathbf{x}_{1:t-1})\, d\mathbf{z}_{t-1} \\
&= \int \frac{P(\mathbf{x}_t, \mathbf{z}_t, \mathbf{z}_{t-1}|\mathbf{x}_{1:t-1})}{P(\mathbf{x}_t|\mathbf{x}_{1:t-1})}\, d\mathbf{z}_{t-1} \\
&\propto \int P(\mathbf{x}_t|\mathbf{z}_t, \mathbf{z}_{t-1}, \mathbf{x}_{1:t-1}) P(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:t-1}) P(\mathbf{z}_{t-1}|\mathbf{x}_{1:t-1})\, d\mathbf{z}_{t-1} \\
&= \int P(\mathbf{x}_t|\mathbf{z}_t) P(\mathbf{z}_t|\mathbf{z}_{t-1}) P(\mathbf{z}_{t-1}|\mathbf{x}_{1:t-1})\, d\mathbf{z}_{t-1}
\end{aligned}
$$

Markov property

This is a **forward recursion** based on Bayes rule.

THE HMM: FORWARD PASS

The **forward recursion** for the HMM is a form of **dynamic programming**. Define:

$$\alpha_t(i) = P(\mathbf{x}_1, \ldots, \mathbf{x}_t, s_t = i|\theta)$$

Then much like the Bayesian filtering updates, we have:

$$\alpha_1(i) = \pi_i A_i(\mathbf{x}_1) \qquad \alpha_{t+1}(i) = \left(\sum_{j=1}^{K} \alpha_t(j)\Phi_{ji} \right) A_i(\mathbf{x}_{t+1})$$

We've defined $\alpha_t(i)$ to be a joint rather than a posterior. It's easy to obtain the posterior by normalization:

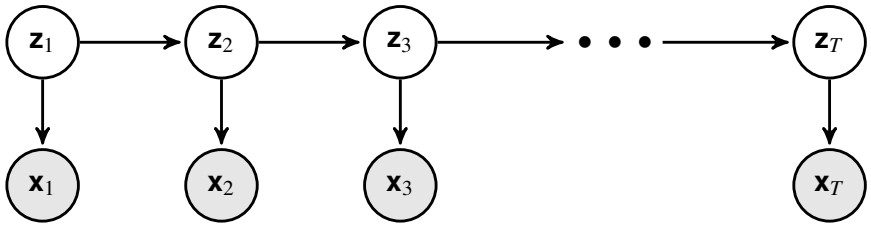
$$P(s_t = i|\mathbf{x}_1, \ldots, \mathbf{x}_t, \theta) = \frac{\alpha_t(i)}{\sum_k \alpha_t(k)}$$

This form enables us to compute the likelihood for $\theta = \{\mathsf{A}, \Phi, \pi\}$ efficiently in $\mathcal{O}(TK^2)$ time:

$$P(\mathbf{x}_1 \ldots \mathbf{x}_T|\theta) = \sum_{s_1,\ldots,s_T} P(\mathbf{x}_1, \ldots, \mathbf{x}_T, s_1, \ldots, s_T|\theta) = \sum_{k=1}^{K} \alpha_T(k)$$

avoiding the exponential number of paths in the naïve sum (number of paths = K^T).

THE LGSSM: KALMAN FILTERING



$$\begin{aligned} \mathbf{z}_1 &\sim \mathcal{N}(\mu_0, Q_0) \\ \mathbf{z}_t|\mathbf{z}_{t-1} &\sim \mathcal{N}(A\mathbf{z}_{t-1}, Q) \\ \mathbf{x}_t|\mathbf{z}_t &\sim \mathcal{N}(C\mathbf{z}_t, R) \end{aligned}$$

For the SSM, the sums become integrals. Let $\hat{\mathbf{z}}_1^0 = \mu_0$ and $\hat{V}_1^0 = Q_0$; then

$$P(\mathbf{z}_1|\mathbf{x}_1) = \mathcal{N}(\underbrace{\hat{\mathbf{z}}_1^0 + G_1(\mathbf{x}_1 - C\hat{\mathbf{z}}_i^0)}_{\hat{\mathbf{z}}_1^1}, \underbrace{\hat{V}_1^0 - G_1 C\hat{V}_1^0}_{\hat{V}_1^1}) \qquad G_1 = \hat{V}_1^0 C^\mathsf{T}(C\hat{V}_1^0 C^\mathsf{T} + R)^{-1}$$

In general, we define $\hat{\mathbf{z}}_t^\tau \equiv E[\mathbf{z}_t|\mathbf{x}_1, \ldots, \mathbf{x}_\tau]$ and $\hat{V}_t^\tau \equiv V[\mathbf{z}_t|\mathbf{x}_1, \ldots, \mathbf{x}_\tau]$. Then,

$$P(\mathbf{z}_t|\mathbf{x}_{1:t-1}) = \int d\mathbf{z}_{t-1} P(\mathbf{z}_t|\mathbf{z}_{t-1}) P(\mathbf{z}_{t-1}|\mathbf{x}_{1:t-1}) = \mathcal{N}(\underbrace{A\hat{\mathbf{z}}_{t-1}^{t-1}}_{\hat{\mathbf{z}}_t^{t-1}}, \underbrace{A\hat{V}_{t-1}^{t-1}A^\mathsf{T} + Q}_{\hat{V}_t^{t-1}})$$

$$P(\mathbf{z}_t|\mathbf{x}_{1:t}) = \mathcal{N}(\underbrace{\hat{\mathbf{z}}_t^{t-1} + G_t(\mathbf{x}_t - C\hat{\mathbf{z}}_t^{t-1})}_{\hat{\mathbf{z}}_t^t}, \underbrace{\hat{V}_t^{t-1} - G_t C\hat{V}_t^{t-1}}_{\hat{V}_t^t})$$

The matrix $G_t := (\hat{V}_t^{t-1}C^\mathsf{T})(C\hat{V}_t^{t-1}C^\mathsf{T} + R)^{-1} = \mathrm{Cov}[z_t, x_t \mid x_{1:t-1}]\,\mathrm{Cov}[x_t \mid x_{1:t-1}]$ is called the Kalman gain.

Conditioning within a multivariate Gaussian

Let (A, B) be a partition of the set $\{1, \ldots, d\}$ and let $\mathbf{X} = (\mathbf{X}_\mathrm{A}, \mathbf{X}_\mathrm{B})$ be a Gaussian random vector in $\mathbb{R}^d = \mathbb{R}^A \times \mathbb{R}^B$, with

$$\mathbb{E}[\mathbf{X}] = \begin{pmatrix} \mu_\mathrm{A} \\ \mu_\mathrm{B} \end{pmatrix} \qquad \text{and} \qquad \mathrm{Cov}[\mathbf{X}] = \begin{pmatrix} \Sigma_\mathrm{A} & \Sigma_\mathrm{AB} \\ \Sigma_\mathrm{AB}^t & \Sigma_\mathrm{B} \end{pmatrix} .$$

Then the conditional distribution of $\mathbf{X}_\mathrm{A} | (\mathbf{X}_\mathrm{B} = \mathbf{x}_\mathrm{B})$ is again Gaussian, with mean

$$\mathbb{E}[\mathbf{X}_\mathrm{A} | \mathbf{X}_\mathrm{B} = \mathbf{x}_\mathrm{B}] = \mu_\mathrm{A} + \Sigma_\mathrm{AB} \Sigma_\mathrm{B}^{-1} (\mathbf{x}_\mathrm{B} - \mu_\mathrm{B})$$

and covariance

$$\mathrm{Cov}[\mathbf{X}_\mathrm{A} | \mathbf{X}_\mathrm{B} = \mathbf{x}_\mathrm{B}] = \Sigma_\mathrm{A} - \Sigma_\mathrm{AB} \Sigma_\mathrm{B}^{-1} \Sigma_\mathrm{AB}^t \ .$$

Matrix inversion lemma (Woodbury identity)

For $k \leq n$, invertible matrices $A \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{k\times k}$, and any $B \in \mathbb{R}^{n\times k}$ and $D \in \mathbb{R}^{k\times n}$:

$$(A - \underset{n\times k}{B}\underset{k\times k}{C}\underset{k\times n}{D})^{-1} = A^{-1} + A^{-1}B(C^{-1} - DA^{-1}B)^{-1}DA^{-1}$$

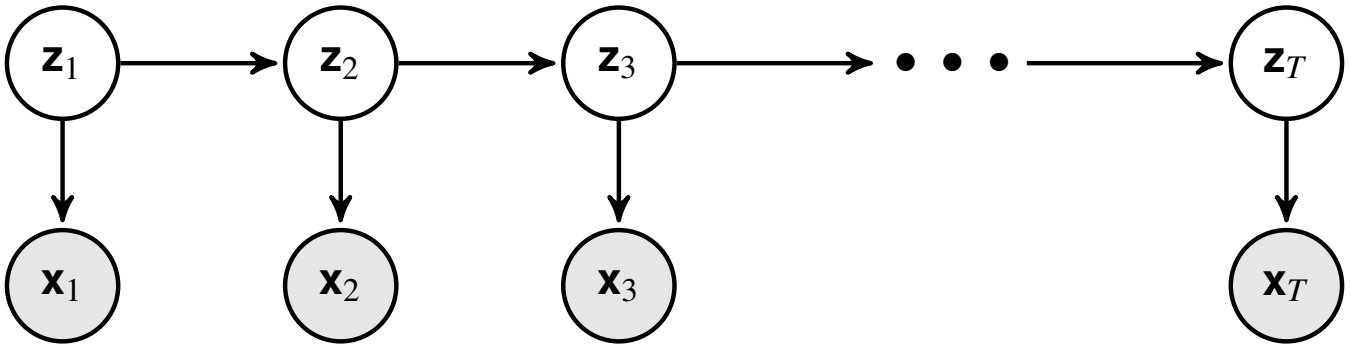
provided A and C are invertible.

Note: A and C refers to generic matrices, not to A and C defined on the Kalman filter slide.

Intepretation

- In essence: $(A - BCD)^{-1} = f(A^{-1}, C^{-1}, B, D)$
- Since C is $k \times k$, the $n \times n$-matrix BCD has rank $\leq k$.
- Substracting BCD modifies A on a k-dimensional subspace.
- Typical use case: We already know A^{-1}, and k is small. We can then invert the $n \times n$ matrix $(A - BCD)$ by inverting only a $k \times k$ matrix.

THE MARGINAL POSTERIOR: "BAYESIAN SMOOTHING"



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

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

The marginal combines a backward message with the forward message found by filtering.

The HMM: Forward–Backward Algorithm

State estimation: compute marginal posterior distribution over state at time t:

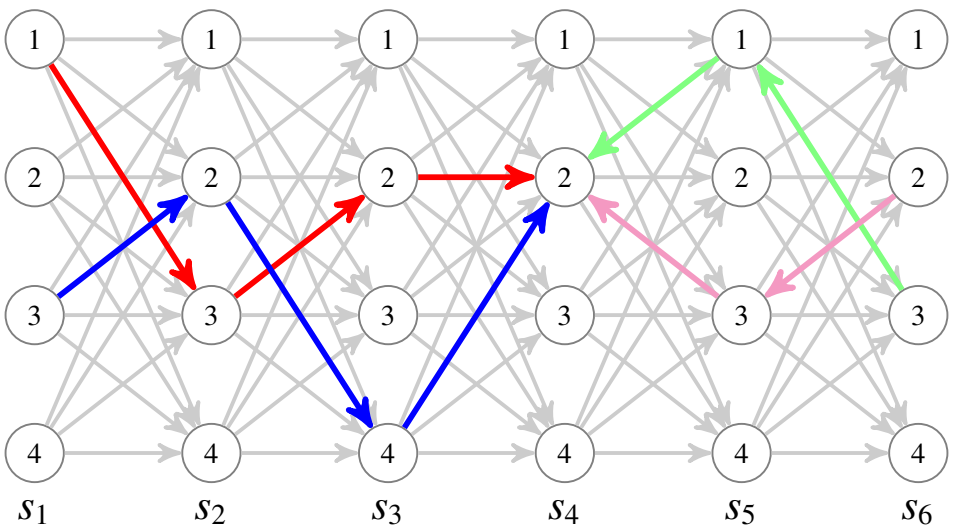
$$\gamma_t(i) \equiv P(s_t = i|\mathbf{x}_{1:T}) = \frac{P(s_t = i, \mathbf{x}_{1:t})P(\mathbf{x}_{t+1:T}|s_t = i)}{P(\mathbf{x}_{1:T})} = \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)}$$

where there is a simple backward recursion for

$$\beta_t(i) \equiv P(\mathbf{x}_{t+1:T}|s_t = i) = \sum_{j=1}^{K} P(s_{t+1} = j, \mathbf{x}_{t+1}, \mathbf{x}_{t+2:T}|s_t = i)$$

$$= \sum_{j=1}^{K} P(s_{t+1} = j|s_t = i)P(\mathbf{x}_{t+1}|s_{t+1} = j)P(\mathbf{x}_{t+2:T}|s_{t+1} = j) = \sum_{j=1}^{K} \Phi_{ij}A_j(\mathbf{x}_{t+1})\beta_{t+1}(j)$$

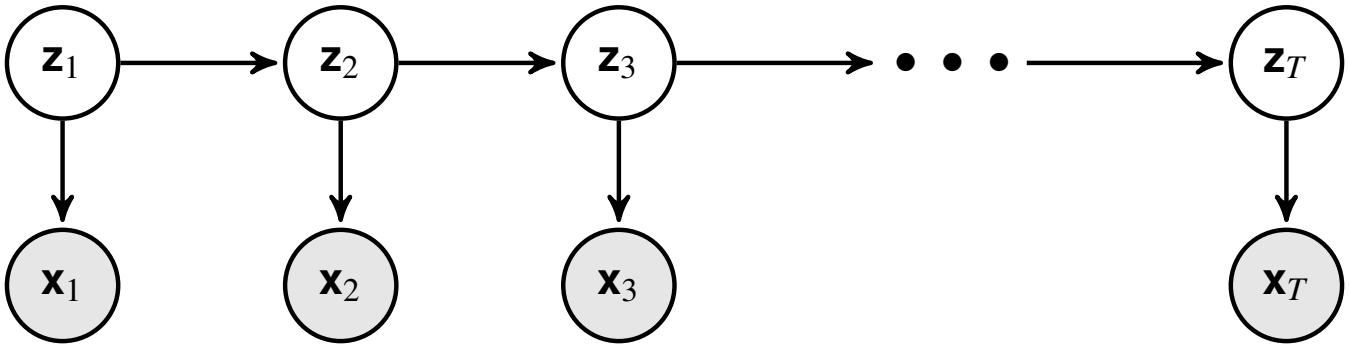
$\alpha_t(i)$ gives total *inflow* of probabilities to node (t, i); $\beta_t(i)$ gives total *outflow* of probabiilties.



VITERBI DECODING

- The numbers $\gamma_t(i)$ computed by forward-backward give the marginal posterior distribution over states at each time.
- By choosing the state i_t^* with the largest $\gamma_t(i)$ at each time, we can make a "best" state path. This is the path with the maximum expected number of correct states.
- But it is not the single path with the highest probability of generating the data.
  In fact it may be a path of probability zero!
- To find the single best path, we use the *Viterbi decoding algorithm* which is just Bellman's dynamic programming algorithm applied to this problem. This is an inference algorithm which computes the most probable state sequences: $\arg\max_{s_{1:T}} P(s_{1:T}|\mathbf{x}_{1:T}, \theta)$
- The recursions look the same as forward-backward, except with max instead of $\sum$.
- There is also a modified EM training based on the Viterbi decoder (assignment).

THE LGSSM: KALMAN SMOOTHING



We use a slightly different decomposition:

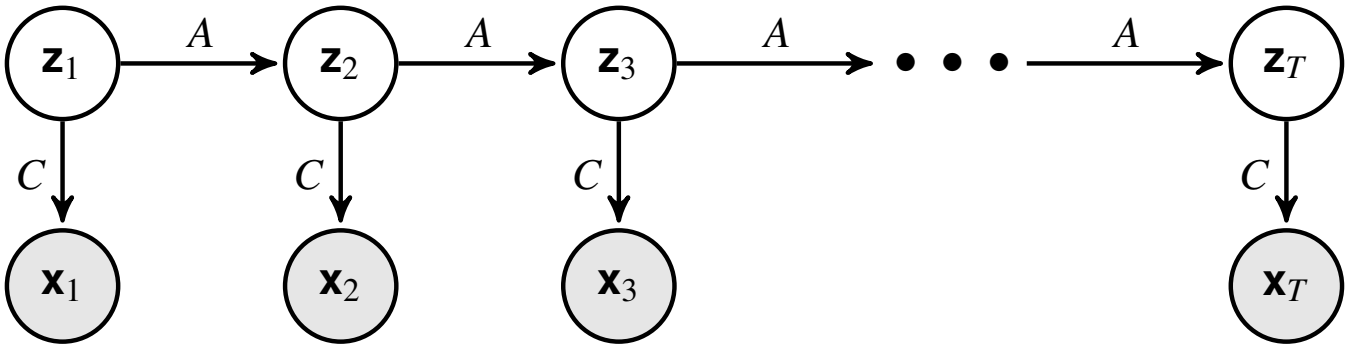
$$
\begin{aligned}
P(\mathbf{z}_t|\mathbf{x}_{1:T}) &= \int P(\mathbf{z}_t, \mathbf{z}_{t+1}|\mathbf{x}_{1:T})\, d\mathbf{z}_{t+1} \\
&= \int P(\mathbf{z}_t|\mathbf{z}_{t+1}, \mathbf{x}_{1:T}) P(\mathbf{z}_{t+1}|\mathbf{x}_{1:T})\, d\mathbf{z}_{t+1} \\
&= \int P(\mathbf{z}_t|\mathbf{z}_{t+1}, \mathbf{x}_{1:t}) P(\mathbf{z}_{t+1}|\mathbf{x}_{1:T})\, d\mathbf{z}_{t+1}
\end{aligned}
$$

Markov property

This gives the additional **backward recursion**:

$$
\begin{aligned}
J_t &= \hat{V}_t^t A^\mathsf{T} (\hat{V}_{t+1}^t)^{-1} \\
\hat{\mathbf{z}}_t^T &= \hat{\mathbf{z}}_t^t + J_t(\hat{\mathbf{z}}_{t+1}^T - A\hat{\mathbf{z}}_t^t) \\
\hat{V}_t^T &= \hat{V}_t^t + J_t(\hat{V}_{t+1}^T - \hat{V}_{t+1}^t) J_t^\mathsf{T}
\end{aligned}
$$

ML Learning for SSMs using batch EM



Parameters: $\theta = \{\mu_0, Q_0, A, Q, C, R\}$

Free energy:

$$\mathcal{F}(q, \theta) = \int d\mathbf{z}_{1:T}\; q(\mathbf{z}_{1:T})(\log P(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}|\theta) - \log q(\mathbf{z}_{1:T}))$$

E-step: Maximize $\mathcal{F}$ w.r.t. q with θ fixed: $q^*(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \theta)$

This is achieved by an extension of the Kalman smoother that returns the pairwise marginals $P(z_t, z_{t+1} \mid x_{1:T})$.

M-step: Maximize $\mathcal{F}$ w.r.t. θ with q fixed.

This boils down to solving a few weighted least squares problems, since all variables in

$$p(\mathbf{z}, \mathbf{x}|\theta) = p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1) \prod_{t=2}^{T} p(\mathbf{z}_t|\mathbf{z}_{t-1})p(\mathbf{x}_t|\mathbf{z}_t)$$

form a multivariate Gaussian.

THE M STEP FOR C

$$p(\mathbf{x}_t|\mathbf{z}_t) \propto \exp\left[-\tfrac{1}{2}(\mathbf{x}_t - C\mathbf{z}_t)^\mathsf{T} R^{-1}(\mathbf{x}_t - C\mathbf{z}_t)\right] \quad \Rightarrow$$

$$\begin{aligned}
C_{\text{new}} &= \arg\max_C \left\langle \sum_t \ln p(\mathbf{x}_t|\mathbf{z}_t) \right\rangle_q \\
&= \arg\max_C \left\langle -\frac{1}{2}\sum_t (\mathbf{x}_t - C\mathbf{z}_t)^\mathsf{T} R^{-1}(\mathbf{x}_t - C\mathbf{z}_t) \right\rangle_q + \text{const} \\
&= \arg\max_C \left\{ -\frac{1}{2}\sum_t \left(\mathbf{x}_t^\mathsf{T} R^{-1}\mathbf{x}_t - 2\mathbf{x}_t^\mathsf{T} R^{-1} C\langle \mathbf{z}_t\rangle + \langle \mathbf{z}_t^\mathsf{T} C^\mathsf{T} R^{-1} C\mathbf{z}_t\rangle \right) \right\} \\
&= \arg\max_C \left\{ \mathsf{Tr}\left[C\sum_t \langle \mathbf{z}_t\rangle \mathbf{x}_t^\mathsf{T} R^{-1}\right] - \frac{1}{2}\mathsf{Tr}\left[C^\mathsf{T} R^{-1} C \left\langle \sum_t \mathbf{z}_t\mathbf{z}_t^\mathsf{T} \right\rangle \right] \right\}
\end{aligned}$$

using $\frac{\partial \mathsf{Tr}[AB]}{\partial A} = B^\mathsf{T}$, we have $\frac{\partial\{\cdot\}}{\partial C} = R^{-1}\sum_t \mathbf{x}_t\langle \mathbf{z}_t\rangle^\mathsf{T} - R^{-1}C\left\langle \sum_t \mathbf{z}_t\mathbf{z}_t^\mathsf{T}\right\rangle$

$$\Rightarrow C_{\text{new}} = \left(\sum_t \mathbf{x}_t\langle \mathbf{z}_t\rangle^\mathsf{T}\right)\left(\sum_t \left\langle \mathbf{z}_t\mathbf{z}_t^\mathsf{T}\right\rangle\right)^{-1}$$

Note the connection to linear regression.

The M step for A

$$p(\mathbf{z}_{t+1}|\mathbf{z}_t) \propto \exp\left\{-\tfrac{1}{2}(\mathbf{z}_{t+1} - A\mathbf{z}_t)^\mathsf{T} Q^{-1}(\mathbf{z}_{t+1} - A\mathbf{z}_t)\right\} \quad \Rightarrow$$

$$\begin{aligned}
A_{\text{new}} &= \arg\max_A \left\langle \sum_t \ln p(\mathbf{z}_{t+1}|\mathbf{z}_t) \right\rangle_q \\
&= \arg\max_A \left\langle -\frac{1}{2}\sum_t (\mathbf{z}_{t+1} - A\mathbf{z}_t)^\mathsf{T} Q^{-1}(\mathbf{z}_{t+1} - A\mathbf{z}_t) \right\rangle_q + \text{const} \\
&= \arg\max_A \left\{ -\frac{1}{2}\sum_t \left(\left\langle \mathbf{z}_{t+1}^\mathsf{T} Q^{-1} \mathbf{z}_{t+1} \right\rangle - 2\left\langle \mathbf{z}_{t+1}^\mathsf{T} Q^{-1} A \mathbf{z}_t \right\rangle + \left\langle \mathbf{z}_t^\mathsf{T} A^\mathsf{T} Q^{-1} A \mathbf{z}_t \right\rangle \right) \right\} \\
&= \arg\max_A \left\{ \mathsf{Tr}\left[A \sum_t \left\langle \mathbf{z}_t \mathbf{z}_{t+1}^\mathsf{T} \right\rangle Q^{-1} \right] - \frac{1}{2}\mathsf{Tr}\left[A^\mathsf{T} Q^{-1} A \sum_t \left\langle \mathbf{z}_t \mathbf{z}_t^\mathsf{T} \right\rangle \right] \right\}
\end{aligned}$$

using $\frac{\partial \mathsf{Tr}[AB]}{\partial A} = B^\mathsf{T}$, we have $\frac{\partial\{\cdot\}}{\partial A} = Q^{-1}\sum_t \left\langle \mathbf{z}_{t+1}\mathbf{z}_t^\mathsf{T} \right\rangle - Q^{-1} A \sum_t \left\langle \mathbf{z}_t \mathbf{z}_t^\mathsf{T} \right\rangle$

$$\Rightarrow A_{\text{new}} = \left(\sum_t \left\langle \mathbf{z}_{t+1}\mathbf{z}_t^\mathsf{T} \right\rangle \right) \left(\sum_t \left\langle \mathbf{z}_t \mathbf{z}_t^\mathsf{T} \right\rangle \right)^{-1}$$

This is still analagous to linear regression, with an extra expectation.

LEARNING (ONLINE GRADIENT)

Time series data must often be processed in real-time, and we may want to update parameters online as observations arrive. We can do so by updating a local version of the likelihood based on the Kalman filter estimates.
Consider the log likelihood contributed by each data point (ℓ_t):

$$\ell = \sum_{t=1}^{T} \ln p(\mathbf{x}_t | \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) = \sum_{t=1}^{T} \ell_t$$

Then,

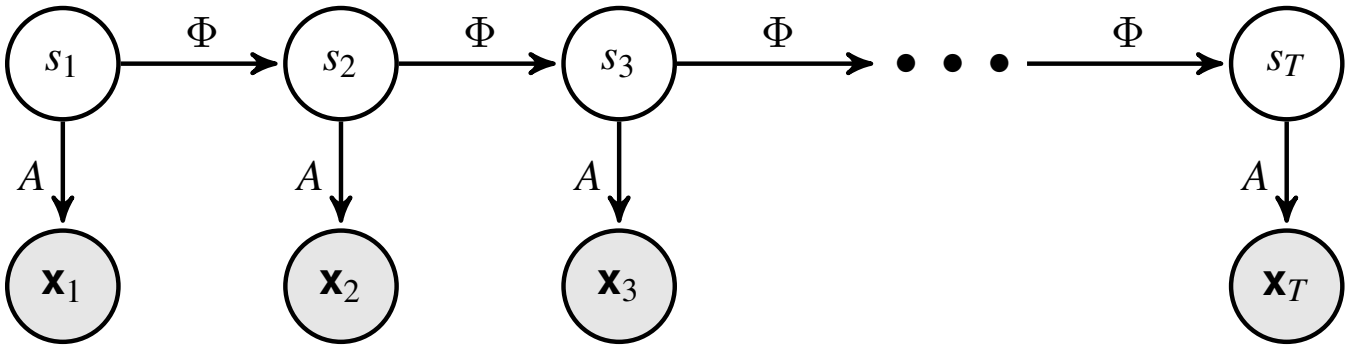
$$\ell_t = -\frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma| - \frac{1}{2} (\mathbf{x}_t - C\hat{\mathbf{z}}_t^{t-1})^{\mathsf{T}} \Sigma^{-1} (\mathbf{x}_t - C\hat{\mathbf{z}}_t^{t-1})$$

where d is dimension of $\mathbf{x}$, and:

$$\begin{aligned} \hat{\mathbf{z}}_t^{t-1} &= A\hat{\mathbf{z}}_{t-1}^{t-1} \\ \Sigma &= C\hat{V}_t^{t-1} C^{\mathsf{T}} + R \\ \hat{V}_t^{t-1} &= A\hat{V}_{t-1}^{t-1} A^{\mathsf{T}} + Q \end{aligned}$$

We differentiate ℓ_t to obtain gradient rules for A, C, Q, R. The size of the gradient step (learning rate) reflects our expectation about nonstationarity.

LEARNING HMMS USING EM



Parameters: $\theta = \{\pi, \Phi, \mathsf{A}\}$

Free energy:

$$\mathcal{F}(q, \theta) = \sum_{s_{1:T}} q(s_{1:T})(\log P(\mathbf{x}_{1:T}, s_{1:T}|\theta) - \log q(s_{1:T}))$$

E-step: Maximize $\mathcal{F}$ w.r.t. q with θ fixed: $\quad q^*(s_{1:T}) = P(s_{1:T}|\mathbf{x}_{1:T}, \theta)$

We will only need the marginal probabilities $q(s_t, s_{t+1})$, which can also be obtained from the forward–backward algorithm.

M-step: Maximize $\mathcal{F}$ w.r.t. θ with q fixed.

We can re-estimate the parameters by computing the expected number of times the HMM was in state i, emitted symbol k and transitioned to state j.

This is the **Baum-Welch algorithm** and it predates the (more general) EM algorithm.

M STEP: PARAMETER UPDATES ARE GIVEN BY RATIOS OF EXPECTED COUNTS

We can derive the following updates by taking derivatives of $\mathcal{F}$ w.r.t. θ.

- The initial state distribution is the expected number of times in state i at $t = 1$:

$$\hat{\pi}_i = \gamma_1(i)$$

- The expected number of transitions from state i to j which begin at time t is:

$$\xi_t(i \to j) \equiv P(s_t = i, s_{t+1} = j | \mathbf{x}_{1:T}) = \alpha_t(i)\Phi_{ij}A_j(\mathbf{x}_{t+1})\beta_{t+1}(j)/P(x_{1:T})$$

  so the estimated transition probabilities are:

$$\widehat{\Phi}_{ij} = \sum_{t=1}^{T-1} \xi_t(i \to j) \Big/ \sum_{t=1}^{T-1} \gamma_t(i)$$

- The output distributions are the expected number of times we observe a particular symbol in a particular state:

$$\widehat{A}_{ik} = \sum_{t:\mathbf{x}_t = k} \gamma_t(i) \Big/ \sum_{t=1}^{T} \gamma_t(i)$$

  (or the state-probability-weighted mean and variance for a Gaussian output model).

- Numerical scaling: the conventional message definition is in terms of a large joint:

  $$\alpha_t(i) = P(\mathbf{x}_{1:t}, s_t = i) \;\to 0 \text{ as } t \text{ grows, and so can easily underflow.}$$

  Rescale:

  $$\overline{\alpha}_t(i) = A_i(\mathbf{x}_t) \sum_j \tilde{\alpha}_{t-1}(j)\Phi_{ji} \qquad \rho_t = \sum_{i=1}^{K} \overline{\alpha}_t(i) \qquad \tilde{\alpha}_t(i) = \overline{\alpha}_t(i)/\rho_t$$

  Exercise: show that:

  $$\rho_t = P(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \theta) \qquad \prod_{t=1}^{T} \rho_t = P(\mathbf{x}_{1:T}|\theta)$$

  What does this make $\tilde{\alpha}_t(i)$?

- Multiple observed sequences: average numerators and denominators in the ratios of updates.

HMM PSEUDOCODE: E STEP

Forward-backward including scaling tricks.
[$\odot$ is the element-by-element (Hadamard/Schur) product: '$*$' in NumPy, '$.*$' in matlab.]

$$
\begin{aligned}
&\text{for } t = 1:T,\ i = 1:K && p_t(i) = A_i(\mathbf{x}_t) \\
& && \alpha_1 = \pi \odot p_1 && \rho_1 = \textstyle\sum_{i=1}^{K} \alpha_1(i) && \alpha_1 = \alpha_1/\rho_1 \\
&\text{for } t = 2:T && \alpha_t = (\Phi^{\mathsf{T}} * \alpha_{t-1}) \odot p_t && \rho_t = \textstyle\sum_{i=1}^{K} \alpha_t(i) && \alpha_t = \alpha_t/\rho_t \\
& && \beta_T = 1 \\
&\text{for } t = T-1:1 && \beta_t = \Phi * (\beta_{t+1} \odot p_{t+1})/\rho_{t+1} \\
& && \log P(\mathbf{x}_{1:T}) = \textstyle\sum_{t=1}^{T} \log(\rho_t) \\
&\text{for } t = 1:T && \gamma_t = \alpha_t \odot \beta_t \\
&\text{for } t = 1:T-1 && \xi_t = \Phi \odot (\alpha_t * (\beta_{t+1} \odot p_{t+1})^{\mathsf{T}})/\rho_{t+1}
\end{aligned}
$$

HMM PSEUDOCODE: M STEP

Baum-Welch parameter updates:

For each sequence $l = 1 : L$, run forward–backward to get $\gamma^{(l)}$ and $\xi^{(l)}$, then

$$\pi_i = \frac{1}{L} \sum_{l=1}^{L} \gamma_1^{(l)}(i)$$

$$\Phi_{ij} = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}-1} \xi_t^{(l)}(ij)}{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}-1} \gamma_t^{(l)}(i)}$$

$$A_{ik} = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}} \delta(\mathbf{x}_t^{(l)} = k) \gamma_t^{(l)}(i)}{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}} \gamma_t^{(l)}(i)}$$

Problem

Given speech in form of a sound signal, determine the words that have been spoken.

Method

- Words are broken down into small sound units (called *phonemes*). The states in the HMM represent phonemes, or parts of phonemes (called subphonemes).
- The incoming sound signal is transformed into a sequence of vectors (feature extraction). Each vector $\mathbf{x}_t$ is indexed by a time step t.
- The sequence $\mathbf{x}_{1:T}$ of feature vectors is the observed data in the HMM.

Phoneme

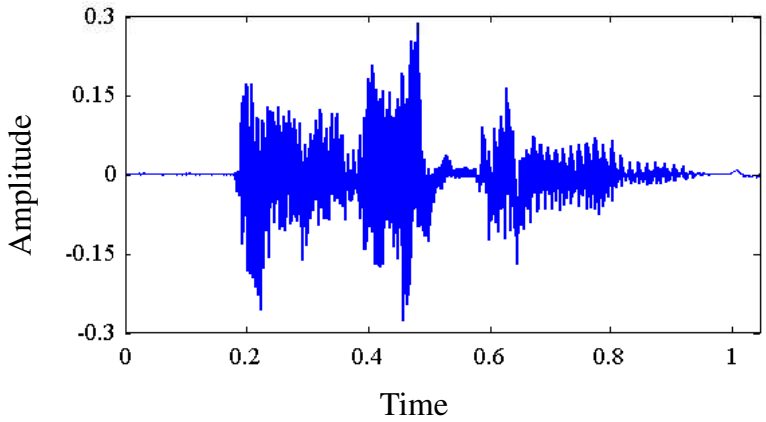
A **phoneme** is defined as the smallest unit of sound in a language that distinguishes between distinct meanings. English uses about 50 phonemes.

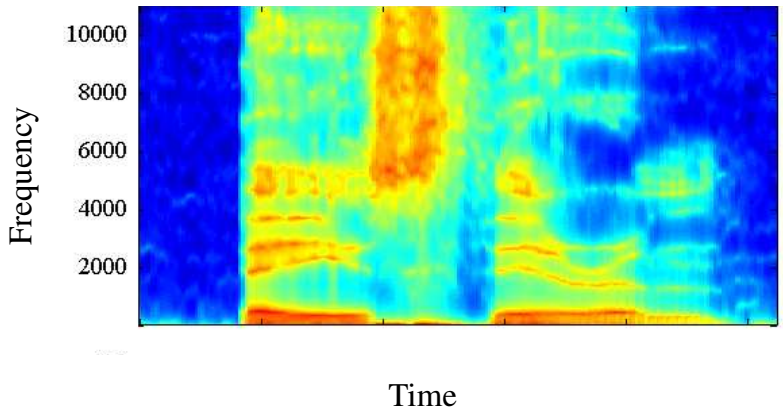
Example

Zero	Z IH R OW	Six	S IH K S
One	W AH N	Seven	S EH V AX N
Two	T UW	Eight	EY T
Three	TH R IY	Nine	N AY N
Four	F OW R	Oh	OW
Five	F AY V		

Subphonemes

Phonemes can be further broken down into subphonemes. The standard in speech processing is to represent a phoneme by three subphonemes.

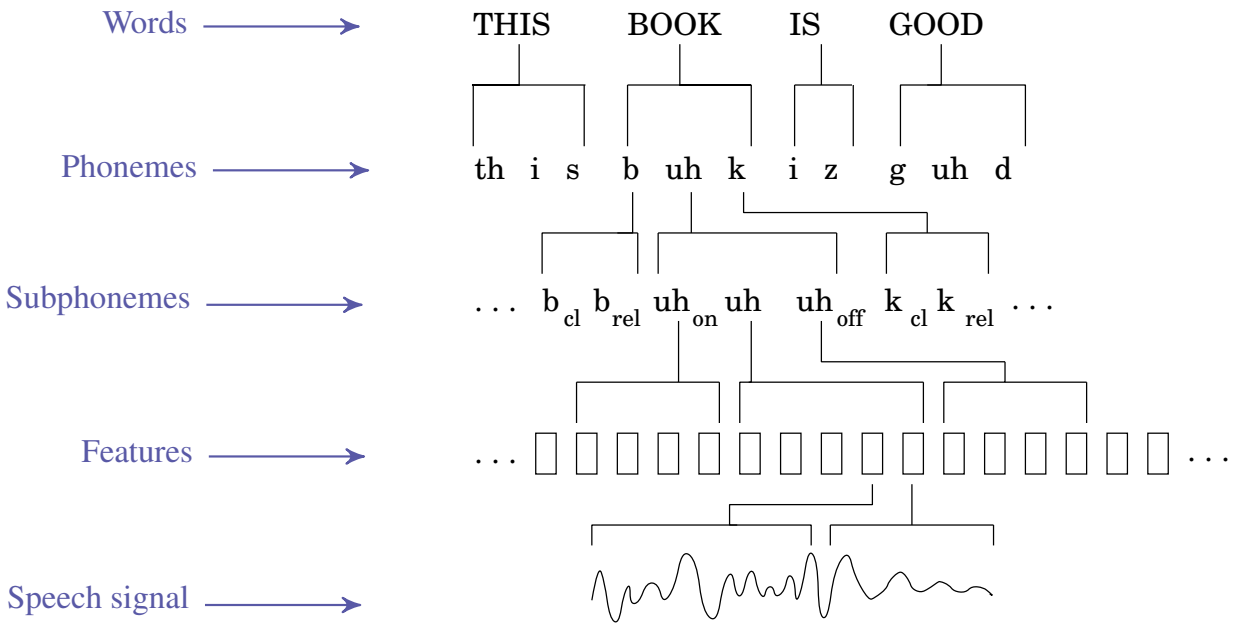




Feature extraction

- A speech signal is measured as amplitude over time.
- The signal is typically transformed into various types of features, including (windowed) Fourier- or cosine-transforms and so-called "cepstral features".
- Each of these transforms is a scalar function of time. All function values for the different transforms at time *t* are collected in a vector, which is the feature vector (at time *t*).

Layers in Phoneme Models



HMM speech recognition

- Training: The HMM parameters (emission parameters and transition probabilities) are estimated from data, often using both supervised and unsupervised techniques.
- Recognition: Given a language signal (= observation sequence $\mathbf{x}_{1:N}$, estimate the corresponding sequence of subphonemes (= states $z_{1:N}$). This is a decoding problem.

Illustration: K. Murphy, "Machine Learning".