

# Statistical Machine Learning (BE4M33SSU)

## Hidden Markov Models

Czech Technical University in Prague: Boris Flach and Jan Drchal

- ◆ Markov models on sequences: inference & parameter learning
- ◆ Hidden Markov Models (HMMs)

## Structured Hidden States

Models discussed so far: mainly classifiers predicting a categorical (class) variable  $y \in \mathcal{Y}$

Often in applications: the hidden state  $y$  is a structured variable.

Here: the hidden state  $y$  is a **sequence** of categorical variables.

### Application examples:

- ◆ language modelling (next token prediction),
- ◆ text recognition (printed, handwritten),
- ◆ token classification (named entity recognition, POS tagging),
- ◆ speech recognition (single word recognition, continuous speech recognition, translation),
- ◆ text-to-speech,
- ◆ bioinformatics (gene prediction),
- ◆ finance (predict stock price),
- ◆ cybersecurity (anomaly detection),
- ◆ robot self-localization.

### Markov Models and Hidden Markov Models on chains:

a class of generative probabilistic models for sequences of features and sequences of categorical variables.

## Markov Models: Preliminaries

Let  $\mathbf{s} = (s_1, s_2, \dots, s_n)$  denote a sequence of length  $n$  with elements from a finite set  $K$ .

Each element corresponds to a *discrete time step*.

Any joint probability distribution on  $K^n$  can be written as

$$p(s_1, s_2, \dots, s_n) = p(s_1) p(s_2 | s_1) p(s_3 | s_2, s_1) \cdot \dots \cdot p(s_n | s_1, \dots, s_{n-1})$$

To see that, use the definition of conditional probability recursively:

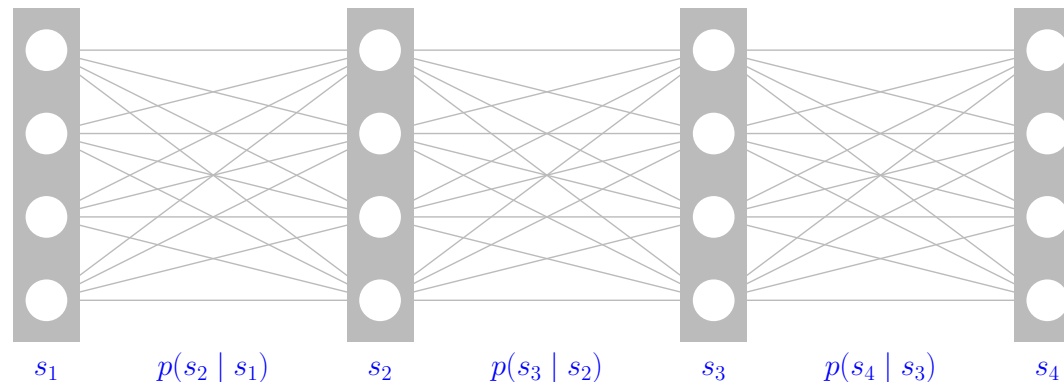
$$\begin{aligned} p(s_1, s_2, \dots, s_n) &= p(s_n | s_1, \dots, s_{n-1}) p(s_1, s_2, \dots, s_{n-1}) = \\ &= p(s_n | s_1, \dots, s_{n-1}) p(s_{n-1} | s_1, s_2, \dots, s_{n-2}) p(s_1, s_2, \dots, s_{n-2}) = \dots \end{aligned}$$

# Markov Models: Definition

**Definition 1.** A joint p.d. on  $K^n$  is a *first-order* Markov model if

$$p(\mathbf{s}) = p(s_1) p(s_2 | s_1) p(s_3 | s_2) \cdot \dots \cdot p(s_n | s_{n-1}) = p(s_1) \prod_{t=2}^n p(s_t | s_{t-1})$$

holds for any  $\mathbf{s} = (s_1, s_2, \dots, s_n)$ .



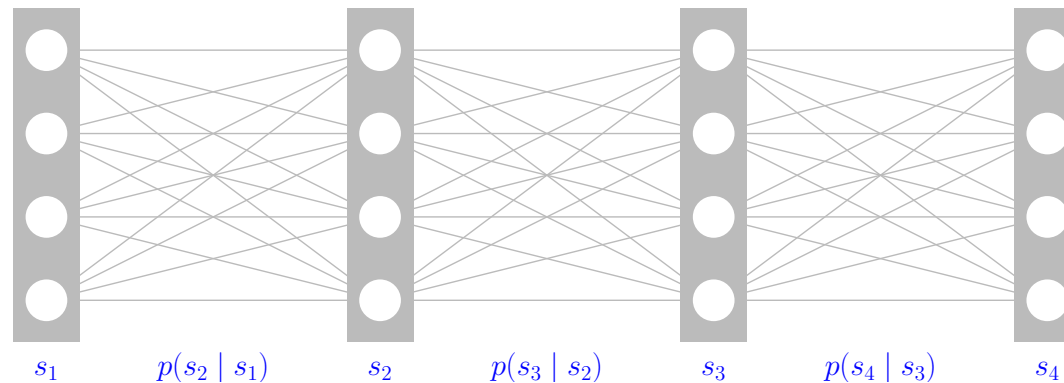
**Remark 1.** Higher degree MMs consider a longer history, e.g., a *second-order* MM uses  $p(s_t | s_{t-1}, s_{t-2})$  transition probabilities.

# Markov Models: Definition

**Definition 1.** A joint p.d. on  $K^n$  is a *first-order* Markov model if

$$p(\mathbf{s}) = p(s_1) p(s_2 | s_1) p(s_3 | s_2) \cdot \dots \cdot p(s_n | s_{n-1}) = p(s_1) \prod_{t=2}^n p(s_t | s_{t-1})$$

holds for any  $\mathbf{s} = (s_1, s_2, \dots, s_n)$ .



**Remark 1.** Higher degree MMs consider a longer history, e.g., a *second-order* MM uses  $p(s_t | s_{t-1}, s_{t-2})$  transition probabilities.

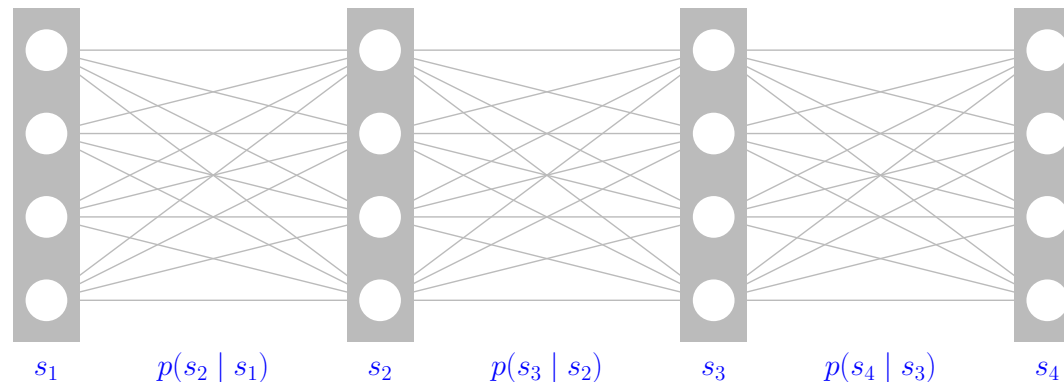
**Question 1:** How to generate a sequence?

# Markov Models: Definition

**Definition 1.** A joint p.d. on  $K^n$  is a *first-order* Markov model if

$$p(\mathbf{s}) = p(s_1) p(s_2 | s_1) p(s_3 | s_2) \cdot \dots \cdot p(s_n | s_{n-1}) = p(s_1) \prod_{t=2}^n p(s_t | s_{t-1})$$

holds for any  $\mathbf{s} = (s_1, s_2, \dots, s_n)$ .



**Remark 1.** Higher degree MMs consider a longer history, e.g., a *second-order* MM uses  $p(s_t | s_{t-1}, s_{t-2})$  transition probabilities.

**Question 1:** How to generate a sequence?

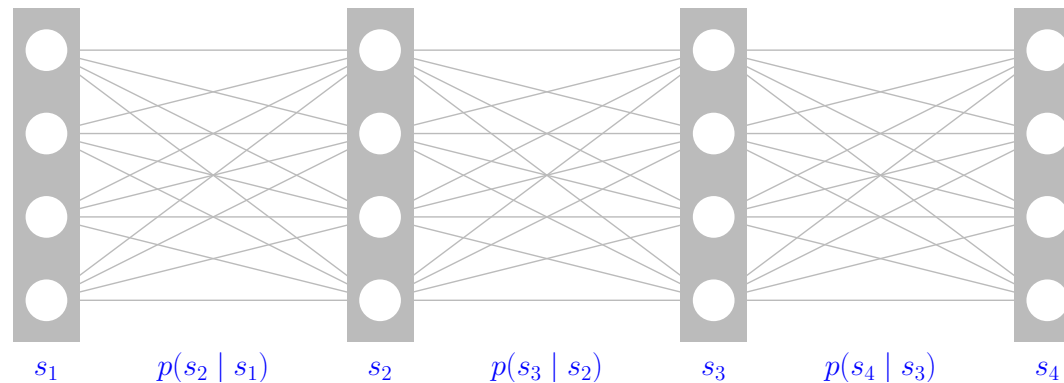
**Question 2:** What is the number of sequences for English alphabet?

# Markov Models: Definition

**Definition 1.** A joint p.d. on  $K^n$  is a *first-order* Markov model if

$$p(\mathbf{s}) = p(s_1) p(s_2 | s_1) p(s_3 | s_2) \cdot \dots \cdot p(s_n | s_{n-1}) = p(s_1) \prod_{t=2}^n p(s_t | s_{t-1})$$

holds for any  $\mathbf{s} = (s_1, s_2, \dots, s_n)$ .



**Remark 1.** Higher degree MMs consider a longer history, e.g., a *second-order* MM uses  $p(s_t | s_{t-1}, s_{t-2})$  transition probabilities.

**Question 1:** How to generate a sequence?

**Question 2:** What is the number of sequences for English alphabet?

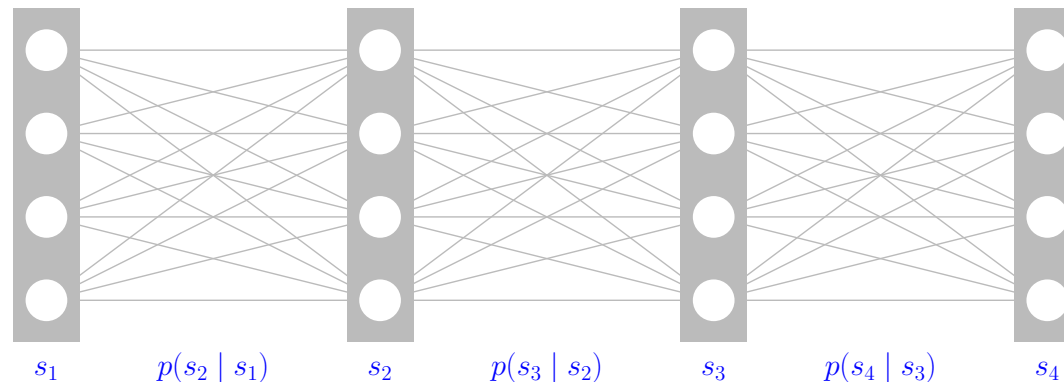
$26^n$  for 26 lowercase letters.

# Markov Models: Definition

**Definition 1.** A joint p.d. on  $K^n$  is a *first-order* Markov model if

$$p(\mathbf{s}) = p(s_1) p(s_2 | s_1) p(s_3 | s_2) \cdot \dots \cdot p(s_n | s_{n-1}) = p(s_1) \prod_{t=2}^n p(s_t | s_{t-1})$$

holds for any  $\mathbf{s} = (s_1, s_2, \dots, s_n)$ .



**Remark 1.** Higher degree MMs consider a longer history, e.g., a *second-order* MM uses  $p(s_t | s_{t-1}, s_{t-2})$  *transition probabilities*.

**Question 1:** How to generate a sequence?

**Question 2:** What is the number of sequences for English alphabet?

$26^n$  for 26 lowercase letters.

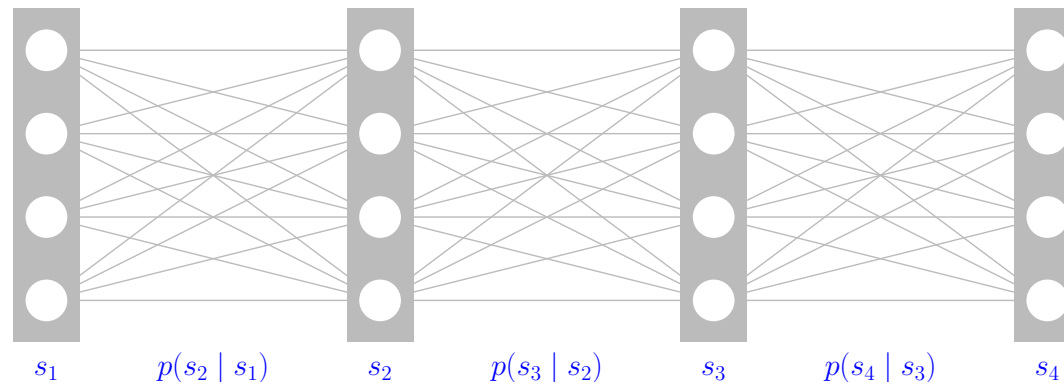
**Question 3:** How many values do we need to fix the *transition probabilities*  $p(s_t | s_{t-1})$ ?

## Markov Models: Definition

**Definition 1.** A joint p.d. on  $K^n$  is a *first-order* Markov model if

$$p(\mathbf{s}) = p(s_1) p(s_2 | s_1) p(s_3 | s_2) \cdot \dots \cdot p(s_n | s_{n-1}) = p(s_1) \prod_{t=2}^n p(s_t | s_{t-1})$$

holds for any  $\mathbf{s} = (s_1, s_2, \dots, s_n)$ .



**Remark 1.** Higher degree MMs consider a longer history, e.g., a *second-order* MM uses  $p(s_t | s_{t-1}, s_{t-2})$  *transition probabilities*.

**Question 1:** How to generate a sequence?

**Question 2:** What is the number of sequences for English alphabet?

$26^n$  for 26 lowercase letters.

**Question 3:** How many values do we need to fix the *transition probabilities*  $p(s_t | s_{t-1})$ ?  
 $K \times K$  (*stochastic matrix*: columns sum to 1)  $\Rightarrow K \times (K - 1)$  parameters is enough.

## Markov Models: Matrix Representation

The elements of a **state distribution** vector  $\boldsymbol{\pi}_t$  for time step  $t$  are defined as:

$$(\boldsymbol{\pi}_t)_j = p(s_t = j).$$

Similarly, a **transition matrix**  $P$  is defined as:

$$P_{ij}(t) = p(s_t = i \mid s_{t-1} = j),$$

where  $\sum_i P_{ij}(t) = 1$ , i.e., the  $P$  is a *stochastic matrix*.

A Markov model is called **homogeneous** if the transition probabilities  $P$  do not depend on time. In this case, the *state distribution* at time step  $t$  can be easily written as:

$$\boldsymbol{\pi}_t = P^{t-1} \boldsymbol{\pi}_1,$$

where  $\boldsymbol{\pi}_1$  is the *initial state (starting) distribution*.

## Markov Models: Random Walker Example

**Example 1** (Random walk on a graph).

- ◆ Let  $(V, E)$  be a directed graph. A random walk in  $(V, E)$  is described by a sequence  $s = (s_1, \dots, s_t, \dots)$  of visited nodes, i.e.,  $s_t \in V$ .
- ◆ The walker starts in node  $i \in V$  with probability  $p(s_1 = i)$ .
- ◆ The edges of the graph are weighted by  $w : E \rightarrow \mathbb{R}_+$ , s.t.

$$\sum_{j: (i,j) \in E} w_{ij} = 1 \quad \forall i \in V$$

- ◆ In the current position  $s_t = i$ , the walker randomly chooses an outgoing edge with probability given by the weights and moves along this edge, i.e.

$$p(s_{t+1} = j \mid s_t = i) = \begin{cases} w_{ij} & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

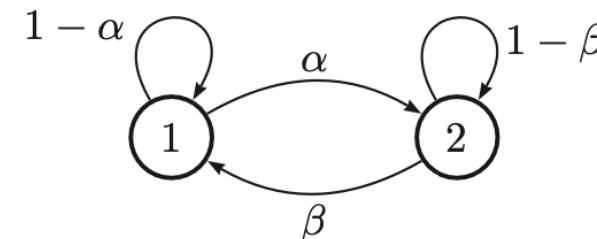
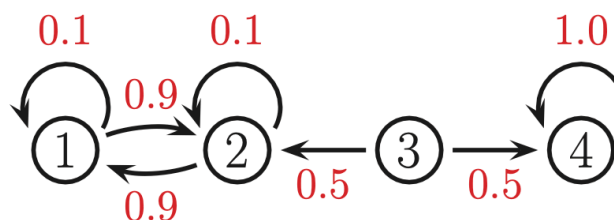
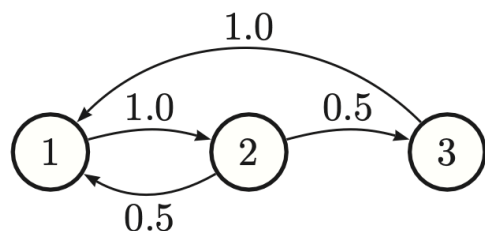
**Questions:** How does the distribution  $p(s_t)$  behave? Does it converge to some fix-point distribution  $\pi$  for  $t \rightarrow \infty$ ?

## Return Times and Limiting Distributions

- ◆ A homogeneous Markov model is **irreducible** if each state  $l$  can be reached starting from any state  $k$  with non-zero probability (after some number of transitions).
- ◆ A state  $k$  has **return time**  $\tau$  if it can be reached with non-zero probability after  $\tau$  transitions when starting from itself.
- ◆ A state  $k \in K$  is **a-periodic** if the greatest common divisor of its *return times* is 1.

### Example 2. Markov chains:

- ◆ **left:** *irreducible, a-periodic* chain:  $d(1) = d(2) = \gcd(2, 3, 4, 5, 6, \dots) = 1$ ,  
 $d(3) = \gcd(3, 5, 6, \dots) = 1$ .
- ◆ **middle:** *reducible* chain: starting in 4 (*absorbing state*) we end up with  $\pi = (0, 0, 0, 1)$ ; starting in 1 or 2 we get  $\pi = (\frac{1}{2}, \frac{1}{2}, 0, 0)$ ; starting in 3 leads to either of the two *stationary distributions*.
- ◆ **right:** *irreducible* chain for  $\alpha, \beta = 0.9$ : we end up with  $\pi = (\frac{1}{2}, \frac{1}{2})$  by symmetry. For  $\alpha, \beta = 1$  we get *periodic oscillations*.



## Return Times and Limiting Distributions, contd.

**Theorem 1.** Let  $P$  be the transition probability matrix of an **irreducible** homogeneous Markov model with all states **a-periodic**. Then there exists a unique **stationary distribution**  $\pi^*$  s.t.  $P\pi^* = \pi^*$ . Moreover, it is a **limiting distribution**, i.e.

$$\lim_{t \rightarrow \infty} P^t \pi_1 = \pi^*$$

for arbitrary starting distributions  $\pi_1$ .

**Remark 2.** The two assumptions for the limiting distribution:

- ◆ the **irreducibility** means that the state transition diagram must be a singly-connected component,
- ◆ the **a-periodicity** means no oscillations.

**Remark 3.** A special case of the above is a **regular** chain for which:  $P_{ij}^n > 0$  for some  $n$  and all  $i$  and  $j$ :

- ◆ Hence, after  $n$  steps, we can end up in any state, no matter where we started.
- ◆ *Irreducibility* and a *self-loop* for all states is all we need.

# Google PageRank

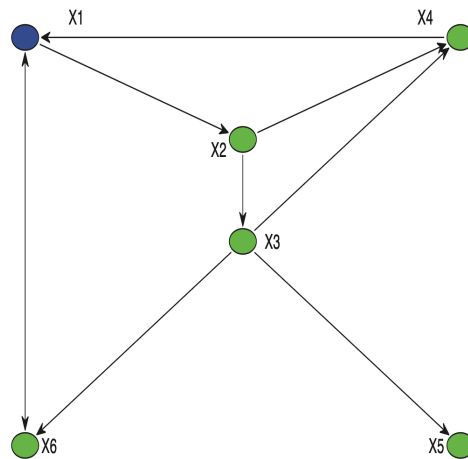
- ◆ One of the methods used to rank the retrieved documents (big in 1997).
- ◆ **Idea**: good pages are referenced more often; a random surfer spends more time on highly reachable pages.
- ◆ It is the *Random walk*:
  - $P$  – transition matrix based on hyperlinks between pages,
  - $\pi_i$  – page  $i$  authority (high for being linked by many other pages),
  - authority is passed to the linked pages:

$$\pi_i = \sum_j P_{ij} \pi_j.$$

- Corresponds to the Markov model *stationary distribution*.
- ◆ We need the  $P$  to be *irreducible* having all states *a-periodic*:
  - Real link adjacency matrix  $G$  is sparse as not all pages are linked to all other pages.
  - We must set  $P$ , so even non-existing links (in  $G$ ) get at least a very small probability, so we end up with a **regular** chain.

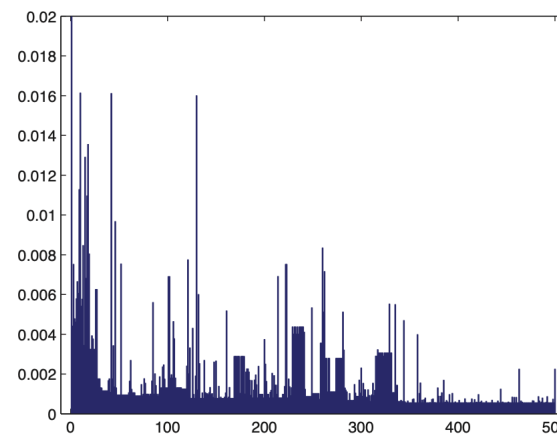
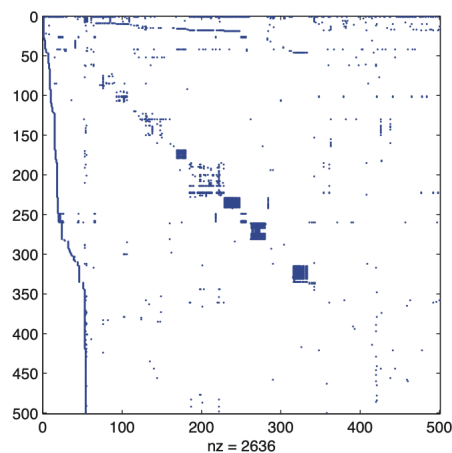
# Google PageRank Example

Simple example



Converges to  $\pi = (0.3209, 0.1706, 0.1065, 0.1368, 0.0643, 0.2008)$

Example for 500 pages linked from `www.harvard.edu`:  
Adjacency matrix  $G$  (left), PageRank vector  $\pi$  (right).



# Algorithms: Computing the Most Probable Sequence

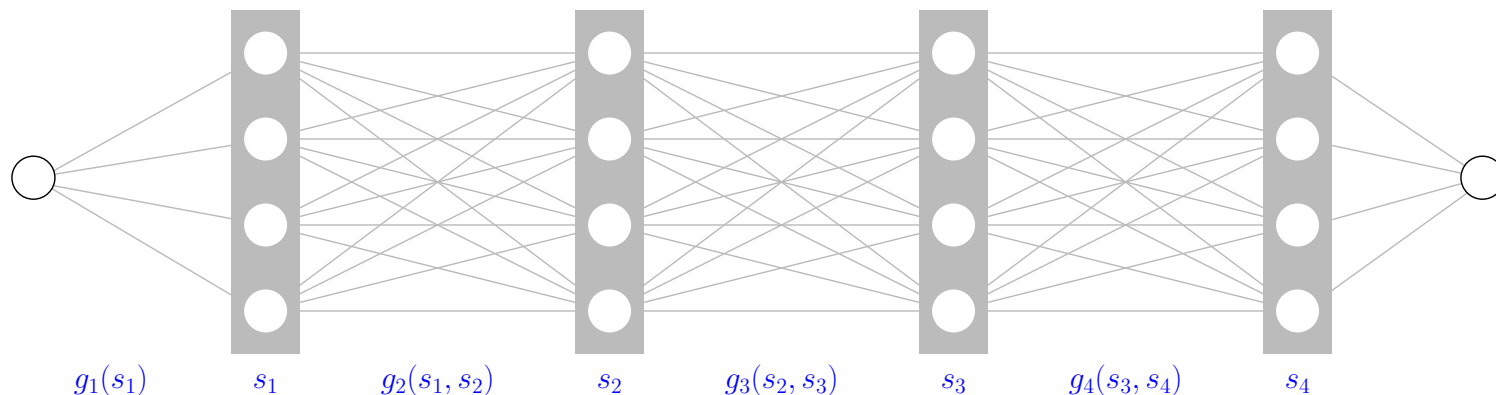
We want to compute the most probable sequence  $\mathbf{s}^* \in \arg \max_{\mathbf{s} \in K^n} \left[ p(s_1) \prod_{t=2}^n p(s_t | s_{t-1}) \right]$ .

Take the logarithm of  $p(\mathbf{s})$ :  $\mathbf{s}^* \in \arg \max_{\mathbf{s} \in K^n} \left[ g_1(s_1) + \sum_{t=2}^n g_t(s_{t-1}, s_t) \right]$

and apply dynamic programming: Set  $\phi_1(s_1) \equiv g_1(s_1)$  and compute

$$\phi_t(s_t) = \max_{s_{t-1} \in K} \left[ \phi_{t-1}(s_{t-1}) + g_t(s_{t-1}, s_t) \right] \quad \forall s_t \in K.$$

Finally, find  $s_n^* \in \arg \max_{s_n \in K} \phi_n(s_n)$  and back-track the solution. This corresponds to searching the best path in the graph



**Question:** What is the run-time complexity? .

# Algorithms: Computing the Most Probable Sequence

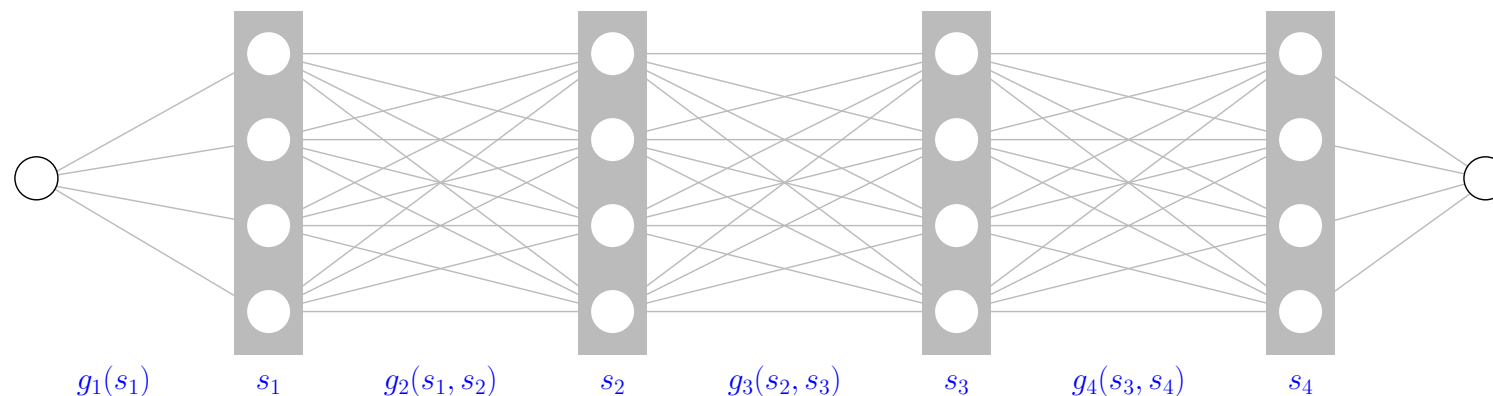
We want to compute the most probable sequence  $\mathbf{s}^* \in \arg \max_{\mathbf{s} \in K^n} \left[ p(s_1) \prod_{t=2}^n p(s_t | s_{t-1}) \right]$ .

Take the logarithm of  $p(\mathbf{s})$ :  $\mathbf{s}^* \in \arg \max_{\mathbf{s} \in K^n} \left[ g_1(s_1) + \sum_{t=2}^n g_t(s_{t-1}, s_t) \right]$

and apply dynamic programming: Set  $\phi_1(s_1) \equiv g_1(s_1)$  and compute

$$\phi_t(s_t) = \max_{s_{t-1} \in K} \left[ \phi_{t-1}(s_{t-1}) + g_t(s_{t-1}, s_t) \right] \quad \forall s_t \in K.$$

Finally, find  $s_n^* \in \arg \max_{s_n \in K} \phi_n(s_n)$  and back-track the solution. This corresponds to searching the best path in the graph

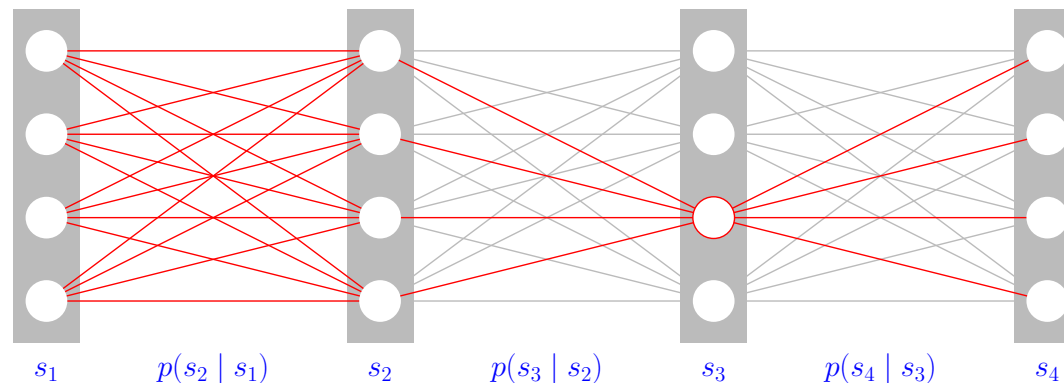


**Question:** What is the run-time complexity?  $\mathcal{O}(nK^2)$ .

# Algorithms: Computing Marginal Probabilities

How to compute marginal probabilities for the sequence element  $s_j$  in position  $j$

$$p(s_j) = \sum_{s_1 \in K} \cdots \cancel{\sum_{s_j \in K}} \cdots \sum_{s_n \in K} p(s_1) \prod_{t=2}^n p(s_t | s_{t-1})$$



Summation over the **trailing** variables is easily done because:

$$\sum_{s_n \in K} p(s_1) \cdots p(s_{n-1} | s_{n-2}) p(s_n | s_{n-1}) = p(s_1) \cdots p(s_{n-1} | s_{n-2})$$

## Algorithms: Computing Marginal Probabilities, contd.

The summation over the **leading** variables is done dynamically: Begin with  $p(s_1)$  and compute:

$$p(s_t) = \sum_{s_{t-1} \in K} p(s_t | s_{t-1}) p(s_{t-1}) \quad \forall s_t \in K.$$

Equivalently using matrix representation the computation reads:  $\pi_t = P(t)\pi_{t-1}$ .

Run-time complexity once again  $\mathcal{O}(nK^2)$ .

### Remark 4.

- ◆ Notice that the preferred direction (from first to last) in the definition of a Markov model is only apparent. By computing the marginal probabilities  $p(s_t)$  and by using  $p(s_t | s_{t-1})p(s_{t-1}) = p(s_{t-1}, s_t) = p(s_{t-1} | s_t)p(s_t)$ , we can rewrite the model in reverse order.

## Algorithms: Learning a Markov Model

Suppose we are given i.i.d. training data  $\mathcal{T}^m = \{\mathbf{s}^k \in K^n \mid k = 1, \dots, m\}$  and want to estimate the parameters of the Markov model by the maximum likelihood estimate. This is very easy:

- ◆ Denote by  $a(s_{t-1} = j, s_t = i)$  the number of sequences in  $\mathcal{T}^m$  for which  $s_{t-1} = j$  and  $s_t = i$ .
- ◆ The estimates for the conditional probabilities are then given by

$$p(s_t = i \mid s_{t-1} = j) = \frac{a(s_{t-1} = j, s_t = i)}{\sum_k a(s_{t-1} = j, s_t = k)}.$$

## Algorithms: Learning a Markov Model, contd.

But is our estimate an MLE?

Let's recall the log-likelihood:

$$\frac{1}{m} \sum_{\mathbf{s} \in \mathcal{T}^m} \left[ \log p(s_1) + \sum_{t=2}^n \log p(s_t | s_{t-1}) \right]$$

**Proof** (idea):

Consider all terms in the log-likelihood that depend on the transition probability from  $(t-1) \rightarrow t$  and rewrite them using transition counts  $a(s_{t-1} = j, s_t = i)$

$$\frac{1}{m} \sum_{\mathbf{s} \in \mathcal{T}^m} \log p(s_t | s_{t-1}) = \frac{1}{m} \sum_{i,j \in K} a(s_{t-1} = j, s_t = i) \log p(s_t = i | s_{t-1} = j)$$

Maximise this w.r.t.  $p(s_t | s_{t-1})$  under the constraint  $\sum_{s_t \in K} p(s_t | s_{t-1}) = 1$  using Lagrange multipliers.

## Markov Models are Exponential Families

Markov models are **exponential families**. For simplicity we show this for the family of homogeneous Markov models on sequences  $\mathbf{s} = (s_1, s_2, \dots, s_n)$  of length  $n$  under the additional assumption that  $p(s_1) = \frac{1}{K}$ .

We have

$$p(\mathbf{s}) = \frac{1}{K} \prod_{t=2}^n p(s_t | s_{t-1}),$$

- ◆ sufficient statistic:  $\Phi(\mathbf{s})$  is a  $K \times K$  matrix with entries  $\Phi_{kl}(\mathbf{s})$  counting the number of transitions from state  $l$  to state  $k$  in the sequence  $\mathbf{s}$ ,
- ◆ natural parameter:  $H$  is a  $K \times K$  matrix with entries  $H_{kl} = \log p(s_t = k | s_{t-1} = l)$ ,
- ◆ base measure:  $h(\mathbf{s}) = 1$ ,
- ◆ cumulant function:  $\log(K)$ .

We can write the probability of sequences as

$$p(\mathbf{s}; H) = \exp[\langle \Phi(\mathbf{s}), H \rangle - \log(K)]$$

**Remark 5.** This can be generalised for models with non-uniform  $p(s_1)$  and also for general (i.e. non-homogeneous) Markov models.

## Hidden Markov Models

- ◆ Let  $\mathbf{s} = (s_1, s_2, \dots, s_n)$  denote a sequence of hidden states from a finite set  $K$ .
- ◆ Let  $\mathbf{x} = (x_1, x_2, \dots, x_n)$  denote a sequence of features from some feature space  $\mathcal{X}$ .

**Definition 2.** A joint p.d. on  $\mathcal{X}^n \times K^n$  is a Hidden Markov model if

- (a) the prior p.d.  $p(\mathbf{s})$  for the sequences of hidden states is a Markov model, and
- (b) the conditional distribution  $p(\mathbf{x} | \mathbf{s})$  for the feature sequence is independent, i.e.

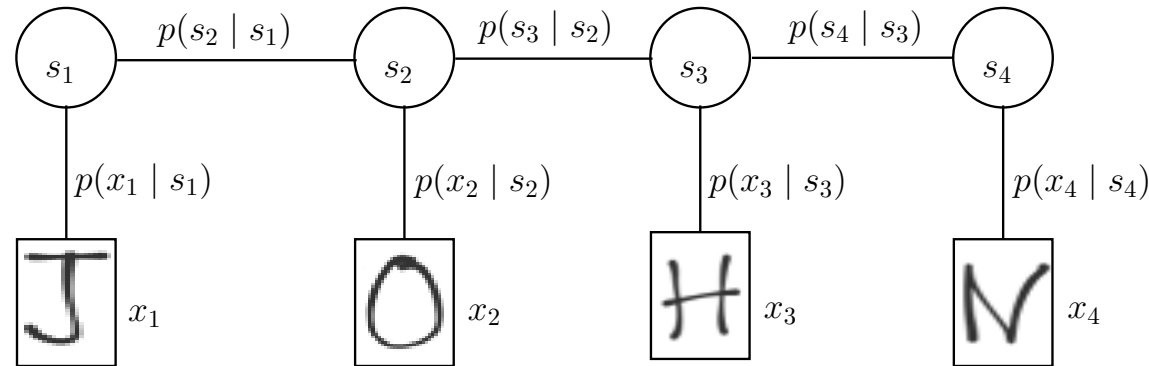
$$p(\mathbf{x} | \mathbf{s}) = \prod_{t=1}^n p(x_t | s_t).$$

The joint model is given by

$$p(\mathbf{x}, \mathbf{s}) = p(s_1) \prod_{t=2}^n p(s_t | s_{t-1}) \prod_{t=1}^n p(x_t | s_t).$$

## Hidden Markov Model: OCR Example

**Example 3** (Text recognition, OCR).

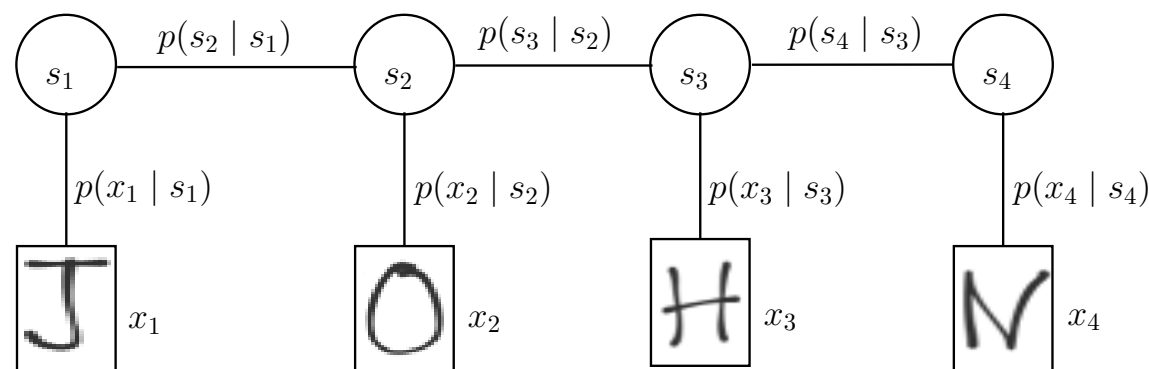


- ◆  $\mathbf{x} = (x_1, x_2, \dots, x_n)$  – sequence of images with characters,
- ◆  $\mathbf{s} = (s_1, s_2, \dots, s_n)$  – sequence of alphabetic characters,
- ◆  $p(s_t | s_{t-1})$  – language model (*transition probabilities*),
- ◆  $p(x_t | s_t)$  – appearance model for characters (*emission probabilities*).

**Question:** How to generate a sequence?

## Hidden Markov: Inference Tasks

- ◆ **Filtering**: compute the **belief state**  $p(s_t | \mathbf{x}_{1:t})$  *online* using **forward** algorithm. Get less noise than using just  $p(s_t | x_t)$ .
- ◆ **Smoothing**: compute  $p(s_t | \mathbf{x}_{1:n})$  *offline* given all evidence using **forward-backward** algorithm. Even less uncertainty given future data.
- ◆ **Fixed lag smoothing**:  $p(s_{t-l} | \mathbf{x}_{1:t})$ , where  $l > 0$  is called the lag, compromise of online and offline.
- ◆ **Most probable sequence**: compute  $\arg \max_{\mathbf{s}_{1:n}} p(\mathbf{s}_{1:n} | \mathbf{x}_{1:n})$  using **Viterbi** algorithm.
- ◆ **Prediction**: compute  $p(s_{t+h} | \mathbf{x}_{1:t})$ , where  $h > 0$  is called the prediction **horizon**: just use the *belief state* and power of the transition matrix  $P^h$  (for a homogeneous HMM).
- ◆ **Posterior samples**: generate new samples  $\mathbf{s}_{1:n}^s \sim p(\mathbf{s}_{1:n} | \mathbf{x}_{1:n})$ .



## Filtering: Forward Algorithm

We want to compute  $\alpha_t(k) \triangleq p(s_t = k | \mathbf{x}_{1:t})$ ,  $\forall k \in K$  in some sequence position  $t$ .

Let us start with one-step-ahead predictive density:

$$p(s_t = k | \mathbf{x}_{1:t-1}) = \sum_j \underbrace{p(s_t = k | s_{t-1} = j)}_{\Psi(j,k)} \underbrace{p(s_{t-1} = j | \mathbf{x}_{1:t-1})}_{\alpha_{t-1}(j)},$$

now define the update step:

$$\alpha_t(k) = p(s_t = k | \mathbf{x}_{1:t}) = p(s_t = k | x_t, \mathbf{x}_{1:t-1}) = \frac{1}{Z_t} p(x_t | s_t = k, \mathbf{x}_{1:t-1}) p(s_t = k | \mathbf{x}_{1:t-1}),$$

where  $Z_t = p(x_t | \mathbf{x}_{1:t-1}) = \sum_{k'} p(x_t | s_t = k') p(s_t = k' | \mathbf{x}_{1:t-1})$ .

The matrix-vector form of the update is:

$$\alpha_t \propto \psi_t \odot (\Psi^T \alpha_{t-1}),$$

where  $\psi_t(k) \triangleq p(x_t | s_t = k)$  denotes the emission probabilities and  $\odot$  is the Hadamard product (elementwise vector multiplication). Initialize with  $\alpha_1 \propto \psi_1 \odot \pi_1$ .

This is the **forward** algorithm.

## Smoothing: Forward-Backward Algorithm

How to compute marginal probabilities for hidden states given the sequence of features?

$$p(s_t = k | \mathbf{x}) = p(s_t = k | \mathbf{x}_{1:t}, \mathbf{x}_{t+1:n}) \propto \underbrace{p(s_t = k | \mathbf{x}_{1:t})}_{\text{forward}} \underbrace{p(\mathbf{x}_{t+1:n} | s_t = k, \mathbf{x}_{1:t})}_{\text{backward}}$$

We know how to compute the *forward*, now let us look at the *backward* part, firstly define the conditional likelihood of future evidence given that the hidden state at time  $i$ :

$$\beta_t(k) \triangleq p(\mathbf{x}_{t+1:n} | s_t = k).$$

Let us establish the recurrence:

$$\begin{aligned} \beta_{t-1}(k) &= p(\mathbf{x}_{t:n} | s_{t-1} = k) = \sum_j p(s_t = j, \mathbf{x}_t, \mathbf{x}_{t+1:n} | s_{t-1} = k) = \\ &= \sum_j p(\mathbf{x}_{t+1:n} | s_t = j, \cancel{s_{t-1} = k}, \mathbf{x}_t) p(s_t = j, \mathbf{x}_t | s_{t-1} = k) = \\ &= \sum_j p(\mathbf{x}_{t+1:n} | s_t = j) p(\mathbf{x}_t | s_t = j, \cancel{s_{t-1} = k}) p(s_t = j | s_{t-1} = k) = \sum_j \beta_t(j) \psi_t(j) \Psi(k, j). \end{aligned}$$

The **backward** algorithm is then  $\beta_{t-1} \propto \Psi(\psi_t \odot \beta_t)$ , with  $\beta_n(k) = p(\underbrace{\mathbf{x}_{n+1:n}}_{\emptyset} | s_n = k) = 1$ .

## Smoothing: Forward-Backward Algorithm, contd.

The **forward-backward** algorithm now reads:

$$p(s_t = k | \mathbf{x}) \propto \alpha_t(k) \beta_t(k),$$

where both  $\alpha_t(k)$  and  $\beta_t(k)$  are computed using dynamic programming. The result must be normalized.

The computational complexity for computing  $\alpha_t(k)$  and  $\beta_t(k)$  in **all** positions  $t = 1, \dots, n$  is thus  $\mathcal{O}(nK^2)$ .

## Two-Slice Smoothed Marginals

We can also compute **pairwise** marginal probabilities  $p(s_t, s_{t+1} | \mathbf{x})$  from the  $\alpha$ -s and  $\beta$ -s. Start with:

$$\begin{aligned}
 p(s_t, s_{t+1}, \mathbf{x}) &= p(\mathbf{x}_{1:t} | s_t, s_{t+1}) p(x_{t+1} | s_t, s_{t+1}) p(\mathbf{x}_{t+2:n} | s_t, s_{t+1}) p(s_{t+1} | s_t) p(s_t) \\
 &= \underbrace{p(s_t | \mathbf{x}_{1:t})}_{\alpha_t(s_t)} \underbrace{p(\mathbf{x}_{1:t})}_{\text{constant w.r.t. } s_t, s_{t+1}} \underbrace{p(x_{t+1} | s_{t+1})}_{\text{emission}} \underbrace{p(\mathbf{x}_{t+2:n} | s_{t+1})}_{\beta_{t+1}(s_{t+1})} \underbrace{p(s_{t+1} | s_t)}_{\text{transition}}
 \end{aligned}$$

Now we get:

$$p(s_t = i, s_{t+1} = j | \mathbf{x}) = \frac{p(s_t = i, s_{t+1} = j, \mathbf{x})}{p(\mathbf{x})} \propto \alpha_t(i) p(x_{t+1} | s_{t+1}) p(s_{t+1} | s_t) \beta_{t+1}(j).$$

To summarize the computation decomposes to:

- ◆  $\alpha_t(i)$ : past evidence,
- ◆  $p(s_{t+1} | s_t)$ : transition to the *current* state,
- ◆  $p(x_{t+1} | s_{t+1})$ : *current* observation,
- ◆  $\beta_{t+1}(j)$ : future observations.

The computational complexity remains  $\mathcal{O}(nK^2)$ .

## Most Probable Sequence: the Viterbi Algorithm

How to find the most probable sequence of hidden states given the sequence of features  $x$

$$\mathbf{s}^* \in \arg \max_{\mathbf{s} \in K^n} p(s_1) \prod_{t=2}^n p(s_t | s_{t-1}) \prod_{t=1}^n p(x_t | s_t)$$

Take the logarithm, with

$$g_1(s_1, x_1) = \log[p(s_1)p(x_1 | s_1)]$$

$$g_t(s_{t-1}, s_t, x_t) = \log[p(s_t | s_{t-1})p(x_t | s_t)], \quad t = 2, \dots, n$$

Solve the task

$$\mathbf{s}^* \in \arg \max_{\mathbf{s} \in K^n} \left[ g_1(s_1, x_1) + \sum_{t=2}^n g_t(s_{t-1}, s_t, x_t) \right]$$

by dynamic programming as before for Markov models.

The computational complexity is again  $\mathcal{O}(nK^2)$ .

## Viterbi vs. Smoothing

Compare the most probable sequence computer using the Viterbi algorithm:

$$\mathbf{s}^* \in \arg \max_{\mathbf{s} \in K^n} p(s_1) \prod_{t=2}^n p(s_t | s_{t-1}) \prod_{t=1}^n p(x_t | s_t)$$

with the smoothed marginals computed using forward-backward:

$$\hat{\mathbf{s}} \triangleq \left( \arg \max_{s_1 \in K} p(s_1 | \mathbf{x}), \arg \max_{s_2 \in K} p(s_2 | \mathbf{x}), \dots, \arg \max_{s_n \in K} p(s_n | \mathbf{x}) \right).$$

They are not the same! Example with two time steps and observations = hidden states:

	$x_1 = 0$	$x_1 = 1$	
$x_2 = 0$	0.04	0.3	0.34
$x_2 = 1$	0.36	0.3	0.66
	0.4	0.6	

Viterbi: (0, 1) is globally optimal, smoothed marginals (1, 1) locally optimal.

# Sampling Sequences from the Posterior

The task is to:

$$\mathbf{s}_{1:n}^s \sim p(\mathbf{s}_{1:n} | \mathbf{x}_{1:n})$$

We can follow these steps:

◆ Compute the *two-slice smoothed marginals*  $p(s_t, s_{t+1} | \mathbf{x})$ .

◆ Normalize them:

$$p(s_{t+1} | s_t, \mathbf{x}) = \frac{p(s_t, s_{t+1} | \mathbf{x})}{p(s_t | \mathbf{x})}.$$

◆ Sample from the initial pair of states:  $\mathbf{s}_{1:2}^s \sim p(s_1, s_2 | \mathbf{x})$ .

◆ Recursively sample  $s_{t+1}^s \sim p(s_{t+1} | s_t, \mathbf{x})$ .

Why not to sample all  $p(s_t | \mathbf{x}_{1:n})$  independently?

- We would break temporal consistency, produce invalid transitions, ignore correlations...

Note this is forward-backward pass and a additional sampling pass. It is possible to do it in a single pass...

## Learning algorithms for HMMs

### Supervised learning:

Given i.i.d. training data  $\mathcal{T}^m = \{(\mathbf{x}^j, \mathbf{s}^j) \in \mathcal{X}^n \times K^n \mid j = 1, \dots, m\}$ , estimate the parameters of the HMM by the maximum likelihood estimator.

This is done by simple “counting” as before for Markov models.

- ◆ Denote by  $a_t(s_{t-1} = j, s_t = i)$  the number of examples in  $\mathcal{T}^m$  for which  $s_{t-1} = j$  and  $s_t = i$ .
- ◆ Denote by  $b_t(s_t = i, x_t = x)$  the number of examples in  $\mathcal{T}^m$  for which  $s_t = i$  and  $x_t = x$ .

The estimates for the model parameters are then given by

$$p(s_t = i \mid s_{t-1} = j) = \frac{a_t(s_{t-1} = j, s_t = i)}{\sum_{i'} a_t(s_{t-1} = j, s_t = i')}, \quad p(x_t = x \mid s_t = i) = \frac{b_t(s_t = i, x_t = x)}{\sum_{x'} b_t(s_t = k, x_t = x')}$$

**Remark 6.** This is easy to generalise for the case that  $\mathcal{X} = \mathbb{R}^m$  and  $p(x_t \mid s_t)$  is from some parametric distribution family.

## Learning algorithms for HMMs

### Unsupervised learning:

Given i.i.d. training data  $\mathcal{T}^m = \{\mathbf{x}^j \in \mathcal{X}^n \mid j = 1, \dots, m\}$ , estimate the parameters of the HMM by the maximum likelihood estimator.

We apply the EM-algorithm (aka Baum-Welch algorithm): Initialise the model, then iterate

**E-step** Use the current model estimate and compute the two-slice smoothed marginals

$$\alpha_{\mathbf{x}}(s_t, s_{t-1}) = p(s_t, s_{t-1} \mid \mathbf{x}) \quad s_{t-1}, s_t \in K$$

for each training example  $\mathbf{x} \in \mathcal{T}^m$  and all positions  $i = 2, \dots, n$ .

**M-step** Use the  $\alpha$ -s as soft labels for computing the counts as in supervised learning

$$a_t(s_{t-1}, s_t) = \sum_{\mathbf{x} \in \mathcal{T}^m} \alpha_{\mathbf{x}}(s_t, s_{t-1}), \quad b_t(s_t, x) = \sum_{\mathbf{x} \in \mathcal{T}(x_t=x)} \alpha_{\mathbf{x}}(s_t)$$

where  $\mathcal{T}(x_t = x)$  is the set of training examples with  $x_t = x$  and  $\alpha_{\mathbf{x}}(s_t)$  is given by  $\alpha_{\mathbf{x}}(s_t) = \sum_{s_{t-1}} \alpha_{\mathbf{x}}(s_t, s_{t-1})$ .

## Summary

- ◆ Hidden Markov models are statistical models for pairs of processes (sequences)  $(s, x)$ , where  $s$  is a sequence of hidden states and not directly observable.
- ◆ Markov models and HMMs are exponential families.
- ◆ Their parameters can be estimated by supervised learning using either Maximum likelihood estimates or Empirical risk minimization.
- ◆ Their parameters can be estimated by unsupervised learning using the EM-algorithm.
- ◆ All important inference and learning algorithms for HMMs have time complexity linear in the length of the sequence and quadratic in the size of the hidden state space.
- ◆ What if the space of hidden states is very large, e.g.  $s_t \in \mathbb{Z}^k$ ? We can use recurrent neural networks for modelling  $p(s_t | s_{t-1})$ , e.g. Gated recurrent Units (GRU) or Long short-term memory models (LSTM).

