Sequence modelling

Marco Saerens (UCL)

machine learning group @ UCL

UCL Université catholique de Louvain

Université partenaire de l'Académie universitaire 'Louvain'

Slides references

- Many slides and figures have been adapted from the slides associated to the following books:
  - Alpaydin (2004), Introduction to machine learning. The MIT Press.
  - Duda, Hart & Stork (2000), Pattern classification, John Wiley & Sons.
  - Han & Kamber (2006), Data mining: concepts and techniques, 2nd ed. Morgan Kaufmann.
  - Tan, Steinbach & Kumar (2006), Introduction to data mining. Addison Wesley.
  - Grinstead and Snell's Introduction to Probability (2006; GNU Free Documentation License)
- As well as from Wikipedia, the free online encyclopedia
- I really thank these authors for their contribution to machine learning and data mining teaching

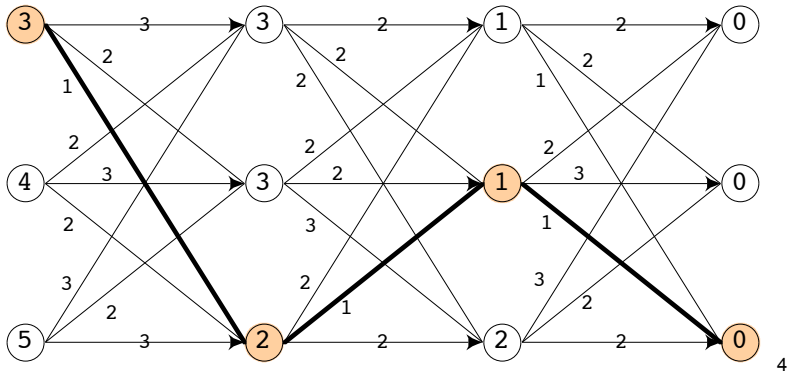
2

Dynamic programming and edit-distance

Marco Saerens (UCL)

Dynamic programming

- Suppose we have a lattice with N levels:



4

Dynamic programming

- The problem is to reach level N
- From level 1
- With minimum cost
- = shortest-path problem

5

Dynamic programming

- Some definitions

$$s_k = \text{variable containing the state at level } k$$

- The local cost associated to the decision to jump to the state $s_k = j$ at level k

$$d(s_k = j \mid s_{k-1} = i)$$

- Given that we were in state $s_{k-1} = i$ at level $k-1$

6

Dynamic programming

- The total cost of a path $(s_0, s_1, \ldots, s_N)$ is
$$D(s_0, s_1, \ldots, s_N) = \sum_{i=1}^{N} d(s_i \mid s_{i-1})$$
- The optimal cost when starting from state s_0 is
$$\begin{aligned} D^*(s_0) &= \min_{(s_1,\ldots,s_N)} \{D(s_0, s_1, \ldots, s_N)\} \\ &= \min_{(s_1,\ldots,s_N)} \left\{ \sum_{i=1}^{N} d(s_i \mid s_{i-1}) \right\} \end{aligned}$$

7

Dynamic programming

- The optimal cost, whatever the initial state, is
$$D^* = \min_{s_0} \{D^*(s_0)\}$$
- The optimal cost when starting from some intermediate state s_k
$$D^*(s_k) = \min_{(s_{k+1},\ldots,s_N)} \left\{ \sum_{i=k+1}^{N} d(s_i \mid s_{i-1}) \right\}$$

8

Dynamic programming

- Here are the recurrence relations allowing to obtain the optimal cost:

$$D^*(s_N) = 0$$

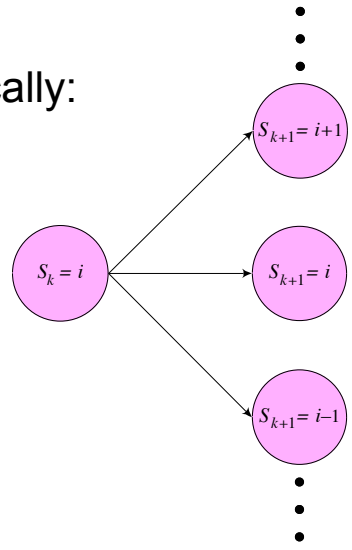
$$D^*(s_k = i) = \min_{s_{k+1}} \{d(s_{k+1} \mid s_k = i) + D^*(s_{k+1})\}$$

$$D^* = \min_{s_0} \{D^*(s_0)\}$$

9

Dynamic programming

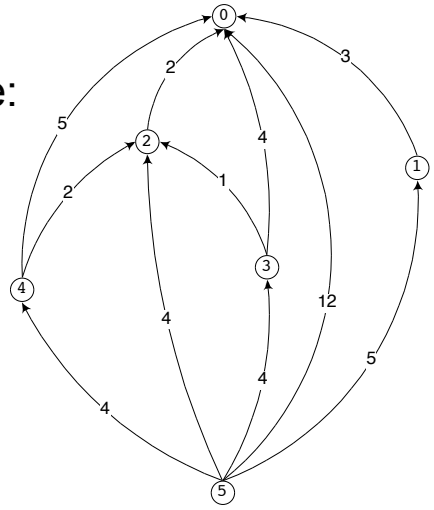
- Graphically:



10

Dynamic programming

- Example:



$$D^*(s_{N-1} = i) = \min_{s_N} \{d(s_N \mid s_{N-1} = i)\}$$

11

Dynamic programming

- Proof:

$$\begin{aligned}
D^*(s_k) &= \min_{(s_{k+1},\ldots,s_N)} \left\{ \sum_{i=k+1}^{N} d(s_i \mid s_{i-1}) \right\} \\
&= \min_{(s_{k+1},\ldots,s_N)} \left\{ d(s_{k+1} \mid s_k) + \sum_{i=k+2}^{N} d(s_i \mid s_{i-1}) \right\} \\
&= \min_{s_{k+1}} \left\{ \min_{(s_{k+2},\ldots,s_N)} \left\{ d(s_{k+1} \mid s_k) + \sum_{i=k+2}^{N} d(s_i \mid s_{i-1}) \right\} \right\} \\
&= \min_{s_{k+1}} \left\{ d(s_{k+1} \mid s_k) + \underbrace{\min_{(s_{k+2},\ldots,s_N)} \left[\sum_{i=k+2}^{N} d(s_i \mid s_{i-1}) \right]}_{D^*(s_{k+1})} \right\} \\
&= \min_{s_{k+1}} \{d(s_{k+1} \mid s_k) + D^*(s_{k+1})\}
\end{aligned}$$

Dynamic programming

- In a symmetric way:

$$D^*(s_0) = 0$$

$$D^*(s_k = i) = \min_{s_{k-1}} \{d(s_k = i \mid s_{k-1}) + D^*(s_{k-1})\}$$

$$D^* = \min_{s_N} \{D^*(s_N)\}$$

13

Dynamic programming

- Now, if there are jumps bypassing some levels,

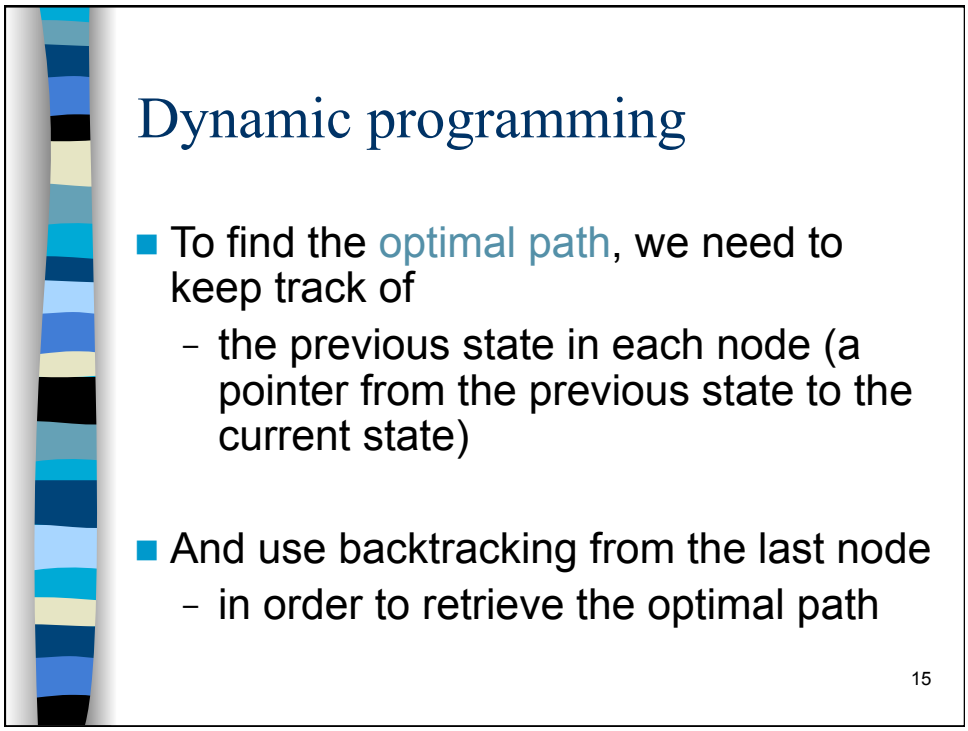
$$D^*(s_N) = 0$$

$$D^*(s_k = i) = \min_{\{s\}|s_k = i \to s} \{d(s \mid s_k = i) + D^*(s)\}$$

$$D^* = \min_{s_0} \{D^*(s_0)\}$$

- where $\{s\}|s_k = i \to s$ is the set of states to which there is a direct jump from $s_k = i$

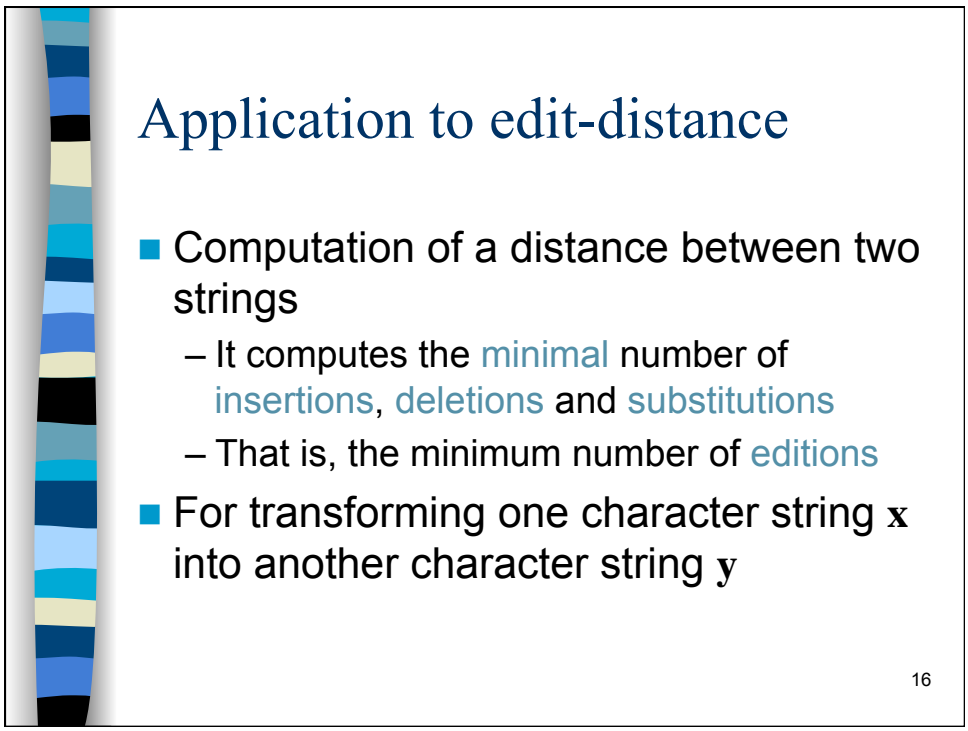
14



Dynamic programming

- To find the optimal path, we need to keep track of
  - the previous state in each node (a pointer from the previous state to the current state)
- And use backtracking from the last node
  - in order to retrieve the optimal path

15



Application to edit-distance

- Computation of a distance between two strings
  - It computes the minimal number of insertions, deletions and substitutions
  - That is, the minimum number of editions
- For transforming one character string **x** into another character string **y**

16

Application to edit-distance

- Also called the "Levenstein distance"
- We thus have two character strings

$$\begin{cases} \mathbf{y} = y_1 y_2 \dots y_{|\mathbf{y}|} \\ \mathbf{x} = x_1 x_2 \dots x_{|\mathbf{x}|} \end{cases}$$

- x_i being the character i of string $\mathbf{x}$

17

Application to edit-distance

- The length of the string $\mathbf{x}$ is denoted by $|\mathbf{x}|$
- In general, we have

$$|\mathbf{x}| \neq |\mathbf{y}|$$

- The substring of $\mathbf{x}$, beginning at character i and ending at j is defined by

$$\mathbf{x}_i^j = x_i x_{i+1} \dots x_j$$

18

Application to edit-distance

- We will now read the characters of $\mathbf{x}$ one by one
- In order to construct the string $\mathbf{y}$
- To this aim, we define the three editing operations:
  - Insertion of a character in $\mathbf{y}$ (without taking any character from $\mathbf{x}$)
  - Deletion of a character from $\mathbf{x}$ (without concatenating it to $\mathbf{y}$)
  - Substitution of a character from $\mathbf{y}$ by one of $\mathbf{x}$

19

Application to edit-distance

- First convention:

$$\mathbf{x}_i^{|\mathbf{x}|}$$

  - Means that we have read (extracted) the $i-1$ first characters of $\mathbf{x}$
  - Thus, $\mathbf{x}$ has been cutted from its $i-1$ first characters
  - They have been taken in order to construct $\mathbf{y}$

20

Application to edit-distance

- Second convention:

$$\mathbf{y}_0^j$$

  - Means that the j first characters of $\mathbf{y}$ have been transcribed
- We progressively read the first characters of $\mathbf{x}$ in order to build $\mathbf{y}$

21

Application to edit-distance

- It corresponds to a process with levels (steps) and states
  - We will now apply dynamic programming

$$\begin{cases} \text{One state corresponds to } (\mathbf{x}_i^{|\mathbf{x}|}, \mathbf{y}_0^j) \\ \text{The level } k \text{ corresponds to } i + j = \text{const} = k \end{cases}$$

  - One state is characterized by a couple (i, j)

22

Application to edit-distance

- Here is the formal definition of the three operations

$$\begin{cases} \text{Insertion with respect to } \mathbf{x}\text{: } (\mathbf{x}_i^{|\mathbf{x}|}, \mathbf{y}_0^{j-1}) \rightarrow (\mathbf{x}_i^{|\mathbf{x}|}, \mathbf{y}_0^{j}) \\ \text{Deletion with respect to } \mathbf{x}\text{: } (\mathbf{x}_{i-1}^{|\mathbf{x}|}, \mathbf{y}_0^{j}) \rightarrow (\mathbf{x}_i^{|\mathbf{x}|}, \mathbf{y}_0^{j}) \\ \text{Substitution with respect to } \mathbf{x}\text{: } (\mathbf{x}_{i-1}^{|\mathbf{x}|}, \mathbf{y}_0^{j-1}) \rightarrow (\mathbf{x}_i^{|\mathbf{x}|}, \mathbf{y}_0^{j}) \end{cases}$$

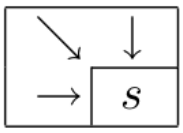
  - For the two first operations, we are jumping from level k to level k+1
  - For the last one, we directly jump to level k +2 (one level is passed)

23

Application to edit-distance

- This situation can be represented in a two-dimensional table
  - One level is represented by

    $(i + j) = \text{constant}$
  - One state is represented by (i, j)
  - One operation corresponds to a valid transition in this table



24

Application to edit-distance

- Exemple of a table computing the distance between “livre” and “lire”

		y					
		∅	l	i	v	r	e
	∅	0	1	2	3	4	5
	l	1	0	1	2	3	4
x	i	2	1	0	1	2	3
	r	3	2	1	1	1	2
	e	4	3	2	2	2	1

25

Application to edit-distance

- Each level corresponds to a diagonal of the table:
  - $(i + j) = \text{constant}$

		y					
		∅	l	i	v	r	e
	∅	0	1	2	3	4	5
	l	1	0	1	2	3	4
x	i	2	1	0	1	2	3
	r	3	2	1	1	1	2
	e	4	3	2	2	2	1

26

Application to edit-distance

- A cost (or penalty) is associated to each operation (insertion, deletion, substitution); for instance

$$\begin{cases} \delta_{\text{ins}}(y_i) = 1 \\ \delta_{\text{del}}(x_i) = 1 \\ \delta_{\text{sub}}(x_i, y_j) = \delta_{ij} \end{cases}$$

- with

$$\begin{cases} \delta_{ij} = 1 \text{ si } x_i \neq y_j \\ \delta_{ij} = 0 \text{ si } x_i = y_j \end{cases}$$

27

Application to edit-distance

- The dynamic programming formula can be applied to this problem:
  - Initialization:

$$D^*(\mathbf{x}_0^{|\mathbf{x}|}, \mathbf{y}_0^0) = 0$$

  - Then:

$$D^*(\mathbf{x}_i^{|\mathbf{x}|}, \mathbf{y}_0^j) = \min \begin{cases} D^*(\mathbf{x}_i^{|\mathbf{x}|}, \mathbf{y}_0^{j-1}) + 1 \\ D^*(\mathbf{x}_{i-1}^{|\mathbf{x}|}, \mathbf{y}_0^j) + 1 \\ D^*(\mathbf{x}_{i-1}^{|\mathbf{x}|}, \mathbf{y}_0^{j-1}) + \delta_{ij} \end{cases}$$

28

Application to edit-distance

- And finally:

$$\mathrm{dist}(\mathbf{x}, \mathbf{y}) = D^*(\mathbf{x}_{|\mathbf{x}|}^{|\mathbf{x}|}, \mathbf{y}_0^{|\mathbf{y}|})$$

- This value is the edit-distance or Levenstein distance

29

Application to edit-distance

- One example:

		y					
		∅	l	i	v	r	e
	∅	0	1	2	3	4	5
	l	1	0	1	2	3	4
x	i	2	1	0	1	2	3
	r	3	2	1	1	1	2
	e	4	3	2	2	2	1

- dist(lire, livre) = 1

30

Application to edit-distance

- In order to find the optimal path (optimal sequence of operations)
  - A pointer to the previous state has to be maintained for each cell of the table
  - The full path can be found by backtracking from the final state
- Numerous extensions and generalizations of this basic algorithm have been developed

31

Application to edit-distance

- Notice that the related quantity, the longest common subsequence, $\mathrm{lcs}(\mathbf{x},\mathbf{y})$, can be obtained by (no proof provided)

$$\begin{aligned} \mathrm{dist}(\mathbf{x},\mathbf{y}) &= \mathrm{lcs}(\mathbf{x},\mathbf{x}) + \mathrm{lcs}(\mathbf{y},\mathbf{y}) - 2\,\mathrm{lcs}(\mathbf{x},\mathbf{y}) \\ &= |\mathbf{x}| + |\mathbf{y}| - 2\,\mathrm{lcs}(\mathbf{x},\mathbf{y}) \end{aligned}$$

- and thus

$$\mathrm{lcs}(\mathbf{x},\mathbf{y}) = \frac{1}{2}(|\mathbf{x}| + |\mathbf{y}| - \mathrm{dist}(\mathbf{x},\mathbf{y}))$$

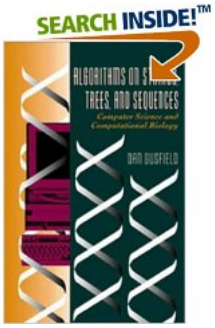
32

Application to edit-distance

- Thus, since
  - dist(lire, livre) = 1
- We have
  - lcs(lire, livre) = 0.5 (4 + 5 – 1) = 4

33

Application to edit-distance



34

A brief introduction to Markov chains

Introduction to Markov chains

- We have a set of states, $S = \{1, 2, \ldots, n\}$
  - $s_t = k$ means that the process, or system, is in state k at time t
- Example:
  - We take as states the kind of weather R (rain), N (nice), and S (snow)

$$\mathbf{P} = \begin{array}{c} \\ \mathrm{R} \\ \mathrm{N} \\ \mathrm{S} \end{array} \begin{array}{c} \begin{array}{ccc} \mathrm{R} & \mathrm{N} & \mathrm{S} \end{array} \\ \begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/4 & 1/4 & 1/2 \end{pmatrix} \end{array}$$

36

Introduction to Markov chains

- Markov chains are models of sequential discrete-time and discrete-state stochastic processes

37

Introduction to Markov chains

- The entries of this matrix **P** are p_{ij} with

$$p_{ij} = \mathrm{P}(s_{t+1} = j | s_t = i)$$

- We assume that the probability of jumping to a state only depends on the current state
  - And not on the past, before this state
  - This is the Markov property

38

Introduction to Markov chains

- The matrix **P** is called the one-step transition probabilities matrix
  - It is a stochastic matrix
  - The row sums are equal to 1
- We also assume that these transition probabilities are stationary
  - That is, independent of time
- Let us now compute $\mathrm{P}(s_{t+2} = j \mid s_t = i)$:

39

Introduction to Markov chains

$$
\begin{aligned}
\mathrm{P}(s_{t+2} = j | s_t = i) &= \sum_{k=1}^{n} \mathrm{P}(s_{t+2} = j, s_{t+1} = k | s_t = i) \\
&= \sum_{k=1}^{n} \mathrm{P}(s_{t+2} = j | s_t = i, s_{t+1} = k) \mathrm{P}(s_{t+1} = k | s_t = i) \\
&= \sum_{k=1}^{n} \mathrm{P}(s_{t+2} = j | s_{t+1} = k) \mathrm{P}(s_{t+1} = k | s_t = i) \\
&= \sum_{k=1}^{n} p_{kj} p_{ik} \\
&= [\mathbf{P}^2]_{ij}
\end{aligned}
$$

40

Introduction to Markov chains

- The matrix $\mathbf{P}^2$ is the two-steps transition probabilities matrix
- By induction, $\mathbf{P}^\tau$ is the τ-steps transition probabilities matrix containing elements

$$p_{ij}^{(\tau)} = \mathrm{P}(s_{t+\tau} = j | s_t = i)$$

41

Introduction to Markov chains

- If $\mathbf{x}(t)$ is the column vector containing the probability distribution of finding the process in each state of the Markov chain at time step t, we have

$$\begin{aligned} x_i(t) &= \mathrm{P}(s_t = i) \\ &= \sum_{k=1}^{n} \mathrm{P}(s_t = i, s_{t-1} = k) \\ &= \sum_{k=1}^{n} \mathrm{P}(s_t = i | s_{t-1} = k) \mathrm{P}(s_{t-1} = k) \\ &= \sum_{k=1}^{n} p_{ki}\, x_k(t-1) \end{aligned}$$

42

Introduction to Markov chains

- In matrix form, $\mathbf{x}(t) = \mathbf{P}^{\mathrm{T}}\,\mathbf{x}(t-1)$
- Or, in function of the initial distribution,
  $\mathbf{x}(t) = (\mathbf{P}^{\mathrm{T}})^t\,\mathbf{x}(0)$
- Now, $x_j(t) = \mathbf{x}(t)^{\mathrm{T}}\,\mathbf{e}_j = \mathbf{x}(0)^{\mathrm{T}}\,\mathbf{P}^t\,\mathbf{e}_j$

43

Introduction to Markov chains

- Thus, when starting from state i, $\mathbf{x}(0) = \mathbf{e}_i$, and

$$x_j(t \mid s_0 = i) = x_{j|i}(t) = (\mathbf{e}_i)^{\mathrm{T}}\,\mathbf{P}^t\,\mathbf{e}_j$$

- It is the probability of observing the process in state j at time t when starting from state i at time $t = 0$

44

Introduction to Markov chains

$$\mathbf{P}^1 = \begin{array}{c} \\ \text{Rain} \\ \text{Nice} \\ \text{Snow} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Rain} & \text{Nice} & \text{Snow} \end{array} \\ \begin{pmatrix} .500 & .250 & .250 \\ .500 & .000 & .500 \\ .250 & .250 & .500 \end{pmatrix} \end{array}$$

$$\mathbf{P}^2 = \begin{array}{c} \\ \text{Rain} \\ \text{Nice} \\ \text{Snow} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Rain} & \text{Nice} & \text{Snow} \end{array} \\ \begin{pmatrix} .438 & .188 & .375 \\ .375 & .250 & .375 \\ .375 & .188 & .438 \end{pmatrix} \end{array}$$

$$\mathbf{P}^3 = \begin{array}{c} \\ \text{Rain} \\ \text{Nice} \\ \text{Snow} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Rain} & \text{Nice} & \text{Snow} \end{array} \\ \begin{pmatrix} .406 & .203 & .391 \\ .406 & .188 & .406 \\ .391 & .203 & .406 \end{pmatrix} \end{array}$$

$$\mathbf{P}^4 = \begin{array}{c} \\ \text{Rain} \\ \text{Nice} \\ \text{Snow} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Rain} & \text{Nice} & \text{Snow} \end{array} \\ \begin{pmatrix} .402 & .199 & .398 \\ .398 & .203 & .398 \\ .398 & .199 & .402 \end{pmatrix} \end{array}$$

$$\mathbf{P}^5 = \begin{array}{c} \\ \text{Rain} \\ \text{Nice} \\ \text{Snow} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Rain} & \text{Nice} & \text{Snow} \end{array} \\ \begin{pmatrix} .400 & .200 & .399 \\ .400 & .199 & .400 \\ .399 & .200 & .400 \end{pmatrix} \end{array}$$

$$\mathbf{P}^6 = \begin{array}{c} \\ \text{Rain} \\ \text{Nice} \\ \text{Snow} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Rain} & \text{Nice} & \text{Snow} \end{array} \\ \begin{pmatrix} .400 & .200 & .400 \\ .400 & .200 & .400 \\ .400 & .200 & .400 \end{pmatrix} \end{array}$$

45

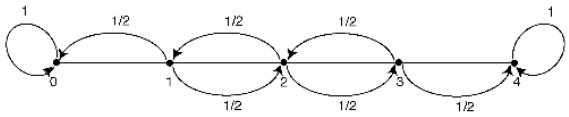
Introduction to Markov chains

- Let us now introduce absorbing Markov chains
- A state i of a Markov chain is called absorbing if it is impossible to leave it, $p_{ii} = 1$
- An absorbing Markov chain chain is a Markov chain containing absorbing states
  - The other states being called transient (TR)

46

Introduction to Markov chains

- Let us take an example: the drunkard's walk (from Grinstead and Snell)



$$\mathbf{P} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

47

Introduction to Markov chains

- The transition matrix can be put in canonical form:

$$\mathbf{P} = \begin{array}{c} \\ \text{TR.} \\ \\ \text{ABS.} \end{array} \begin{array}{c} \begin{array}{cc} \text{TR.} & \text{ABS.} \end{array} \\ \left(\begin{array}{c|c} \mathbf{Q} & \mathbf{R} \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right) \end{array}$$

- $\mathbf{Q}$ is the transition matrix between transient states

48

Introduction to Markov chains

- $\mathbf{R}$ is the transition matrix between transient and absorbing states
- Both $\mathbf{Q}$ and $\mathbf{R}$ are sub-stochastic
  - Their row sums are ≤ 1 and at least one row sum is < 1

49

Introduction to Markov chains

- Now, $\mathbf{P}^t$ can be computed as

$$\mathbf{P}^t = \begin{array}{c} \\ \text{TR.} \\ \text{ABS.} \end{array} \begin{array}{c} \begin{array}{cc} \text{TR.} & \text{ABS.} \end{array} \\ \left(\begin{array}{c|c} \mathbf{Q}^t & \cdots \\ \hline \mathbf{0} & \mathbf{I} \end{array} \right) \end{array}$$

- Since $\mathbf{Q}$ is sub-stochastic, it can be shown that: $\mathbf{Q}^t \to \mathbf{0}$ as $t \to \infty$

50

Introduction to Markov chains

- The matrix

$$\mathbf{N} = \mathbf{I} + \mathbf{Q} + \mathbf{Q}^2 + \mathbf{Q}^3 + \cdots = (\mathbf{I} - \mathbf{Q})^{-1}$$

  – is called the fundamental matrix of the absorbing Markov chain

- Let us interpret the elements $n_{ij} = [\mathbf{N}]_{ij}$ of the fundamental matrix, where i, j are transient states

51

Introduction to Markov chains

- Recall that since i, j are transient states,
  – we have $x_j(t \mid s_0 = i) = x_{j|i}(t) = (\mathbf{e}_i)^{\mathrm{T}} \mathbf{P}^t \mathbf{e}_j = (\mathbf{e}_i)^{\mathrm{T}} \mathbf{Q}^t \mathbf{e}_j$
- Thus, entry i, j of the matrix $\mathbf{N}$ for transient states, n_{ij}, is

$$\begin{aligned} n_{ij} &= \mathbf{e}_i^{\mathrm{T}} \mathbf{N} \mathbf{e}_j \\ &= \mathbf{e}_i^{\mathrm{T}} \left(\sum_{t=0}^{\infty} \mathbf{Q} \right) \mathbf{e}_j \\ &= \sum_{t=0}^{\infty} (\mathbf{e}_i^{\mathrm{T}} \mathbf{Q} \mathbf{e}_j) \\ &= \sum_{t=0}^{\infty} x_{j|i}(t) \end{aligned}$$

52

Introduction to Markov chains

- Thus, element n_{ij} contains the expected number of passages through transient state j when starting from transient state i
- The expected number of visits (and therefore steps) before being absorbed when starting from each state is

$$\mathbf{n} = \mathbf{Ne}$$

53

Introduction to Markov chains

- For the drunkard's walk,

$$\mathbf{P} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \hline 0 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc|cc} 1 & 2 & 3 & 0 & 4 \end{array} \\ \left(\begin{array}{ccc|cc} 0 & 1/2 & 0 & 1/2 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 1/2 \\ \hline 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right) \end{array}.$$

54

Introduction to Markov chains

$$\mathbf{Q} = \begin{pmatrix} 0 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1/2 & 0 \end{pmatrix}$$

$$\mathbf{I} - \mathbf{Q} = \begin{pmatrix} 1 & -1/2 & 0 \\ -1/2 & 1 & -1/2 \\ 0 & -1/2 & 1 \end{pmatrix}$$

$$\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 3/2 & 1 & 1/2 \\ 1 & 2 & 1 \\ 1/2 & 1 & 3/2 \end{pmatrix} \end{array}$$

55

Introduction to Markov chains

$$\mathbf{n} = \mathbf{Ne} = \begin{pmatrix} 3/2 & 1 & 1/2 \\ 1 & 2 & 1 \\ 1/2 & 1 & 3/2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 3 \\ 4 \\ 3 \end{pmatrix}$$

56

Introduction to Markov chains

- We can also compute absorption probabilities from each starting state
- We can compute the probability of being absorbed by absorbing state j given that we started in transient state i by

$$b_{ij} = \sum_{t=0}^{\infty} \sum_{\substack{k=1 \\ k \in \mathrm{TR}}}^{n_{tr}} x_{k|i}(t)\, r_{kj}$$

where n_{tr} is the number of transient states and the sum over k is taken on the set of transient states (TR) only

57

Introduction to Markov chains

- The formula states that the probability of reaching absorbing node j at time $(t+1)$ is given by
  - the probability of passing through any state k at t and then jumping to state j from k at $(t+1)$
- The absorption probability is then given by taking the sum over all possible time steps

58

Introduction to Markov chains

- Let us compute this quantity

$$
\begin{aligned}
b_{ij} &= \sum_{t=0}^{\infty} \sum_{\substack{k=1 \\ k \in \mathrm{TR}}}^{n_{tr}} x_{k|i}(t)\, r_{kj} \\
&= \sum_{\substack{k=1 \\ k \in \mathrm{TR}}}^{n_{tr}} \left[\sum_{t=0}^{\infty} \mathbf{e}_i^{\mathrm{T}} \mathbf{Q}^t \mathbf{e}_k \right] r_{kj} \\
&= \sum_{\substack{k=1 \\ k \in \mathrm{TR}}}^{n_{tr}} n_{ik}\, r_{kj} \\
&= [\mathbf{NR}]_{ij}
\end{aligned}
$$

59

Introduction to Markov chains

- The absorption probabilities are put in the $\mathbf{B}$ matrix
- Let us reconsider the drunkard's example

$$
\mathbf{R} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 4 \end{array} \\ \begin{pmatrix} 1/2 & 0 \\ 0 & 0 \\ 0 & 1/2 \end{pmatrix} \end{array}
$$

60

Introduction to Markov chains

$$\mathbf{B} = \mathbf{NR} = \begin{pmatrix} 3/2 & 1 & 1/2 \\ 1 & 2 & 1 \\ 1/2 & 1 & 3/2 \end{pmatrix} \begin{pmatrix} 1/2 & 0 \\ 0 & 0 \\ 0 & 1/2 \end{pmatrix}$$

$$= \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 4 \end{array} \\ \begin{pmatrix} 3/4 & 1/4 \\ 1/2 & 1/2 \\ 1/4 & 3/4 \end{pmatrix} \end{array}$$

61

Introduction to Markov chains

- Some additional definitions
  - If, in a Markov chain, it is possible to go to every state from each state, the Markov chain is called irreducible
  - Moreover, the Markov chain is called regular if some power of the transition matrix has only positive (non-0) elements

62

Introduction to Markov chains

- It can further be shown that the powers $\mathbf{P}^t$ of a regular transition matrix tend to a matrix with all rows the same

$$\lim_{t\to\infty} \mathbf{P}^t = \begin{pmatrix} \boldsymbol{\pi}^{\mathrm{T}} \\ \boldsymbol{\pi}^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\pi}^{\mathrm{T}} \end{pmatrix}$$

63

Introduction to Markov chains

- Moreover, the limiting probability distribution of states is independent of the initial state:

$$\lim_{t\to\infty} \mathbf{x}(t) = \boldsymbol{\pi}$$

64

Introduction to Markov chains

- The stationary vector $\boldsymbol{\pi}$ is the left eigenvector of $\mathbf{P}$, corresponding to eigenvalue 1 and normalized to a probability vector:

$$\begin{aligned} \boldsymbol{\pi} &= \lim_{t\to\infty} \mathbf{x}(t) \\ &= \lim_{t\to\infty} \mathbf{P}^{\mathrm{T}}\mathbf{x}(t-1) \\ &= \mathbf{P}^{\mathrm{T}} \lim_{t\to\infty} \mathbf{x}(t-1) \\ &= \mathbf{P}^{\mathrm{T}}\boldsymbol{\pi} \end{aligned}$$

65

Introduction to Markov chains

- It provides the probability of finding the process in each state on the long run
- One can prove that this vector is unique for a regular Markov chain

66

Introduction to Markov chains

- Notice that the fundamental matrix for absorbing chains can be generalized
- To regular chains
  - It, for instance, allows to compute the average first-passage times in matrix form
  - See, for instance, Grinstead and Snell

67

Application to marketing

Application to marketing

- Suppose we have the following model
  - We have a number n of customer clusters or segments
  - Based, for instance, on RFM (Recency, Frequency, Monetary value)
- Each cluster is a state of a Markov chain
- The last (nth) cluster corresponds to lost customers
  - It is absorbing and generates no benefit

69

Application to marketing

- Each month, we observe the movements from cluster to cluster
- Transition probabilities are estimated
  - by counting the observed frequencies of jumping from one state to another in the past
  - This provides the entries of the transition probabilities matrix

70

Application to marketing

- Suppose also there is an average profit, m_i per month, associated to each customer in state i
  - which could be negative
- There is also a discounting factor:
  $0 < \gamma < 1$
- The expected profit on an infinite time horizon can be computed

71

Application to marketing

- It is given by

$$\overline{m} = \sum_{t=0}^{\infty} \gamma^t \sum_{i=1}^{n} x_i(t)\, m_i$$

  - It provides the expected profit on a infinite horizon

72

Application to marketing

- Which finally provides

$$
\begin{aligned}
\overline{m} &= \sum_{t=0}^{\infty} \gamma^t \sum_{i=1}^{n} x_i(t)\, m_i \\
&= \sum_{t=0}^{\infty} \gamma^t \mathbf{m}^{\mathrm{T}} (\mathbf{P}^{\mathrm{T}})^t \mathbf{x}(0) \\
&= \sum_{t=0}^{\infty} \gamma^t \mathbf{x}^{\mathrm{T}}(0) \mathbf{P}^t \mathbf{m} \\
&= \mathbf{x}^{\mathrm{T}}(0) (\sum_{t=0}^{\infty} \gamma^t \mathbf{P}^t) \mathbf{m} \\
&= \mathbf{x}^{\mathrm{T}}(0) (\mathbf{I} - \gamma \mathbf{P})^{-1} \mathbf{m}
\end{aligned}
$$

73

Application to marketing

- This is an example of the computation of the lifetime value of a customer
- Which is the expected profit provided by the customer until it leaves the company

74

A brief introduction to dynamic time warping

Marco Saerens, UCL
Alain Soquet, ULB

Dynamic time warping

- Context: word recognition
- Suppose we have a database of word templates or references
  - Someone pronounces a word which is recorded and analysed
  - We want to recover the nearest template word in the database = nearest neighbor
  - Which will be the recognized word

76

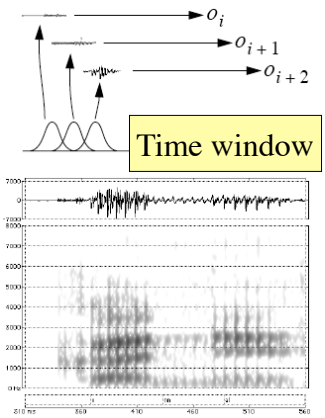
Dynamic time warping

- However, the timing in which the word has been pronounced can differ greatly
- Hence, we have to account for distortions or warping of the signal
  => Dynamic time warping

77

Dynamic time warping

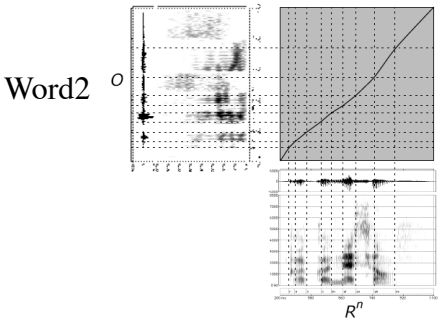
- Here is an example: a spectrogram is produced



78

Dynamic time warping

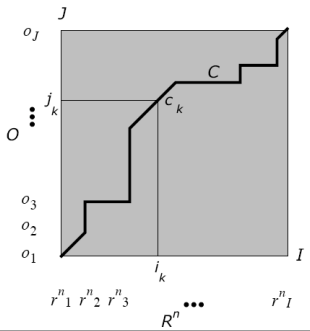
- We have to compare word1 and word2
  - Thus, align the two signals



79

Dynamic time warping

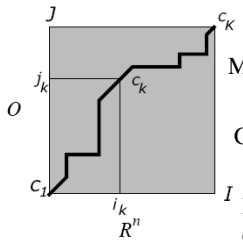
- The two signals are aligned by
  - Defining a distance $d(i,j)$ between two frames, for instance the Euclidean distance
  - Defining a time alignment that allows for warpings



80

Dynamic time warping

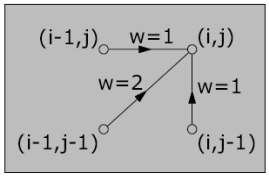
- We have to add monotonicity constraints in order to obtain meaningful alignments



Monotonicity $\begin{cases} i_k \geq i_{k-1} \\ j_k \geq j_{k-1} \end{cases}$

Continuity $\begin{cases} i_k - i_{k-1} \leq 1 \\ j_k - j_{k-1} \leq 1 \end{cases}$

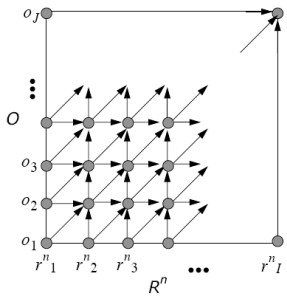
Boundary conditions $\begin{cases} i_1 = 1, j_1 = 1 \\ i_K = I, j_K = J \end{cases}$



81

Dynamic time warping

- The problem can be solved by dynamic programming
- By considering only the valid transitions



82

Dynamic time warping

- Here are the dynamic programming recurence relations

$$g(1,1) = d(r_1^n, o_1)$$

$$g(i,j) = min\begin{Bmatrix} g(i-1,j) + d(r_i^n, o_j) \\ g(i-1,j-1) + 2d(r_i^n, o_j) \\ g(i,j-1) + d(r_i^n, o_j) \end{Bmatrix} \quad \text{for} \quad \begin{cases} i = 1, 2, \ldots, I \\ j = 1, 2, \ldots, J \end{cases}$$

$$D(R^n, O) = \frac{1}{I+J} g(I,J)$$

83

A brief introduction to hidden Markov models

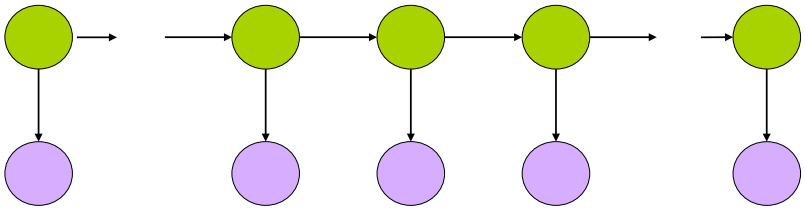
Marco Saerens, UCL

Hidden Markov models

- Most slides are adapted from David Meir Blei's slides, SRI International
- Thanks to David Meir Blei

85

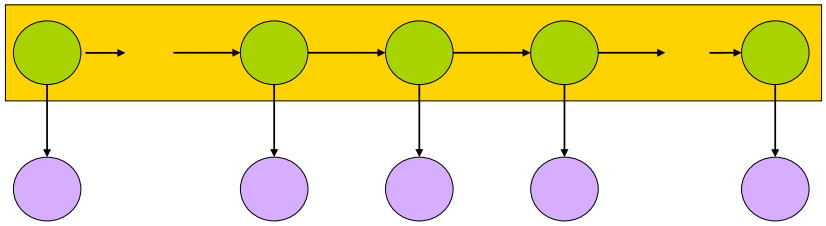
What is an HMM ?



- Graphical model
- Green circles indicate states
- Purple circles indicate observations
- Arrows indicate probabilistic dependencies between states

86

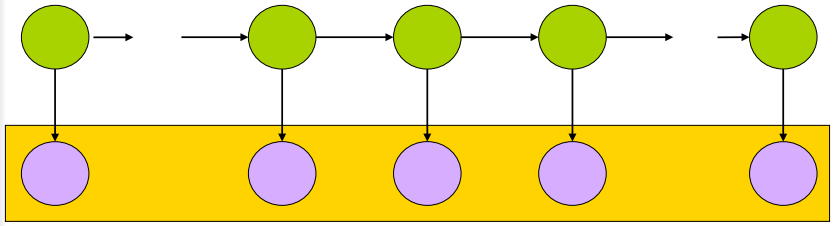
What is an HMM?



- Green circles are **hidden**, unobserved, **states** (n in total)
- Dependent only on **the previous state** (arrow)
- "The past is independent of the future given the present": Markov property

87

What is an HMM ?

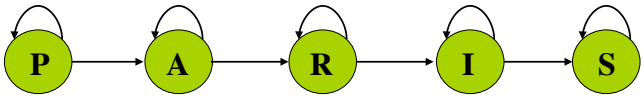


- Purple nodes are **discrete observations** (p in total)
- They dependent only on their corresponding **hidden state** in which they were generated

88

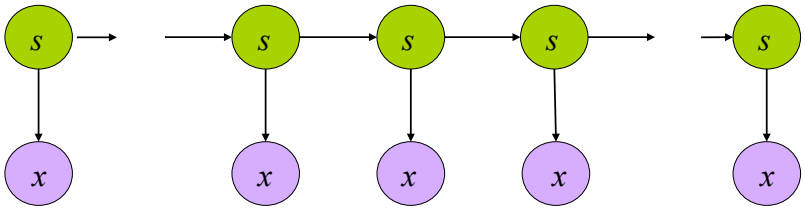
What is an HMM ?

- Example of a word model



89

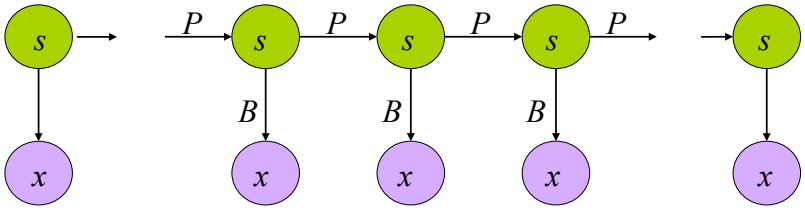
HMM formalism



- $\{s, x, \Pi, P, B\}$
- $s : \{1,\ldots, n\}$ is the random variable for the hidden states taking its values from 1 to n
- $x : \{o_1,\ldots, o_p\}$ is the random variable for the observations whose values are denoted o_i

90

HMM formalism

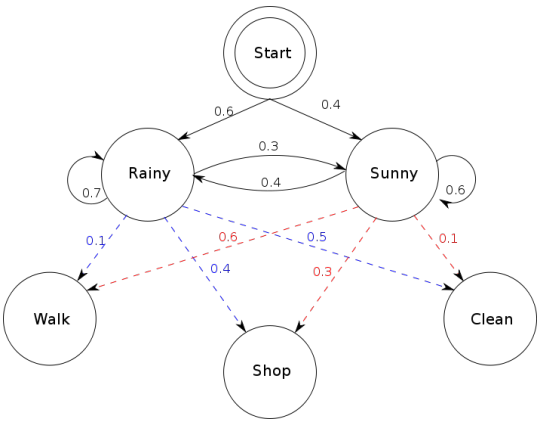


- $\{s, o, \Pi, P, B\}$
- $\Pi = \{\pi_i\}$ are the initial state probabilities, $P(s_1=i)$
- $P = \{p_{ij}\}$ are the state transition probabilities, $P(s_{t+1}=j \mid s_t=i)$
- $B = \{b_i(o_k)\}$ are the observation or emission probabilities, $P(x_t=o_k \mid s_t=i)$

91

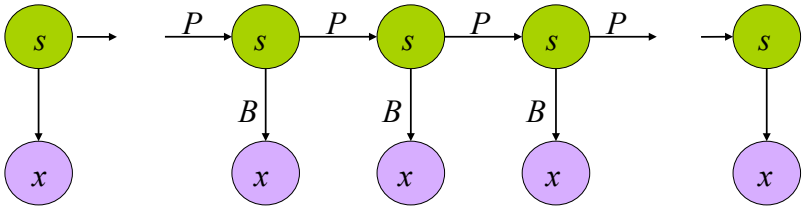
A HMM example

Taken from wikipedia



92

Inference in an HMM



- **Three basic problems**:

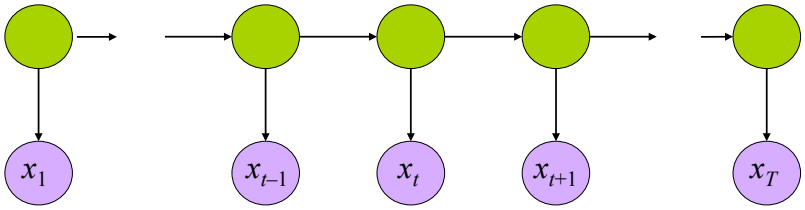
1. Compute the **probability/likelihood** of a given observation sequence (classification) ?
2. Given an observation sequence, compute the **most likely hidden state sequence** (decoding) ?
3. Given an observation sequence, which **model parameters most closely fit the data** (parameters estimation) ?

93

Likelihood computation

94

Likelihood computation



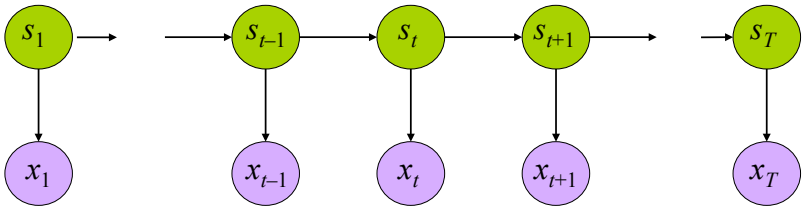
Given an observation sequence and a model, compute the **likelihood** of the observation sequence

$$\mathbf{x} = [x_1, \ldots, x_T]^{\mathrm{T}}, \ \theta = \{\Pi, P, B\}$$

Compute $\mathrm{P}(\mathbf{x} \mid \theta)$

95

Likelihood computation



$$\mathrm{P}(\mathbf{x} \mid \mathbf{s}, \theta) = b_{s_1}(x_1) b_{s_2}(x_2) \ldots b_{s_T}(x_T)$$

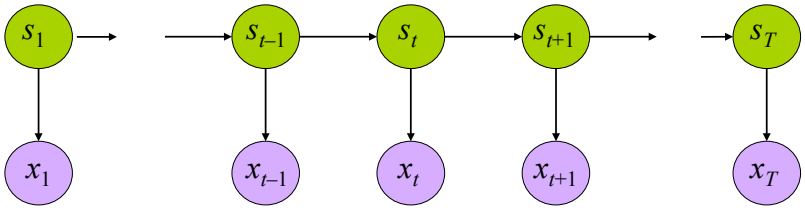
$$\mathrm{P}(\mathbf{s} \mid \theta) = \pi_{s_1} p_{s_1 s_2} p_{s_2 s_3} \ldots p_{s_{T-1} s_T}$$

$$\mathrm{P}(\mathbf{x}, \mathbf{s} \mid \theta) = \mathrm{P}(\mathbf{x} \mid \mathbf{s}, \theta) \mathrm{P}(\mathbf{s} \mid \theta)$$

$$\mathrm{P}(\mathbf{x} \mid \theta) = \sum_{\mathbf{s}} \mathrm{P}(\mathbf{x} \mid \mathbf{s}, \theta) \mathrm{P}(\mathbf{s} \mid \theta)$$

96

Likelihood computation

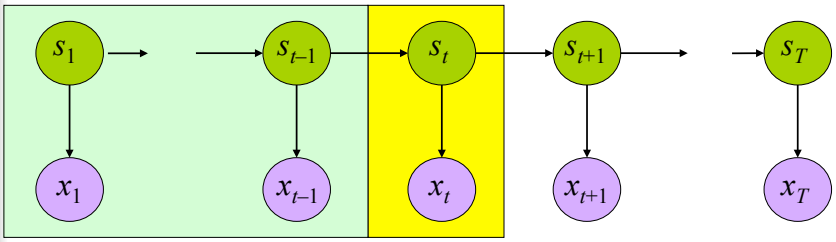


$$P(\mathbf{x} \mid \theta) = \sum_{(s_1 \ldots s_T)} \pi_{s_1} b_{s_1}(x_1) \prod_{t=1}^{T-1} p_{s_t s_{t+1}} b_{s_{t+1}}(x_{t+1})$$

It involves a sum over **all possible sequences of states** !

97

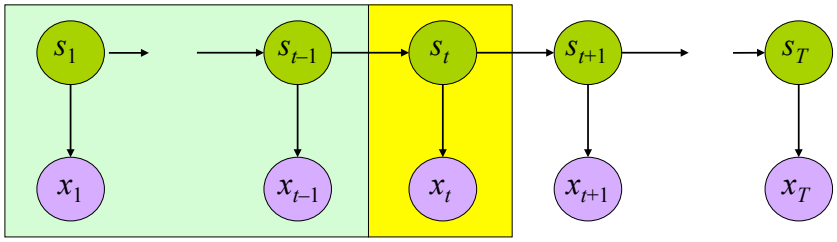
Forward procedure



- The special structure gives us an efficient solution using some **recurrence formulas**
- **Define:** $\alpha_i(t) = P(x_1 \ldots x_t, s_t = i \mid \theta)$
- We will omit the dependency on θ

98

Forward procedure



$$\alpha_j(1) = \mathrm{P}(x_1, s_1 = j)$$
$$= \mathrm{P}(x_1 \mid s_1 = j)\mathrm{P}(s_1 = j)$$
$$= b_j(x_1)\pi_j$$

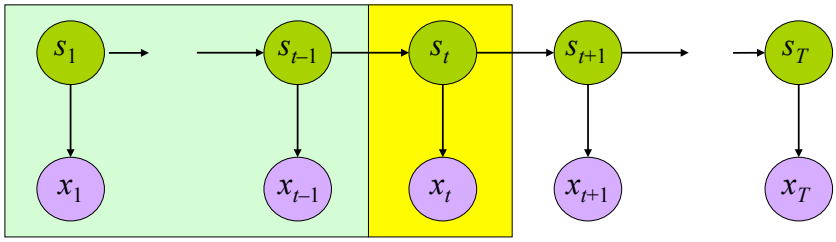
99

Forward procedure

$$\alpha_j(t+1) = \mathrm{P}(x_1 \ldots x_{t+1}, s_{t+1} = j)$$
$$= \sum_{i=1}^{n} \mathrm{P}(x_1 \ldots x_t, s_t = i, x_{t+1}, s_{t+1} = j)$$
$$= \sum_{i=1}^{n} \mathrm{P}(x_{t+1}, s_{t+1} = j \mid x_1 \ldots x_t, s_t = i)\mathrm{P}(x_1 \ldots x_t, s_t = i)$$
$$= \sum_{i=1}^{n} \mathrm{P}(x_{t+1}, s_{t+1} = j \mid s_t = i)\alpha_i(t)$$
$$= \sum_{i=1}^{n} \mathrm{P}(x_{t+1} \mid s_{t+1} = j)\mathrm{P}(s_{t+1} = j \mid s_t = i)\alpha_i(t)$$
$$= \left(\sum_{i=1}^{n} p_{ij}\alpha_i(t)\right) b_j(x_{t+1})$$

00

Forward procedure



$$\alpha_j(1) = b_j(x_1)\pi_j$$

$$\alpha_j(t+1) = \left(\sum_{i=1}^{n} p_{ij}\alpha_i(t)\right) b_j(x_{t+1})$$

101

Forward procedure

From Rabiner et al.:

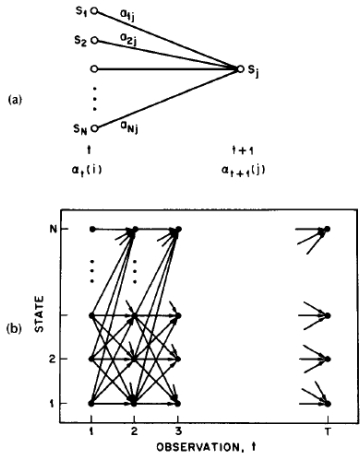
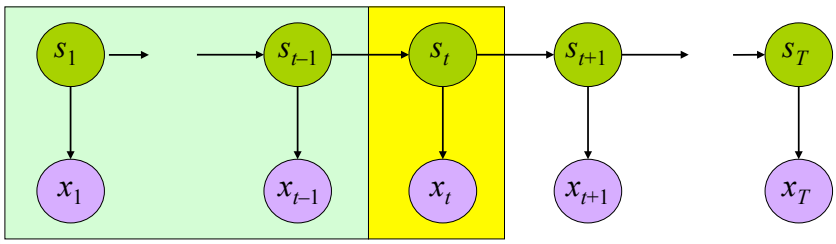


Fig. 4. (a) Illustration of the sequence of operations required for the computation of the forward variable $\alpha_{t+1}(j)$. (b) Implementation of the computation of $\alpha_t(i)$ in terms of a lattice of observations t, and states i.

102

Likelihood computation: solutions

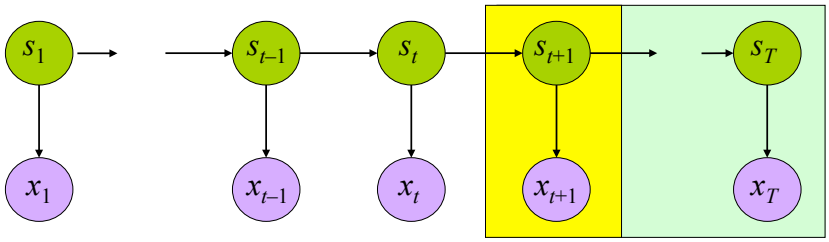


The likelihood is:

$$P(\mathbf{x} \mid \theta) = \sum_{j=1}^{n} P(x_1 \ldots x_T, s_T = j \mid \theta) = \sum_{j=1}^{n} \alpha_j(T)$$

103

Backward procedure



- We now introduce the **backward variable**
- **Define:** $\beta_i(t) = P(x_{t+1} \ldots x_T \mid s_t = i, \theta)$
- We will omit the dependency on θ

104

Backward procedure

$$\beta_i(t) = \mathrm{P}(x_{t+1}...x_T \mid s_t = i)$$

$$= \sum_{j=1}^{n} \mathrm{P}(x_{t+1}x_{t+2}...x_T, s_{t+1} = j \mid s_t = i)$$

$$= \sum_{j=1}^{n} \mathrm{P}(x_{t+1}x_{t+2}...x_T \mid s_t = i, s_{t+1} = j)\mathrm{P}(s_{t+1} = j \mid s_t = i)$$

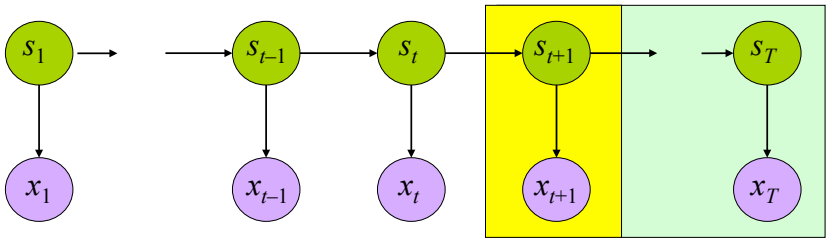
$$= \sum_{j=1}^{n} p_{ij}\,\mathrm{P}(x_{t+1}x_{t+2}...x_T \mid s_{t+1} = j)$$

$$= \sum_{j=1}^{n} p_{ij}\,\mathrm{P}(x_{t+1} \mid x_{t+2}...x_T, s_{t+1} = j)\mathrm{P}(x_{t+2}...x_T \mid s_{t+1} = j)$$

$$= \sum_{j=1}^{n} p_{ij} b_j(x_{t+1})\beta_j(t+1)$$

105

Backward procedure



- When $t = T - 1$, we have

$$\beta_i(T-1) = \mathrm{P}(x_T \mid s_{T-1} = i)$$

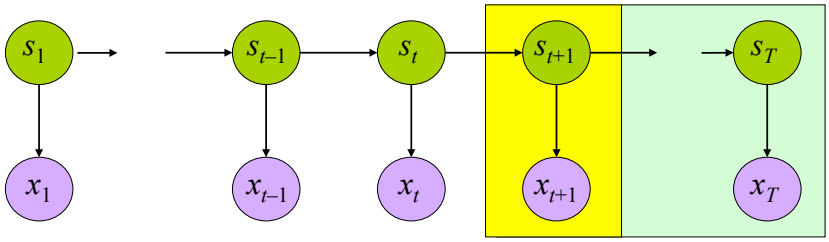
$$= \sum_{j=1}^{n} p_{ij} b_j(x_T)$$

$$\beta_i(T-1) = \sum_{j=1}^{n} p_{ij} b_j(x_T)\beta_j(T)$$

$$\Rightarrow \beta_j(T) = 1$$

106

Backward procedure



$$\beta_i(T) = 1$$

$$\beta_i(t) = \sum_{j=1}^{n} p_{ij} b_j(x_{t+1}) \beta_j(t+1)$$

107

Backward procedure

From Rabiner et al.:

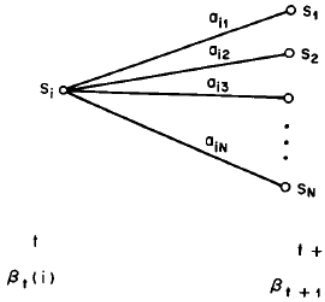
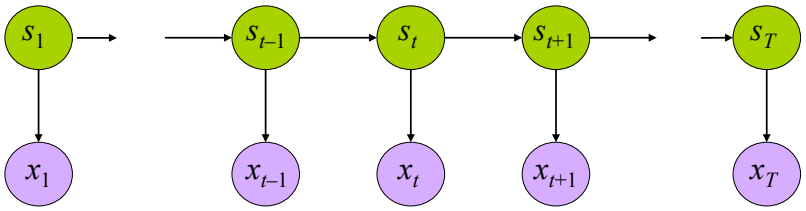


Fig. 5. Illustration of the sequence of operations required for the computation of the backward variable $\beta_t(i)$.

)8

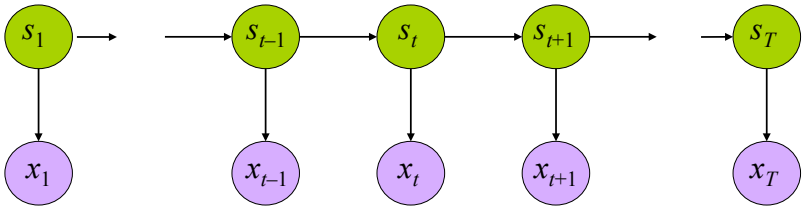
Likelihood computation: solutions



$$\begin{aligned}
P(\mathbf{x}, s_t = i \mid \theta) &= P(x_1 \ldots x_t, s_t = i, x_{t+1} \ldots x_T \mid \theta) \\
&= P(x_{t+1} \ldots x_T \mid x_1 \ldots x_t, s_t = i, \theta) P(x_1 \ldots x_t, s_t = i \mid \theta) \\
&= P(x_{t+1} \ldots x_T \mid s_t = i, \theta) P(x_1 \ldots x_t, s_t = i \mid \theta) \\
&= \alpha_i(t) \beta_i(t)
\end{aligned}$$

109

Likelihood computation: solutions



$$P(\mathbf{x} \mid \theta) = \sum_{i=1}^{n} \alpha_i(T)$$ **Forward procedure**

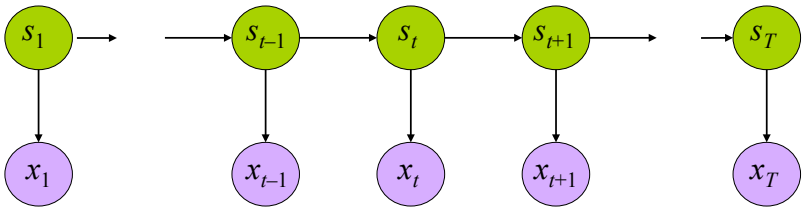
$$P(\mathbf{x} \mid \theta) = \sum_{i=1}^{n} \pi_i \beta_i(1)$$ **Backward procedure**

$$P(\mathbf{x} \mid \theta) = \sum_{i=1}^{n} \alpha_i(t) \beta_i(t)$$ **Combination**

Optimal state sequence (decoding)

111

Best state

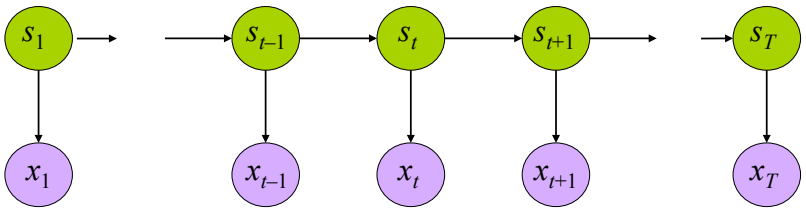


- Find the **most probable state** at time t given the observations

$$\gamma_i(t) = \mathrm{P}(s_t = i \mid \mathbf{x}, \theta) = \frac{\mathrm{P}(\mathbf{x}, s_t = i \mid \theta)}{\mathrm{P}(\mathbf{x} \mid \theta)}$$
$$= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{n} \alpha_j(t)\beta_j(t)}$$

112

Best state sequence

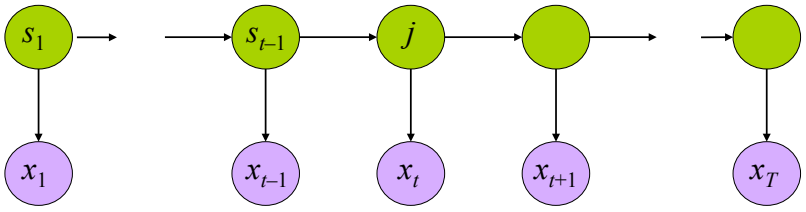


- Find the state sequence that best explains the observations
- Viterbi algorithm = dynamic programming algorithm

$$\arg\max_{\mathbf{s}} P(\mathbf{s} \mid \mathbf{x}, \theta) = \arg\max_{\mathbf{s}} P(\mathbf{s}, \mathbf{x} \mid \theta)$$

113

Viterbi algorithm

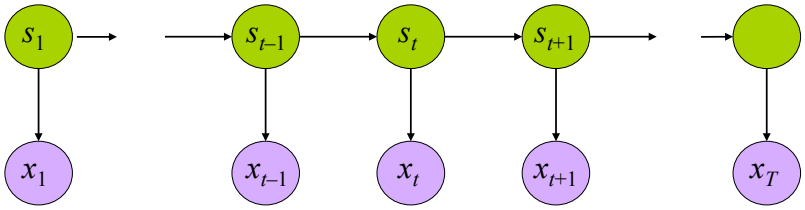


$$\delta_j(t) = \max_{s_1 \ldots s_{t-1}} P(s_1 \ldots s_{t-1}, x_1 \ldots x_{t-1}, s_t = j, x_t \mid \theta)$$

The state sequence which maximizes the probability of generating the observations up to time t–1, landing in state j, and emitting the observation at time t

114

Viterbi algorithm



$$\delta_j(t) = \max_{s_1 \ldots s_{t-1}} \mathrm{P}(s_1 \ldots s_{t-1}, x_1 \ldots x_{t-1}, s_t = j, x_t \mid \theta)$$

$$\delta_j(1) = \pi_j b_j(x_1)$$
$$\delta_j(t+1) = \max_i \left\{ \delta_i(t) p_{ij} b_j(x_{t+1}) \right\}$$

$$\psi_j(1) = 0$$
$$\psi_j(t+1) = \arg\max_i \left\{ \delta_i(t) p_{ij} b_j(x_{t+1}) \right\}$$

Recursive computation

115

Viterbi algorithm

- Indeed, $\delta_j(t+1)$ is equal to

$$\begin{aligned}
\max_{s_1 \ldots s_t} \mathrm{P}(s_1 \ldots s_{t+1} = j, x_1 \ldots x_{t+1} \mid \theta) &= \max_{s_1 \ldots s_t} \mathrm{P}(s_{t+1} = j, x_{t+1} \mid s_1 \ldots s_t, x_1 \ldots x_t \theta) \mathrm{P}(s_1 \ldots s_t, x_1 \ldots x_t \mid \theta) \\
&= \max_{s_1 \ldots s_t} \left\{ \mathrm{P}(x_{t+1} \mid s_{t+1} = j, \theta) \mathrm{P}(s_{t+1} = j \mid s_t, \theta) \mathrm{P}(s_1 \ldots s_t, x_1 \ldots x_t \mid \theta) \right\} \\
&= \max_{s_t} \max_{s_1 \ldots s_{t-1}} \left\{ b_{s_{t+1}}(x_{t+1}) p_{s_t s_{t+1}} \mathrm{P}(s_1 \ldots s_t, x_1 \ldots x_t \mid \theta) \right\} \\
&= \max_{s_t} \left\{ b_j(x_{t+1}) p_{s_t j} \max_{s_1 \ldots s_{t-1}} \mathrm{P}(s_1 \ldots s_t, x_1 \ldots x_t \mid \theta) \right\} \\
&= \max_{s_t} \left\{ b_j(x_{t+1}) p_{s_t j} \delta_{s_t}(t) \right\}
\end{aligned}$$

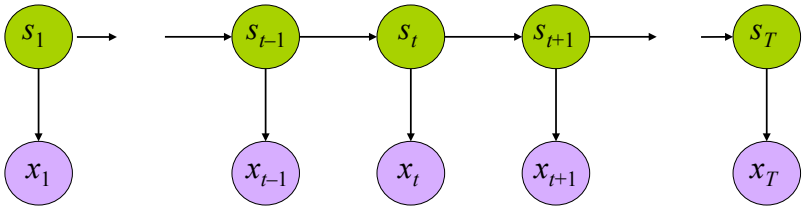
116

Viterbi algorithm

- And we can apply dynamic programming to the log-likelihood
  - The cost is then additive

117

Viterbi algorithm



$$s^*(T) = \arg\max_i \delta_i(T)$$

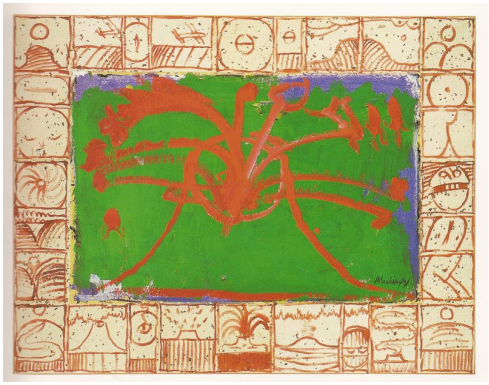
$$s^*(t-1) = \psi_{s^*(t)}(t)$$

$$\mathrm{P}(\mathbf{x}^*) = \max_i \delta_i(T)$$

Compute the most likely state sequence by working backwards

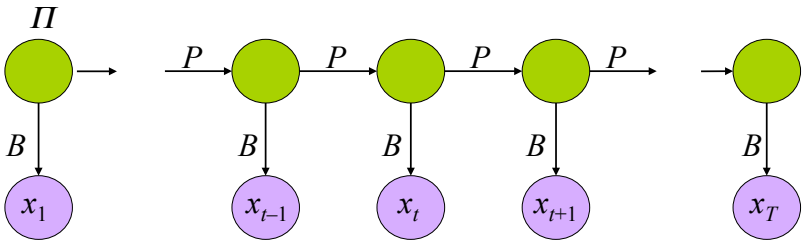
118

Parameter estimation



119

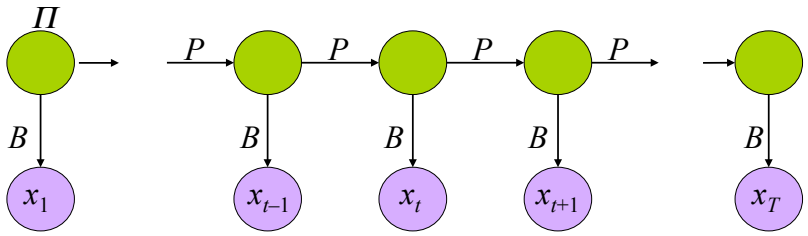
Parameter estimation



- Given an observation sequence, find the model parameters Π, P, B, that most likely produce that sequence (maximum likelihood)
- No closed-form solution
- Instance of the iterative EM algorithm

120

Parameter estimation



$$\gamma_{ij}(t) = \mathrm{P}(s_t = i, s_{t+1} = j \mid \mathbf{x}, \theta)$$

Probability of traversing an arc

$$\gamma_i(t) = \mathrm{P}(s_t = i \mid \mathbf{x}, \theta)$$

Probability of being in state *i*

Parameter estimation

Recall that we already computed

$$\gamma_i(t) = \mathrm{P}(s_t = i \mid \mathbf{x}, \theta) = \frac{\mathrm{P}(\mathbf{x}, s_t = i \mid \theta)}{\mathrm{P}(\mathbf{x} \mid \theta)} = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{n} \alpha_j(t)\beta_j(t)}$$

122

Parameter estimation

$$\gamma_{ij}(t) = \mathrm{P}(s_t = i, s_{t+1} = j \mid \mathbf{x}, \theta) = \frac{\mathrm{P}(x_1 \ldots x_t, s_t = i, s_{t+1} = j, x_{t+1} \ldots x_T \mid \theta)}{\mathrm{P}(\mathbf{x} \mid \theta)}$$

$$= \frac{\mathrm{P}(s_{t+1} = j, x_{t+1} \ldots x_T \mid x_1 \ldots x_t, s_t = i, \theta)\mathrm{P}(x_1 \ldots x_t, s_t = i \mid \theta)}{\mathrm{P}(\mathbf{x} \mid \theta)}$$

$$= \frac{\mathrm{P}(s_{t+1} = j, x_{t+1} \ldots x_T \mid s_t = i, \theta)\alpha_i(t)}{\mathrm{P}(\mathbf{x} \mid \theta)}$$

$$= \frac{\mathrm{P}(x_{t+1} \ldots x_T \mid s_t = i, s_{t+1} = j, \theta)\mathrm{P}(s_{t+1} = j \mid s_t = i, \theta)\alpha_i(t)}{\mathrm{P}(\mathbf{x} \mid \theta)}$$

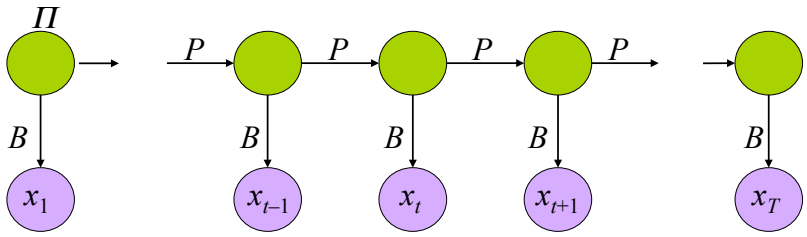
$$= \frac{\mathrm{P}(x_{t+1} \ldots x_T \mid s_{t+1} = j, \theta) p_{ij} \alpha_i(t)}{\mathrm{P}(\mathbf{x} \mid \theta)}$$

$$= \frac{\mathrm{P}(x_{t+1} \mid x_{t+2} \ldots x_T, s_{t+1} = j, \theta)\mathrm{P}(x_{t+2} \ldots x_T \mid s_{t+1} = j, \theta) p_{ij} \alpha_i(t)}{\mathrm{P}(\mathbf{x} \mid \theta)}$$

$$= \frac{\alpha_i(t) b_j(x_{t+1}) p_{ij} \beta_j(t+1)}{\sum_{k=1}^{N} \alpha_k(t)\beta_k(t)}$$

123

Parameter estimation



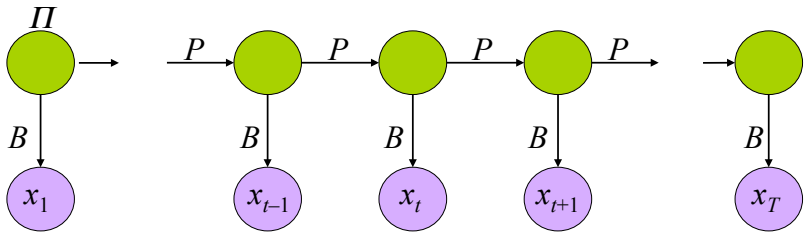
$\hat{\pi}_i$ = probability of starting from i

$$\hat{\pi}_i = \mathrm{P}(s_1 = i \mid \mathbf{x}, \hat{\theta}) = \gamma_i(1)$$

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

124

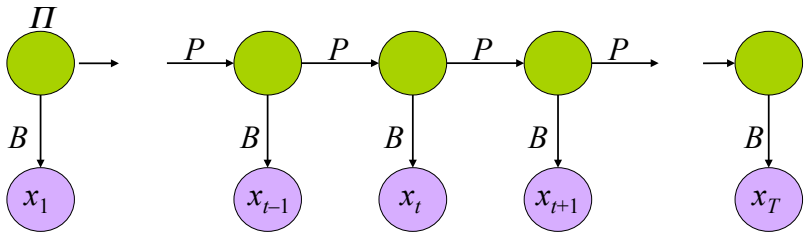
Parameter estimation



$$\hat{p}_{ij} = \frac{\text{expected number of transitions from } i \text{ to } j}{\text{expected number of transitions out of state } i}$$

125

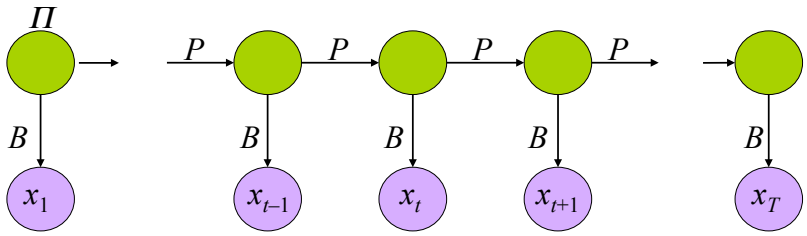
Parameter estimation



$$\hat{p}_{ij} = \frac{\sum_{t=1}^{T-1} \mathrm{P}(s_t = i, s_{t+1} = j \mid \mathbf{x}, \hat{\theta})}{\sum_{t=1}^{T-1} \mathrm{P}(s_t = i \mid \mathbf{x}, \hat{\theta})} = \frac{\sum_{t=1}^{T-1} \gamma_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

126

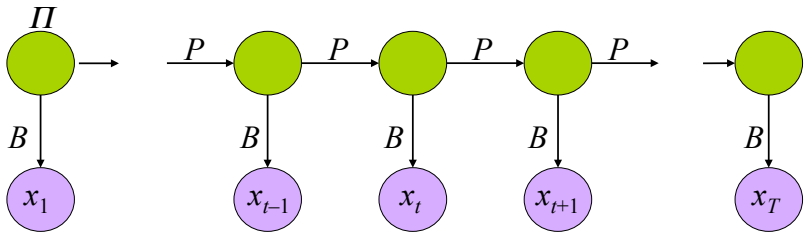
Parameter estimation



$$\hat{b}_i(o_k) = \frac{\text{expected number of emissions of } o_k \text{ in state } i}{\text{total number of emissions in state } i}$$

127

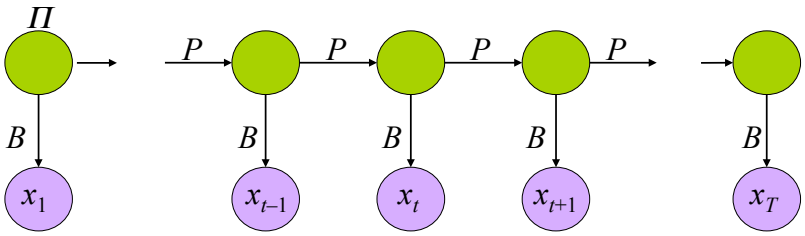
Parameter estimation



$$\hat{b}_i(o_k) = \frac{\sum_{t=1}^{T} P(s_t = i, x_t = o_k \mid \mathbf{x}, \hat{\theta})}{\sum_{t=1}^{T} P(s_t = i \mid \mathbf{x}, \hat{\theta})} = \frac{\sum_{t=1}^{T} P(s_t = i \mid \mathbf{x}, \hat{\theta}) \delta(x_t = o_k)}{\sum_{t=1}^{T} P(s_t = i \mid \mathbf{x}, \hat{\theta})}$$

$$= \frac{\sum_{\{t: x_t = o_k\}} P(s_t = i \mid \mathbf{x}, \hat{\theta})}{\sum_{t=1}^{T} P(s_t = i \mid \mathbf{x}, \hat{\theta})} = \frac{\sum_{\{t: x_t = o_k\}} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(t)}$$

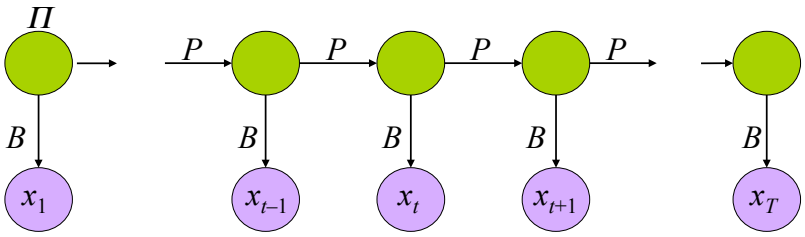
Parameter estimation



- The two following two steps are iterated until convergence:
  - Recompute the forward and backward variables α and β
  - Recompute the parameter estimates for the Π, P, B

129

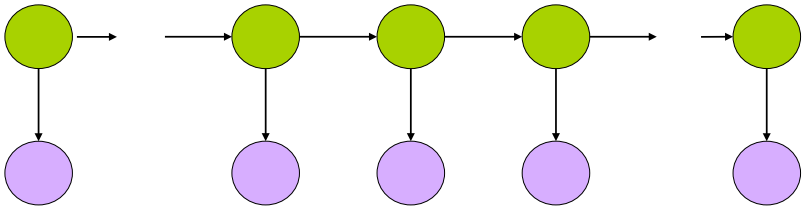
Parameter estimation



- It can be shown that this iterative algorithm increases the likelihood

130

HMM applications



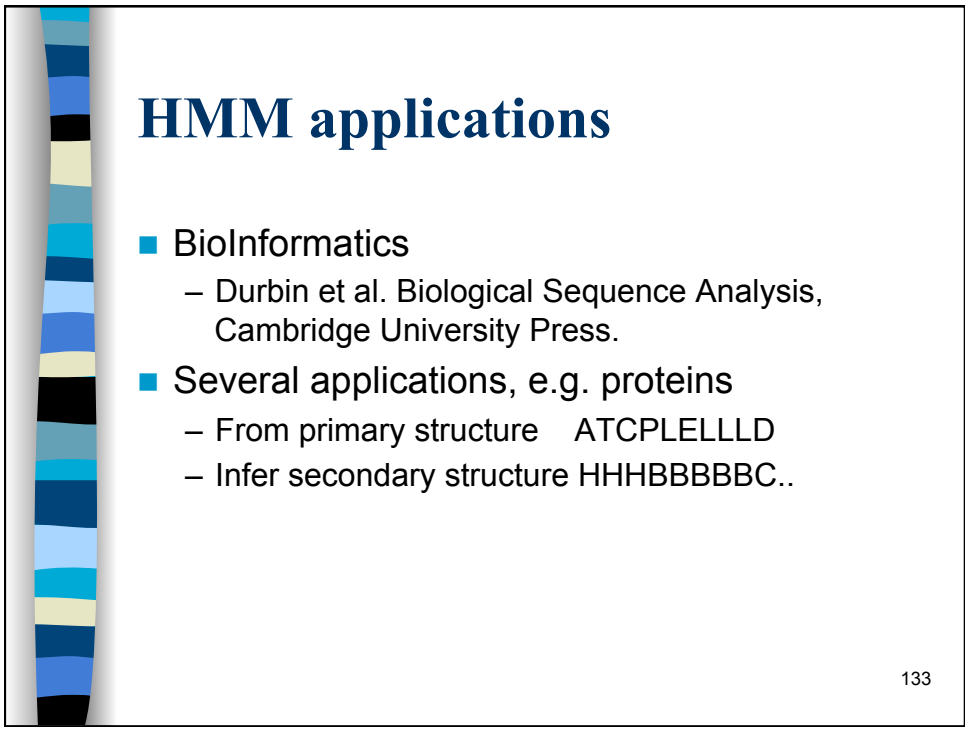
- Generating parameters for n-gram models
- Tagging speech
- Speech recognition
- Bioinformatics sequence modeling

131

HMM applications

- Part-of-speech tagging
  - The representative put chairs on the table
  - AT NN VBD NNS IN AT NN
  - AT JJ NN VBZ IN AT NN
- Some tags :
  - AT: article, NN: singular or mass noun, VBD: verb, past tense, NNS: plural noun, IN: preposition, JJ: adjective

132


HMM applications
BioInformatics
– Durbin et al. Biological Sequence Analysis, Cambridge University Press.
Several applications, e.g. proteins
– From primary structure ATCPLELLLD
– Infer secondary structure HHHBBBBBC..
133