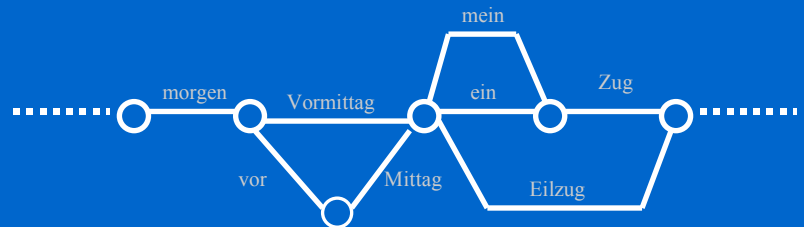
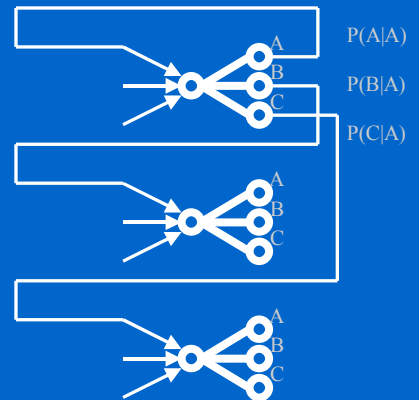
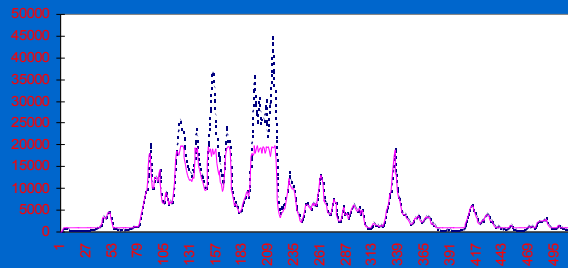
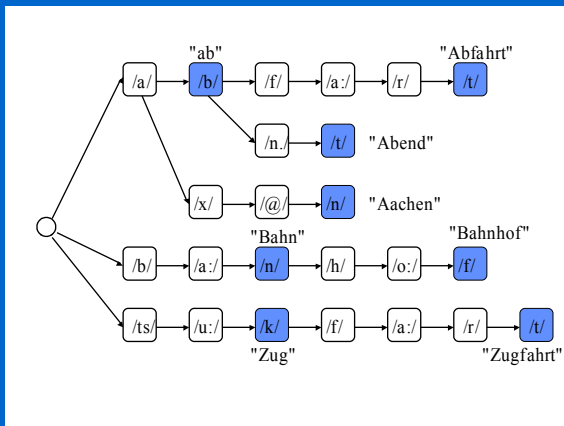
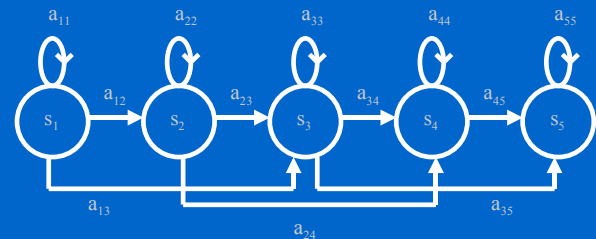
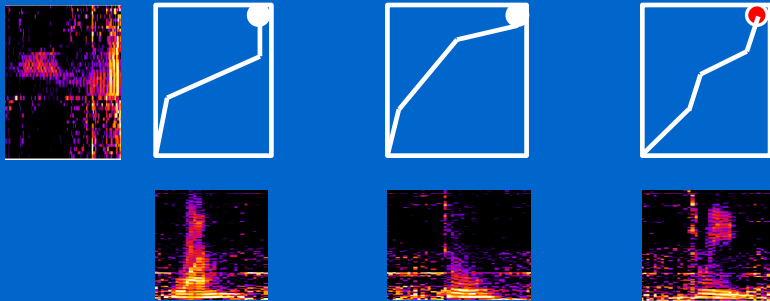


Automatic Speech Recognition

Dr.-Ing. Bernd Plannerer

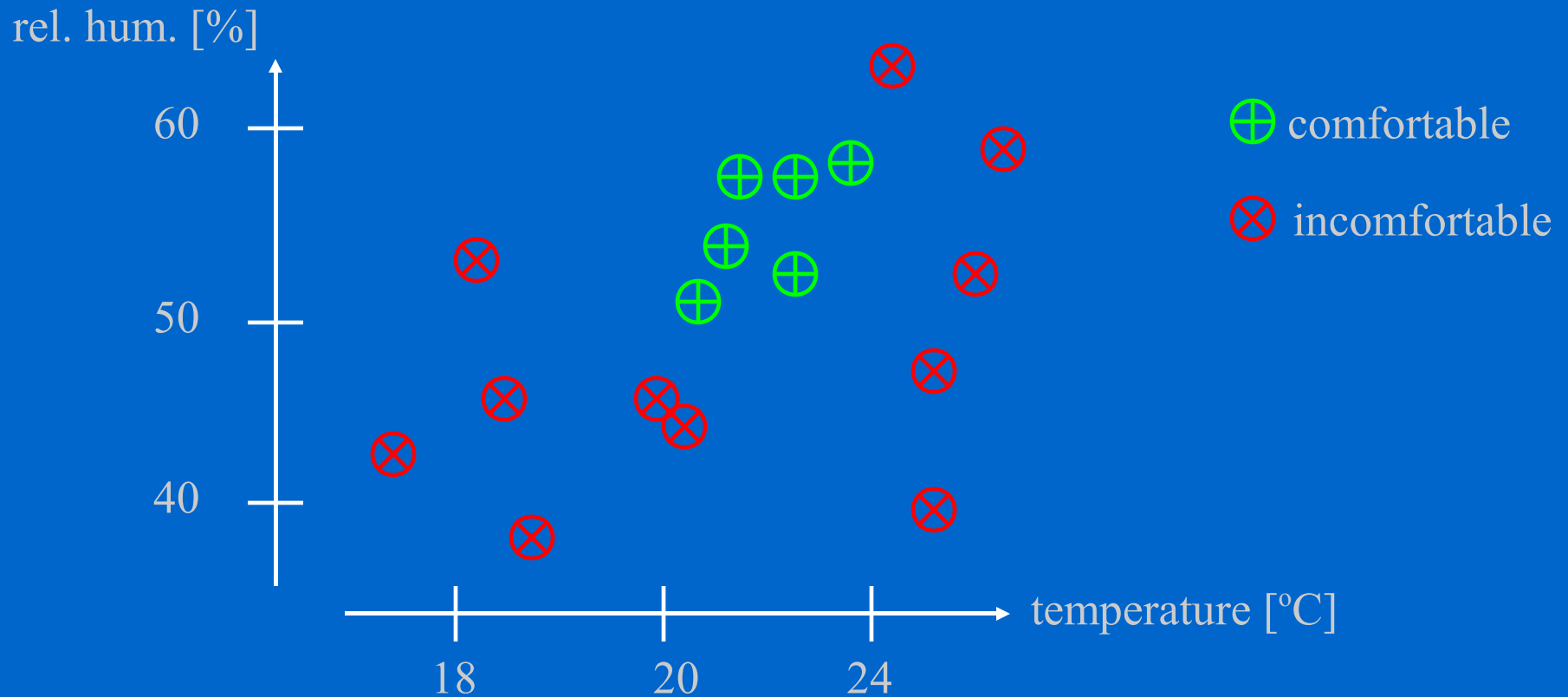
plannerer@ieee.org

CHAPTER 2



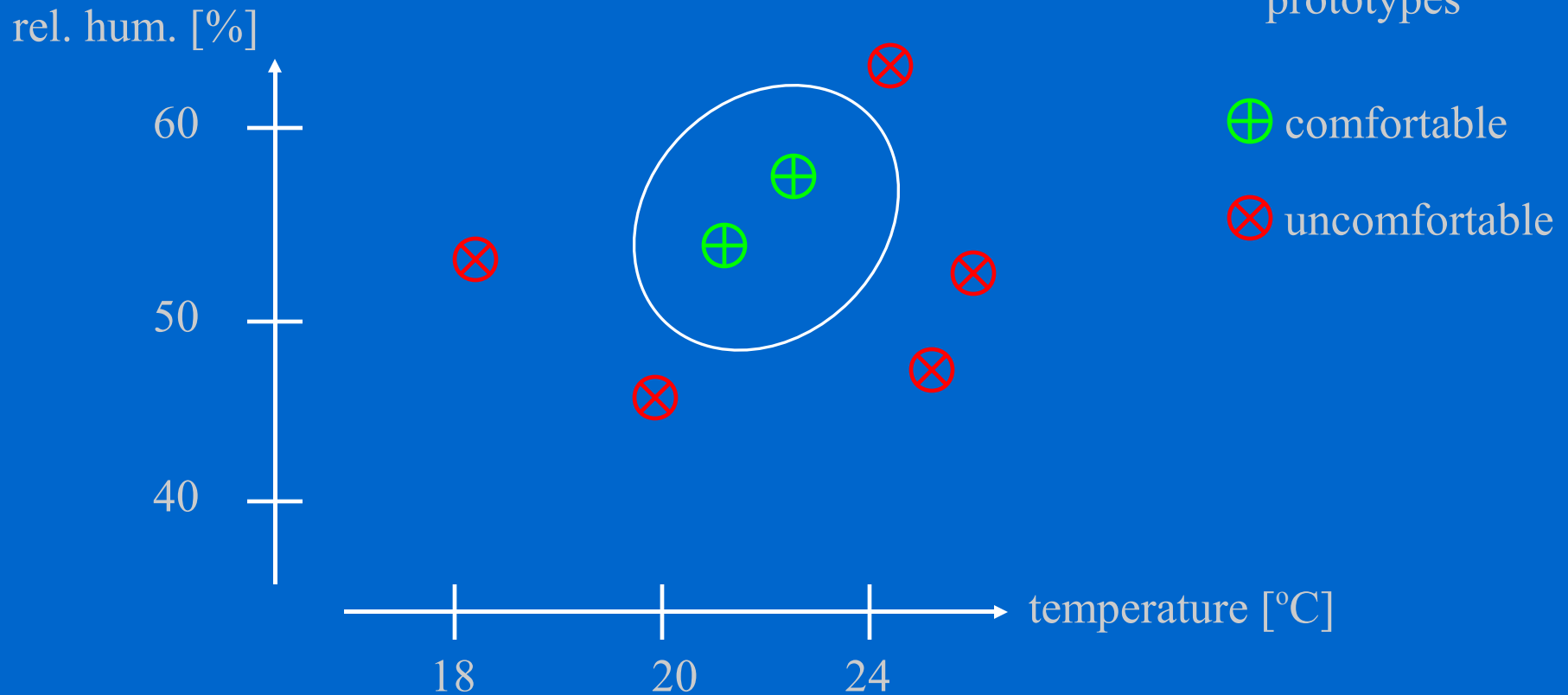
Feature Vectors

Feature space: Temperature and relative humidity

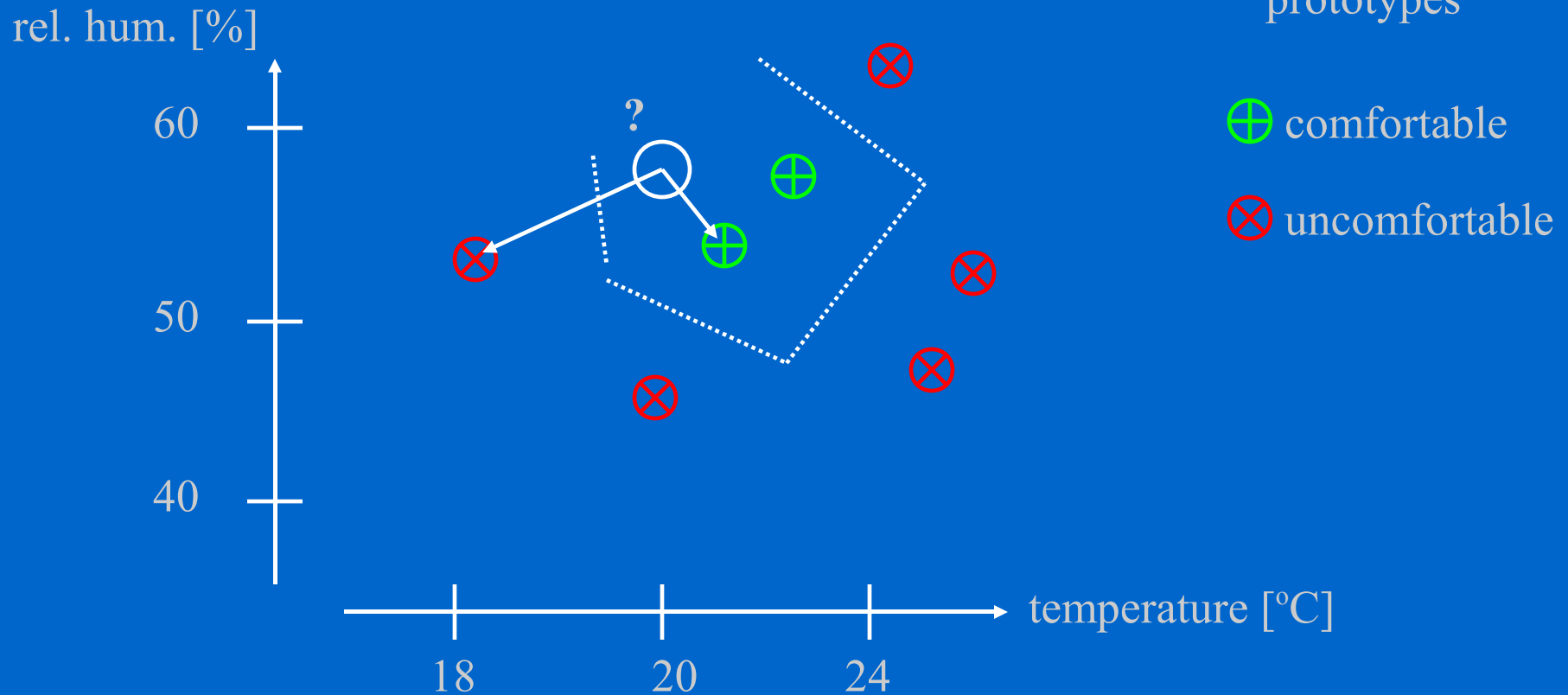


Distance Measures

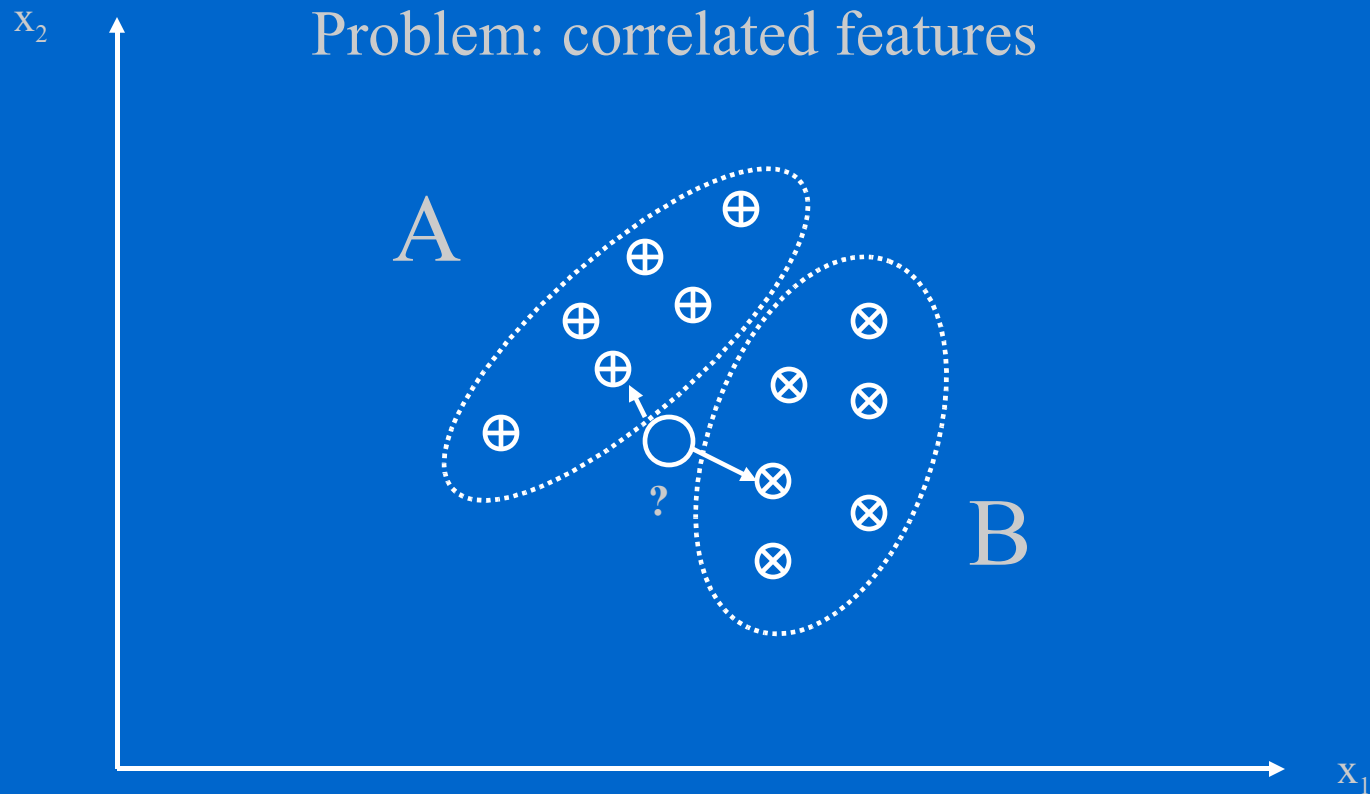
Class prototypes



Distance Measures



Distance Measures



Distance Measures

- Solution: Mahalanobis Distance
 - covariance matrix \mathbf{C}

$$d_{Mahalanobis}(\vec{x}, \vec{p}, \tilde{\mathbf{C}}) = (\vec{x} - \vec{p})' \cdot \tilde{\mathbf{C}}^{-1} \cdot (\vec{x} - \vec{p})$$

- each class has its own covariance matrix

Distance Measures

- Covariance Matrix

$$\tilde{C} = \frac{1}{N-1} \sum_{n=0}^{N-1} (\vec{x}_n - \vec{m}) \cdot (\vec{x}_n - \vec{m})' \quad (3.9)$$

where \vec{m} is the mean vector of the training set:

$$\vec{m} = \frac{1}{N} \sum_{n=0}^{N-1} \vec{x}_n \quad (3.10)$$

looking at a single element of the matrix \tilde{C} , say, $\sigma_{i,j}$, this can also be written as:

$$\sigma_{i,j} = \frac{1}{N-1} \sum_{n=0}^{N-1} (x_{i,n} - m_i) \cdot (x_{j,n} - m_j) \quad (3.11)$$



Distance Measures

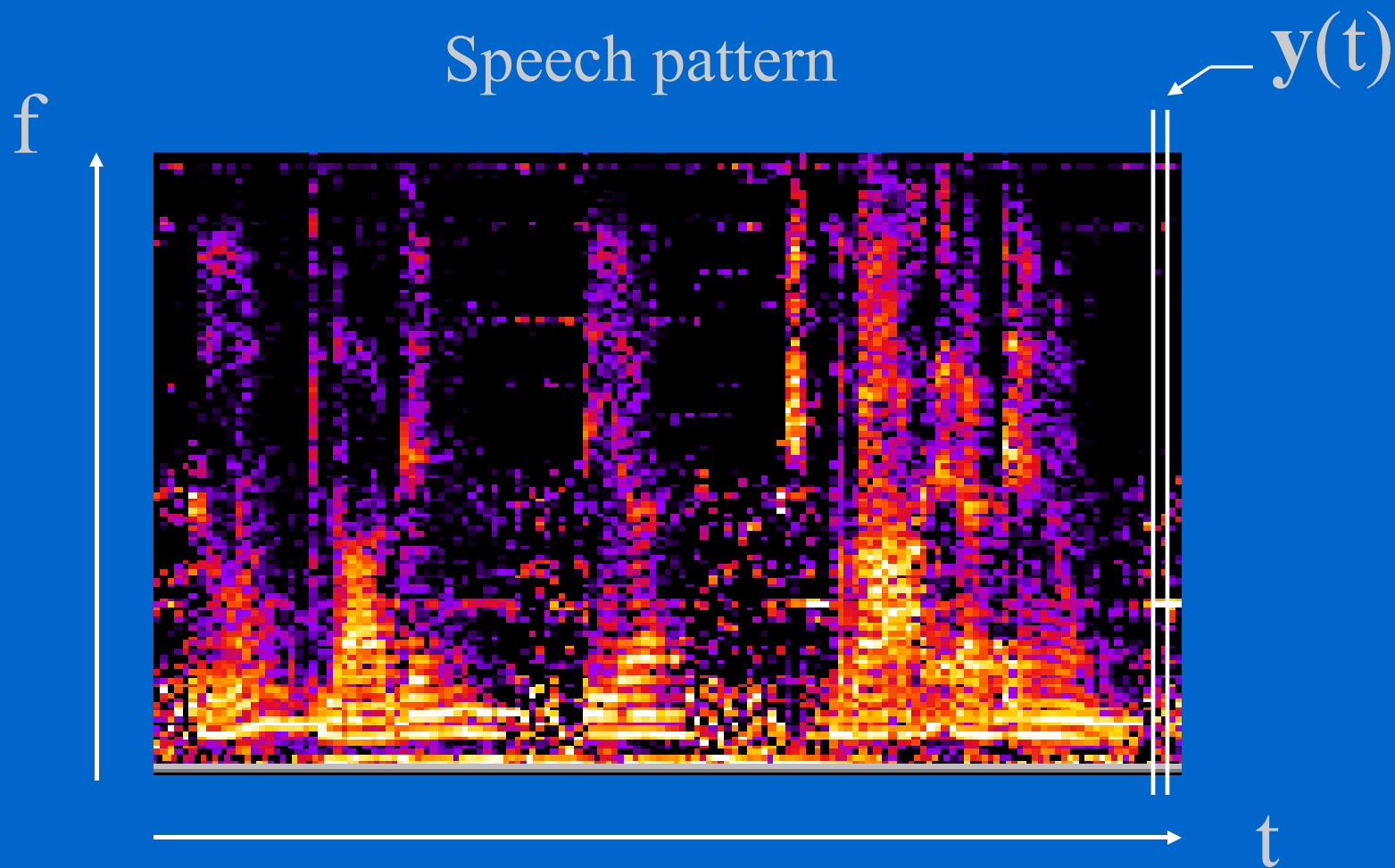
- Classification using the Mahalanobis Distance

$$d_{\omega_v}(\vec{x}) = \min_k \left\{ d_{Mahalanobis} \left(\vec{x}, \vec{p}_{k, \omega_v}, \tilde{C}_{k, \omega_v} \right) \right\}; k = 0, 1, \dots, (K_{\omega_v} - 1) \quad (3.12)$$



-
-
-

Pattern Matching and Dynamid Programming



Pattern Matching and Dynamic Programming

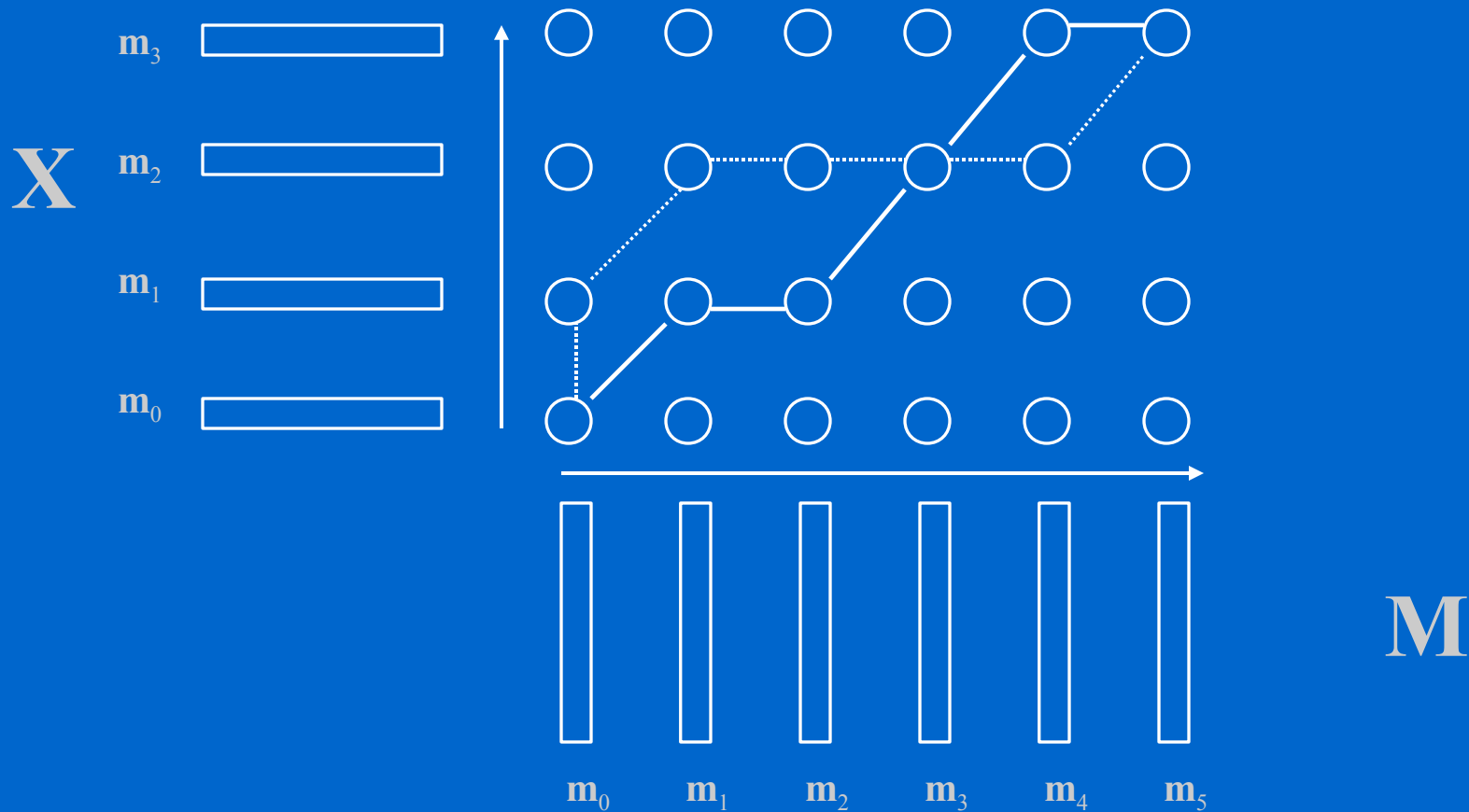
- So far, individual feature vectors have been classified.
 - Speech patterns consist of *sequences* of feature vectors
 - The length of the sequences is different from their prototypes
- $\nabla \rightarrow$ Time alignment is necessary

Pattern Matching and Dynamic Programming

- Dynamic programming
 - define optimal alignment (time warping) of feature vectors
 - compute the distance for the given alignment of vectors (= path)
 - Pattern Matching via DP is also known as DTW (Dynamic Time Warping)

Pattern Matching and Dynamic Programming

Distance measure for vector sequences

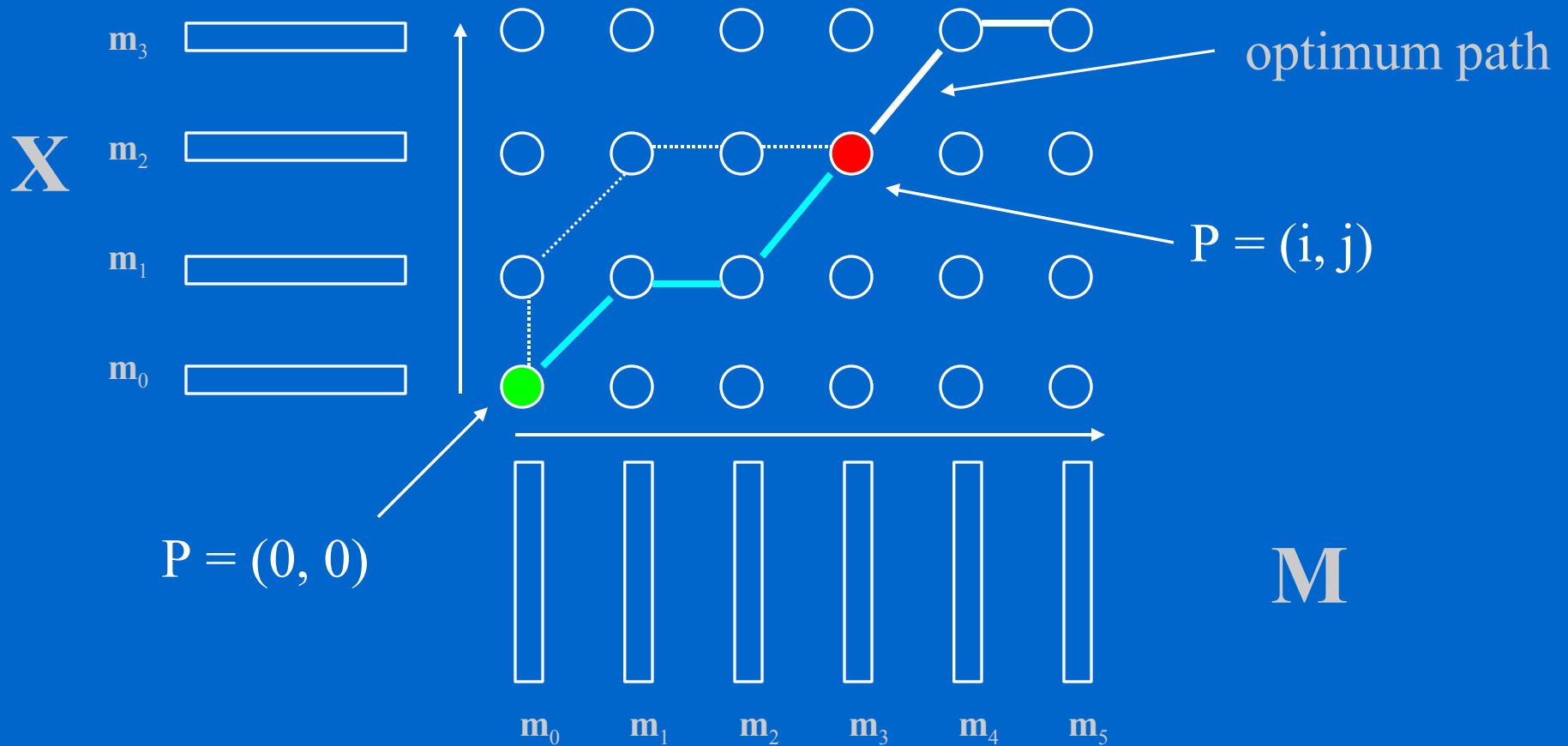


Pattern Matching and Dynamic Programming

- Bellman's Principle:
„If a point $P = (i, j)$ is part of the optimum path, then the optimum path from the start point $(0,0)$ to point P is also part of the optimum path“
- Additive costs: Breaking up the path at the last edge leads to
$$L\{P_0, P_1, \dots, P_{n-1}, P_n\} = L\{P_0, P_1, \dots, P_{n-1}\} + L\{P_{n-1}, P_n\}$$

Pattern Matching and Dynamic Programming

Bellman's principle





Pattern Matching and Dynamic Programming

- If $\{P_0 \dots P_n\}$ is the optimum path to P_n , then the partial path $\{P_0 \dots P_{n-1}\}$ leading to the predecessor P_{n-1} is the optimum path from P_0 to P_{n-1}
- From this, we can iteratively construct the optimal path:

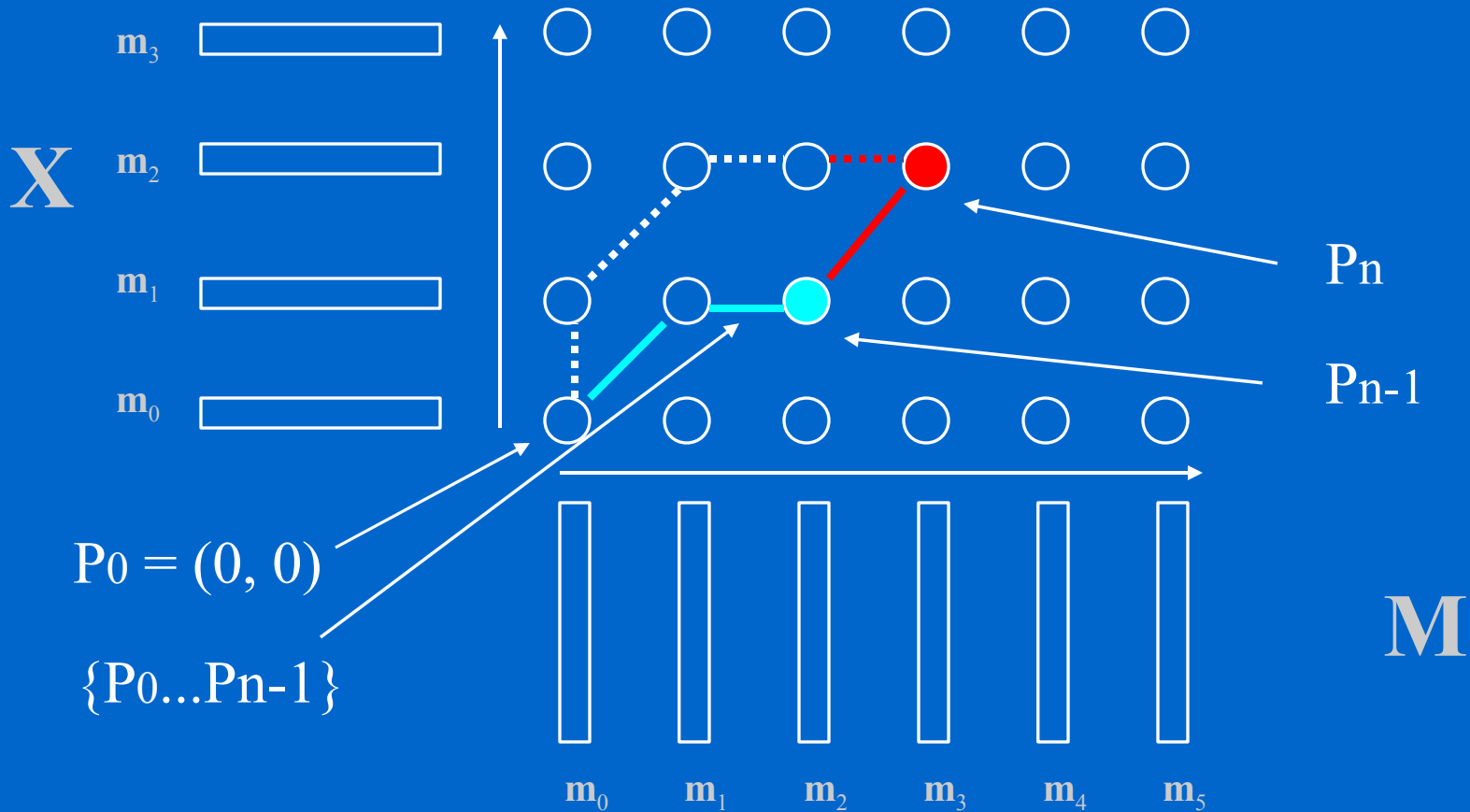


Pattern Matching and Dynamic Programming

- If we know the optimum paths to all possible predecessors of P_n , we can find the optimal path to P_n
- In order to find the optimum path to P_n , the path to point P_n must contain the best of all partial paths to the possible predecessors of P_n .

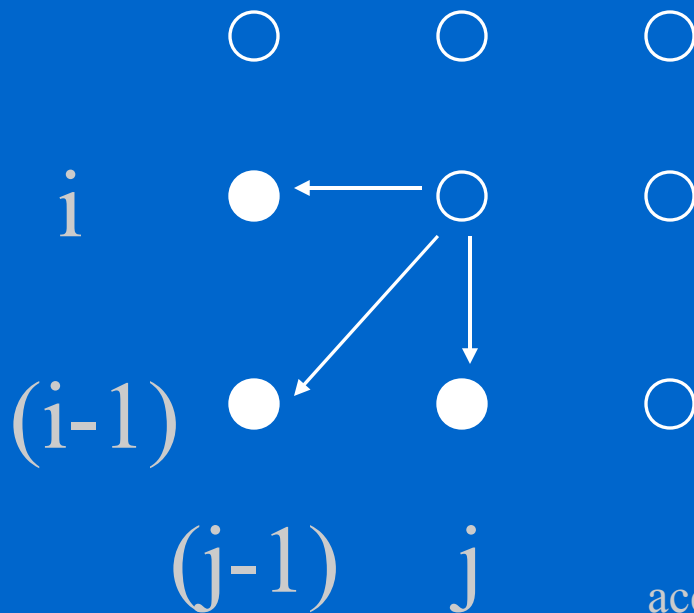
Pattern Matching and Dynamic Programming

Bellman's principle



Pattern Matching and Dynamic Programming

local path diagram



decision rule

$$D_{ij} = d_{ij} + \min \left\{ \begin{array}{l} a_0 + D_{(i-1)j} \\ a_1 + D_{i(j-1)} \\ a_2 + D_{(i-1)(j-1)} \end{array} \right\}$$

accumulated distance
up to point (i,j)

local distance (x_i, m_j)

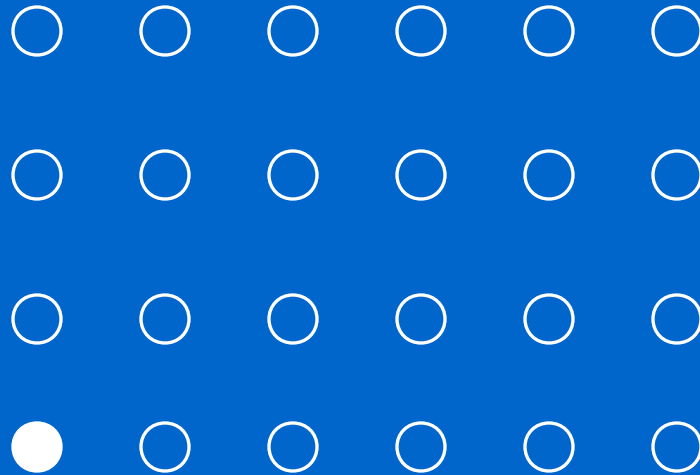
accumulated distance
to predecessor

„costs of warping“

-
-
-

Pattern Matching and Dynamic Programming

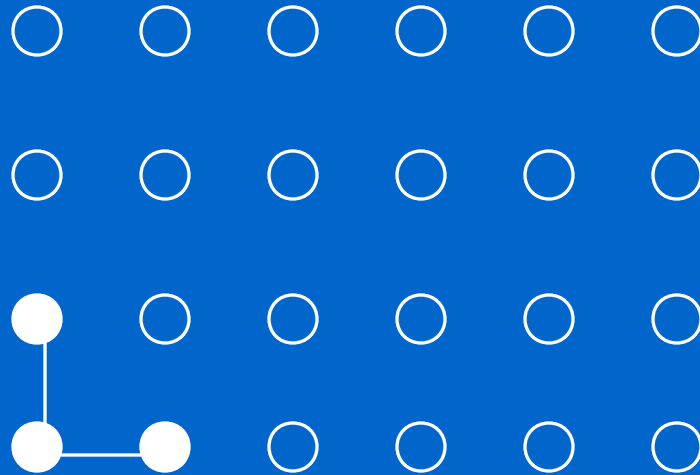
step 0



-
-
-

Pattern Matching and Dynamic Programming

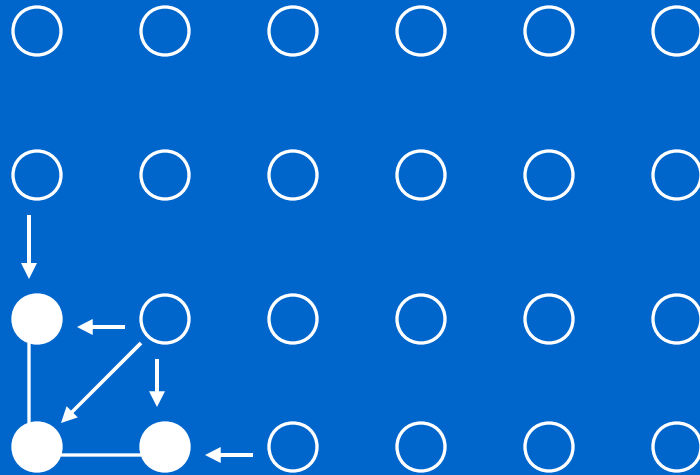
step 1b



-
-
-

Pattern Matching and Dynamic Programming

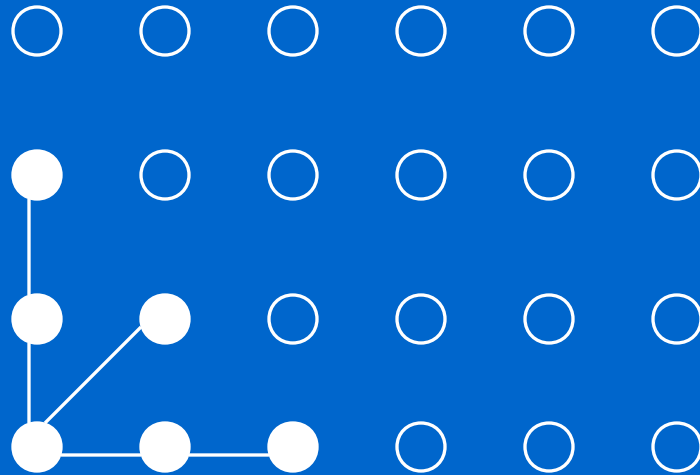
step 2a



-
-
-

Pattern Matching and Dynamic Programming

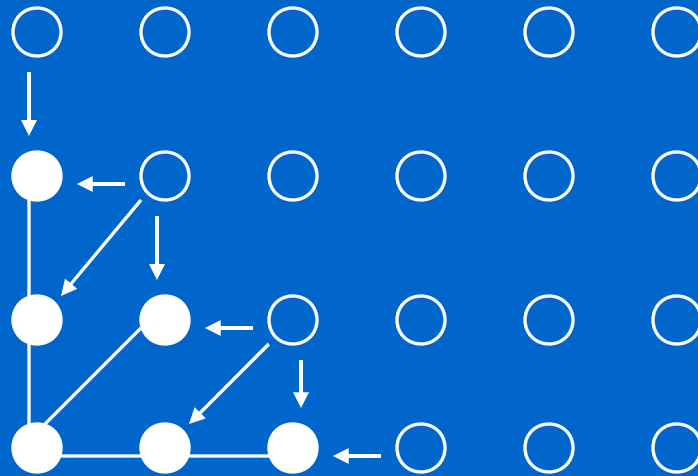
step 2b



-
-
-

Pattern Matching and Dynamic Programming

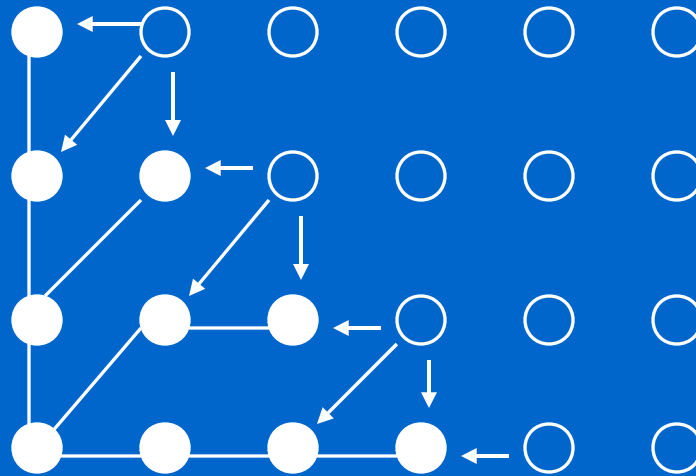
step 3a



-
-
-

Pattern Matching and Dynamic Programming

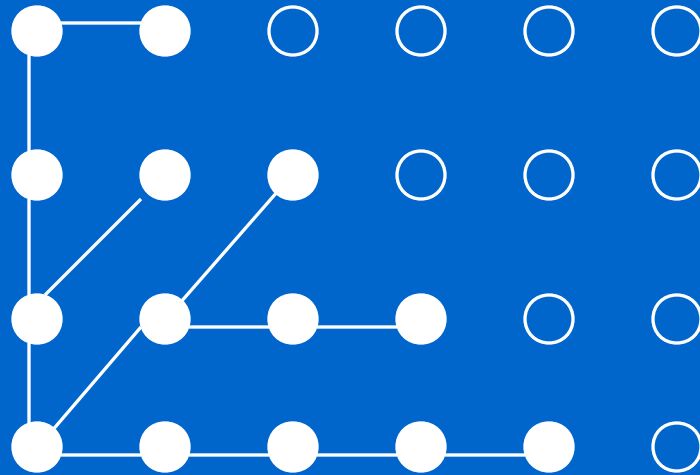
step 4a



-
-
-

Pattern Matching and Dynamic Programming

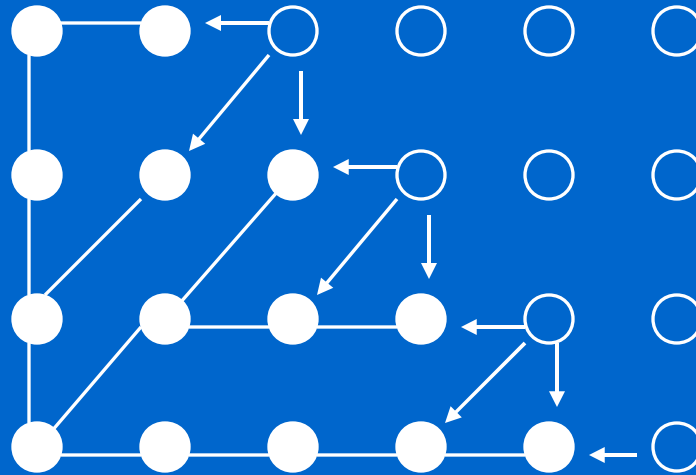
step 4b



-
-
-

Pattern Matching and Dynamic Programming

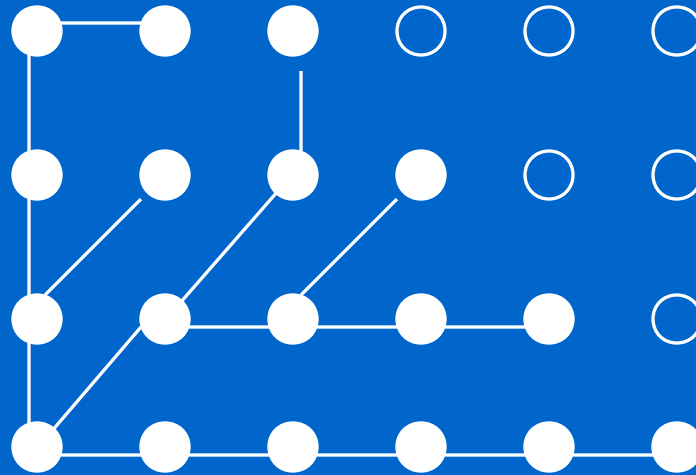
step 5a



-
-
-

Pattern Matching and Dynamic Programming

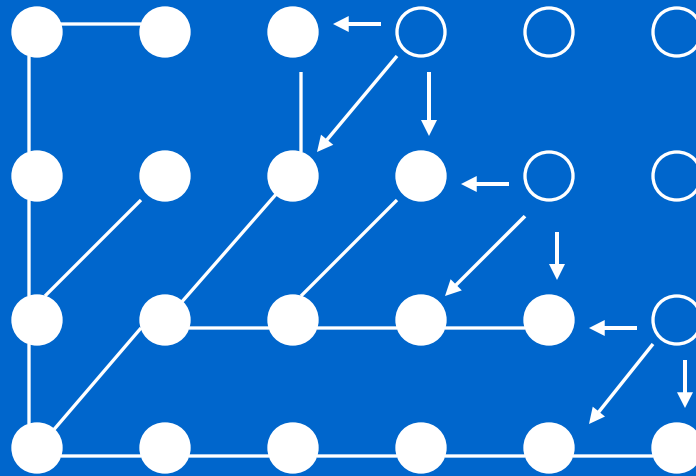
step 5b



-
-
-

Pattern Matching and Dynamic Programming

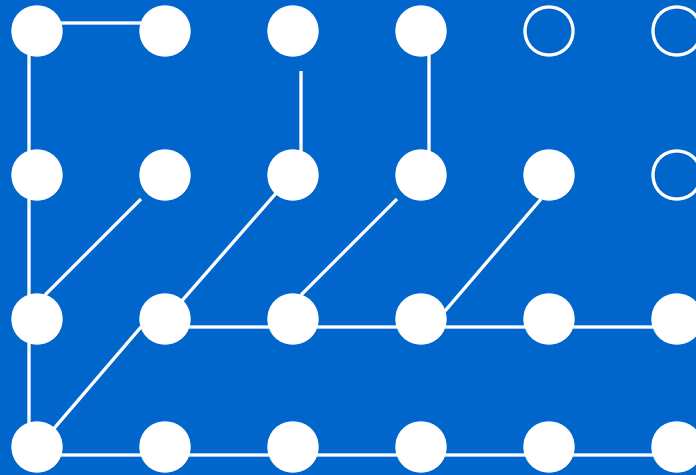
step 6a



-
-
-

Pattern Matching and Dynamic Programming

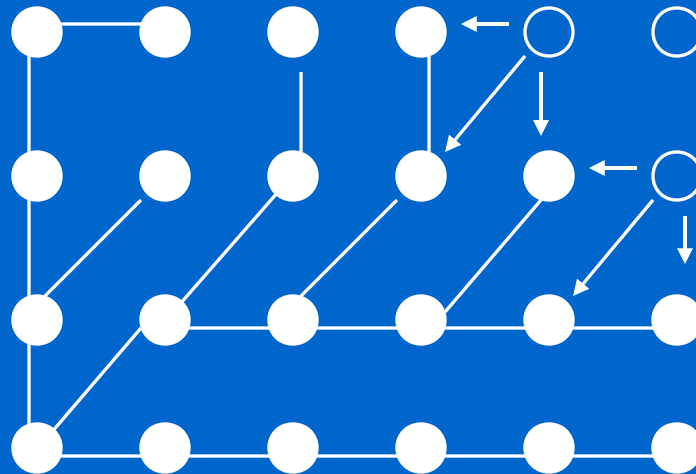
step 6b



-
-
-

Pattern Matching and Dynamic Programming

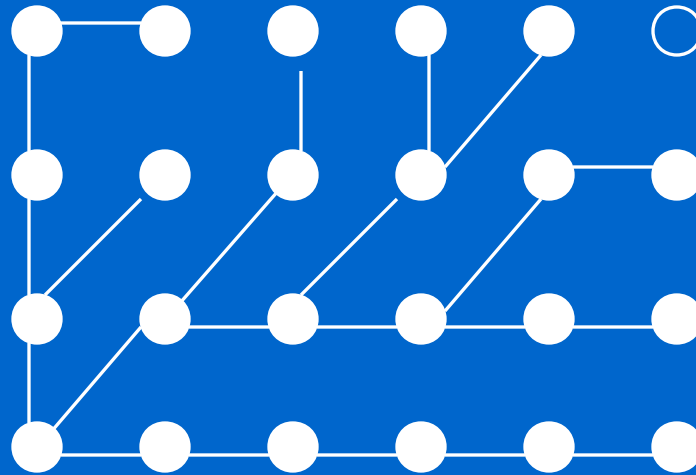
step 7a



-
-
-

Pattern Matching and Dynamic Programming

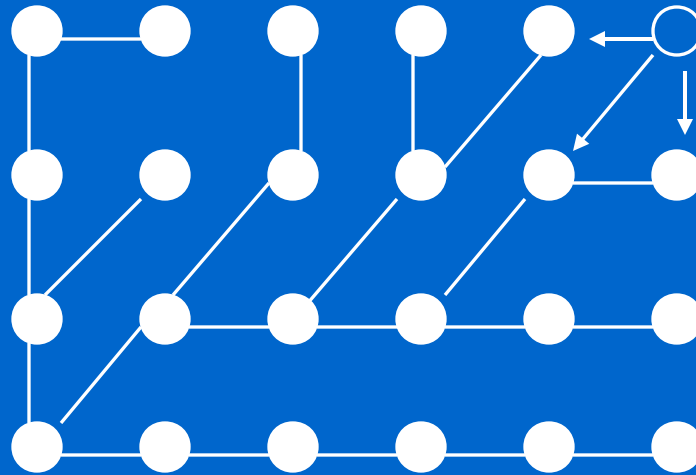
step 7b



-
-
-

Pattern Matching and Dynamic Programming

step 8a

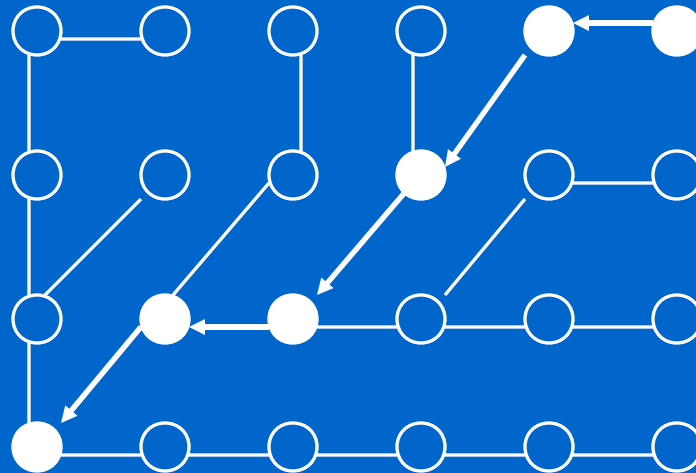


-
-
-

Pattern Matching and Dynamic Programming

step 8b : terminate

$$D_{3,5} = D(\mathbf{X}, \mathbf{M})$$

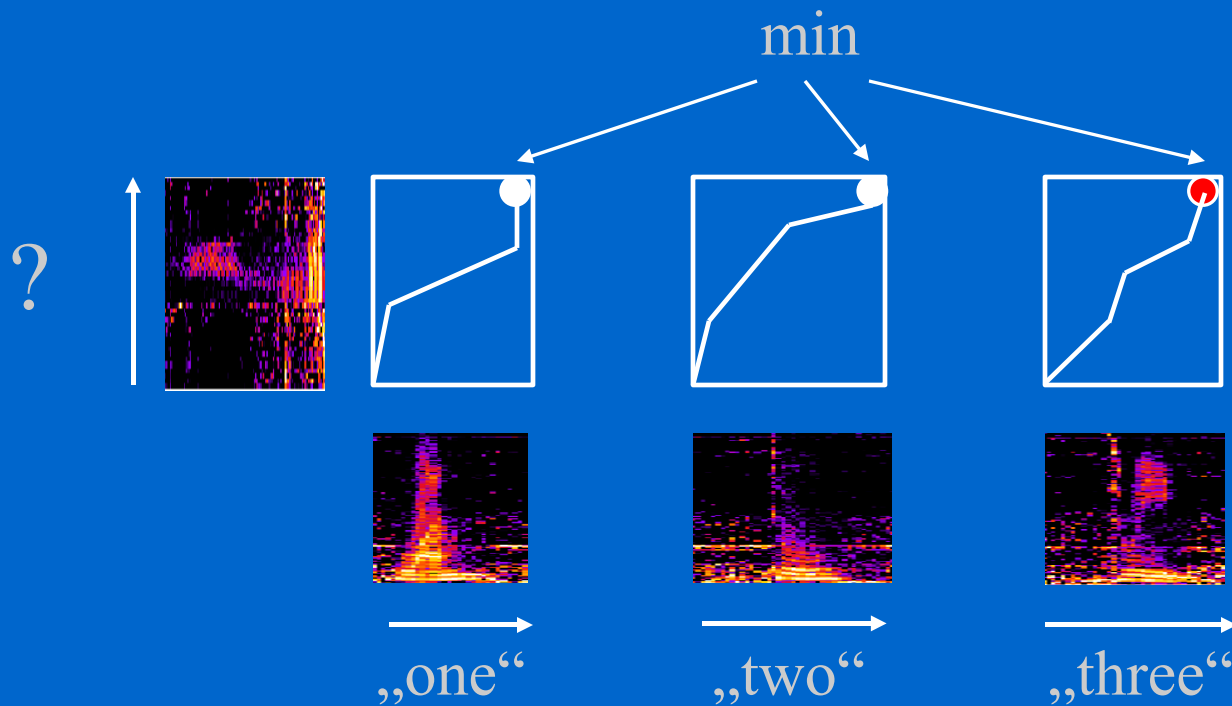


Pattern Matching and Dynamic Programming

- Summary of DP
 - DP finds the optimal time warping function needed to compare two vector sequences \mathbf{X} and \mathbf{M}
 - simultaneously, the „distance“ $D(\mathbf{X}, \mathbf{M})$ between the vector sequences is computed.

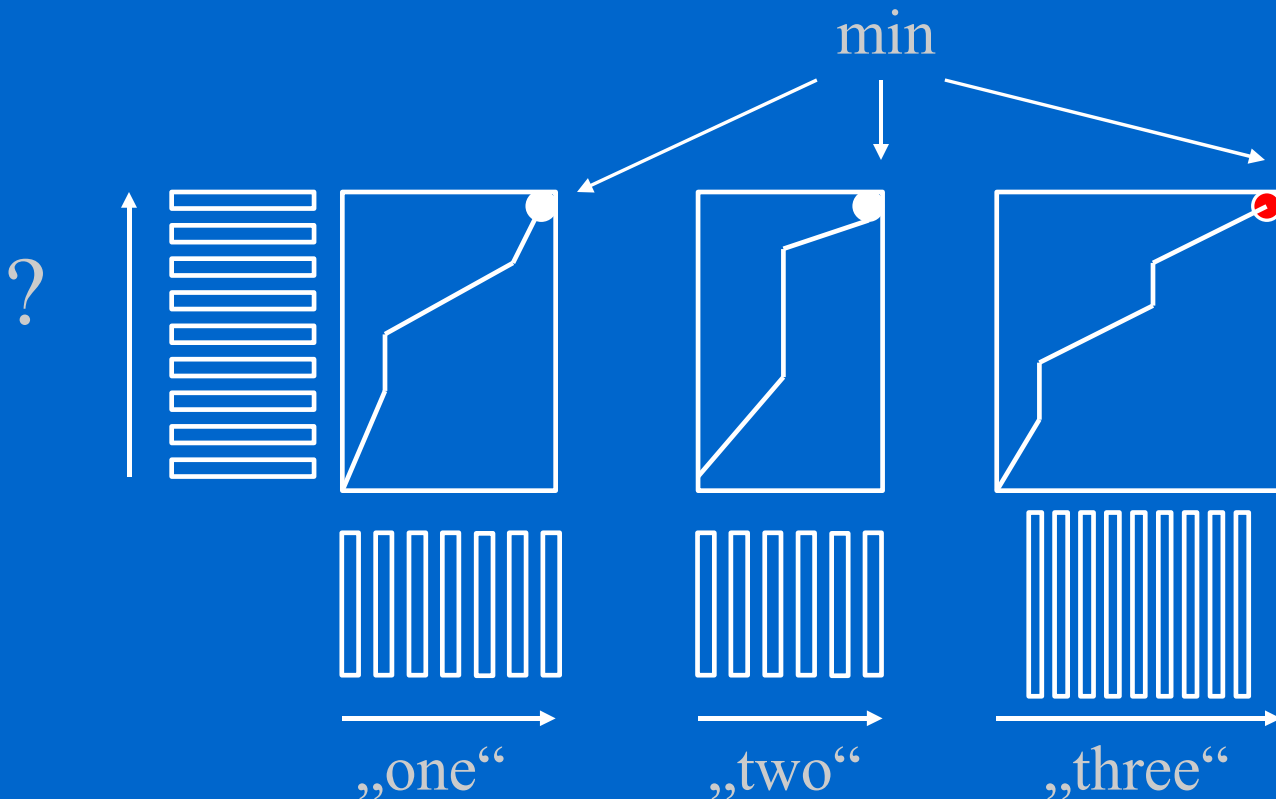
Pattern Matching and Dynamic Programming

$\tilde{X} \in \omega$, if $D(\tilde{X}, M_\omega) < D(\tilde{X}, M_j) \forall j = 1 \dots V, j \neq \omega$



Pattern Matching and Dynamic Programming

$\tilde{X} \in \omega$, if $D(\tilde{X}, M_\omega) < D(\tilde{X}, M_j) \forall j = 1..V, j \neq \omega$

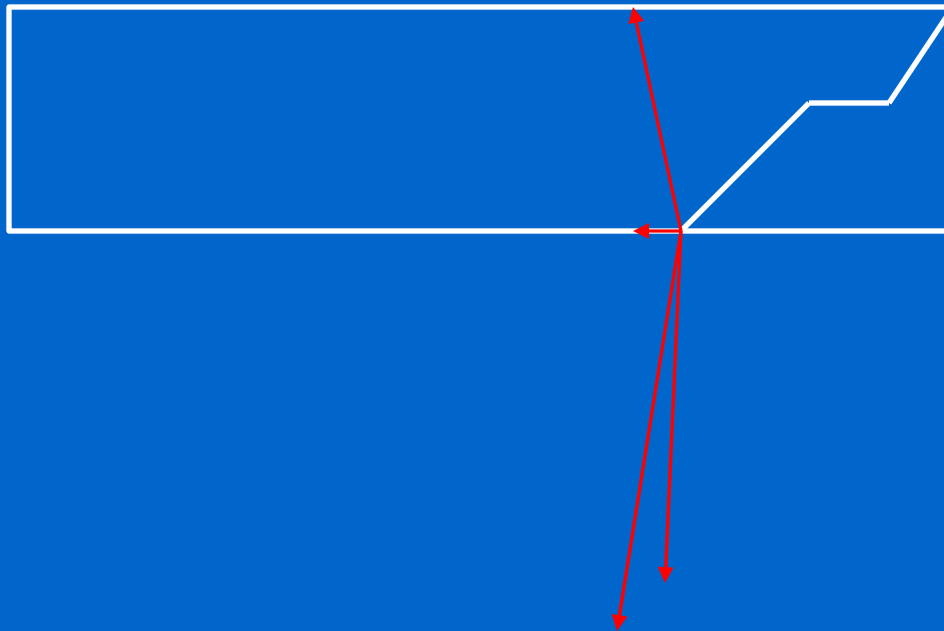


-
-
-

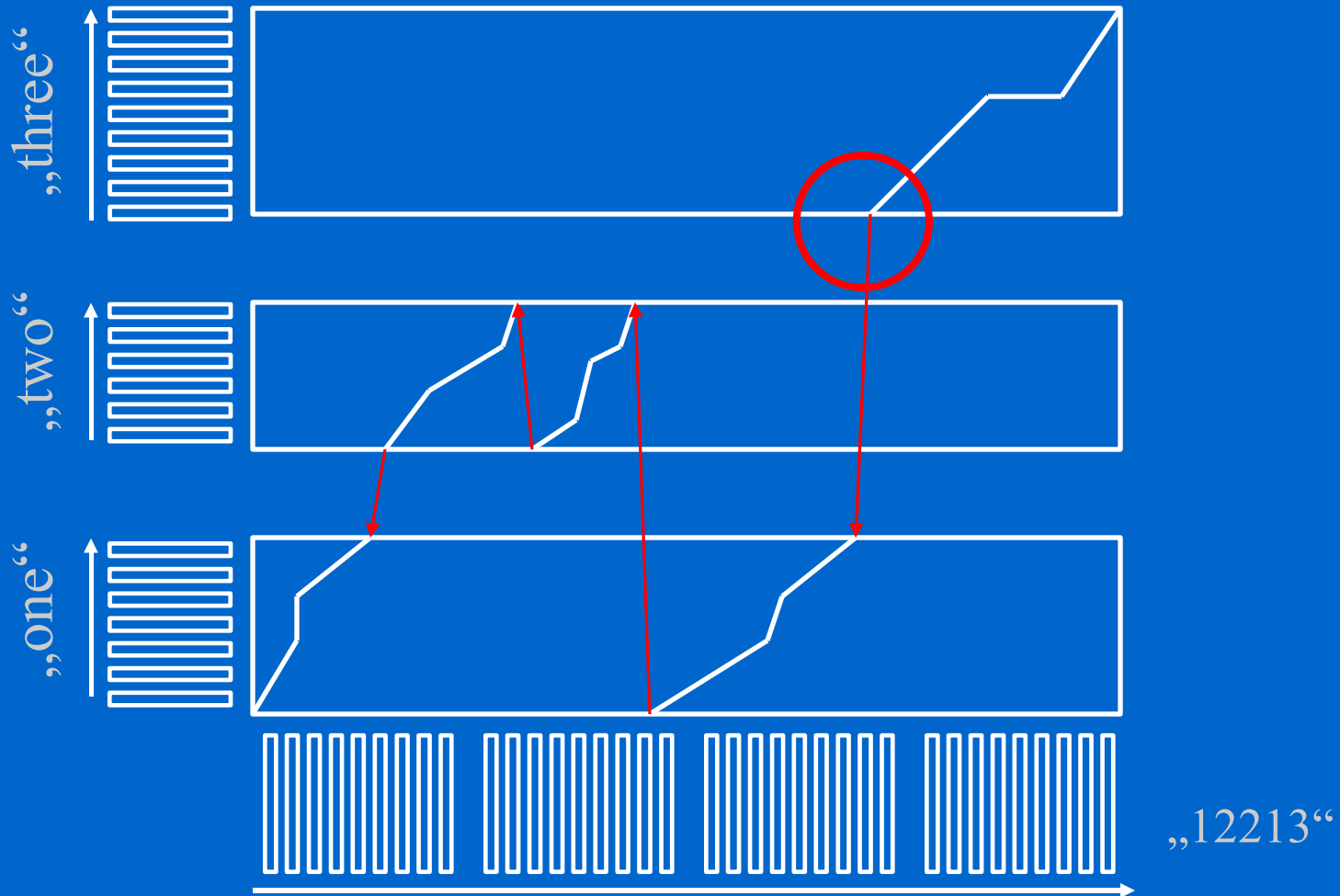
Pattern Matching and Dynamic Programming

Recognition of word sequences use new local path diagram

For first row elements of the matrix:
either choose the best word end at (t-1) or the horizontal transition from (t-1)



Pattern Matching and Dynamic Programming



Pattern Matching and Dynamic Programming

- Recognition of word sequences
 - New: different path diagram at word boundaries
 - Optimum word boundaries are automatically detected (every 10 ms a check for word boundaries is performed)
 - Recognized word sequence is defined by optimal path („backtracking“)
 - Low memory consumption: Only a column for the distances and an array for backtracking is needed.

Pattern Matching and Dynamic Programming

- Summary DP
 - optimum comparison of patterns
 - can be used for word sequences
 - synchronous processing of vectors is possible
 - computational effort grows linearly with length and number of patterns
 - easy to implement

END OF CHAPTER 2

