Speech Recognition – HMM

Jan Černocký, Valentina Hubeika {cernocky|ihubeika}@fit.vutbr.cz

FIT BUT Brno

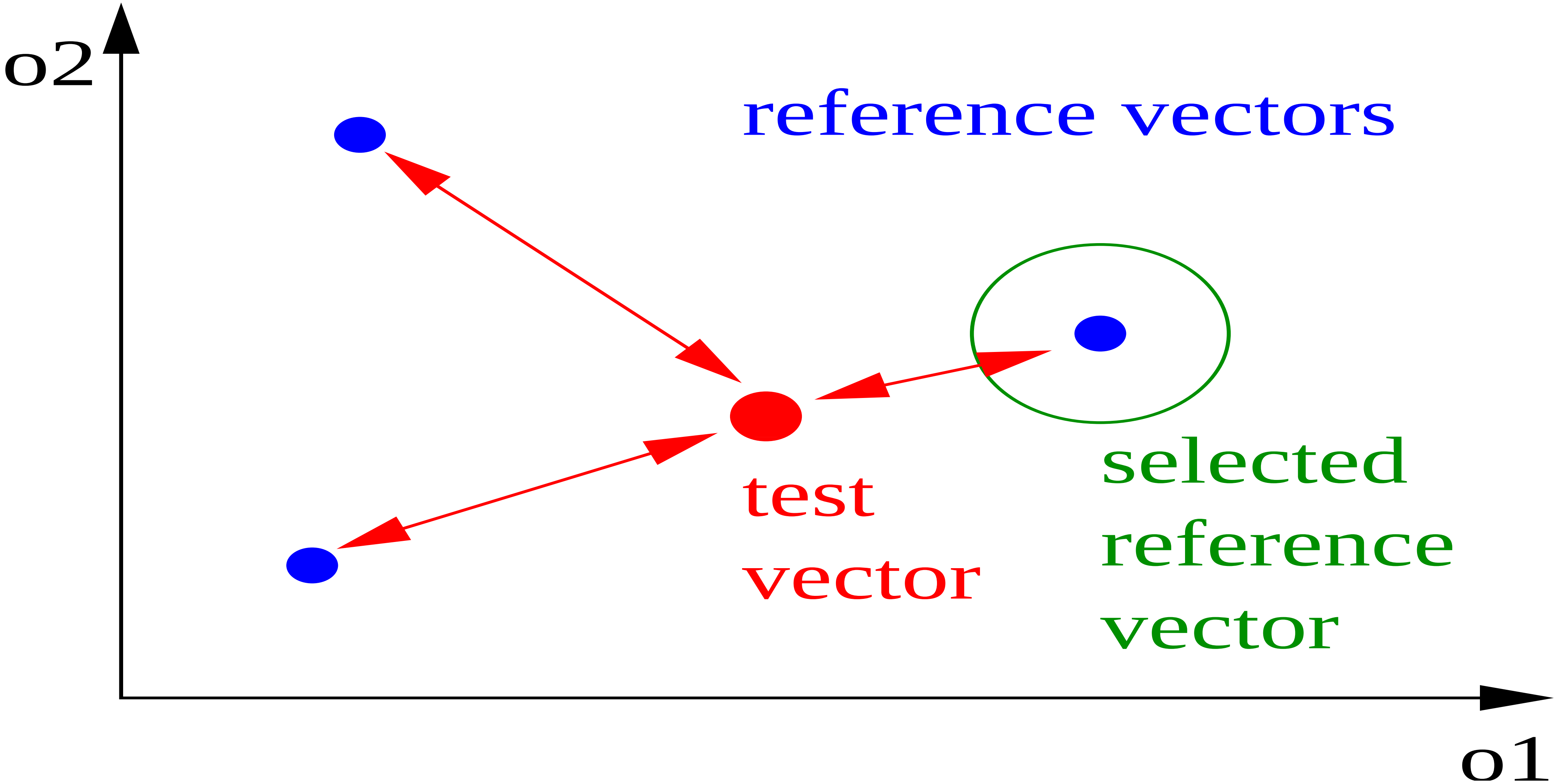
Agenda

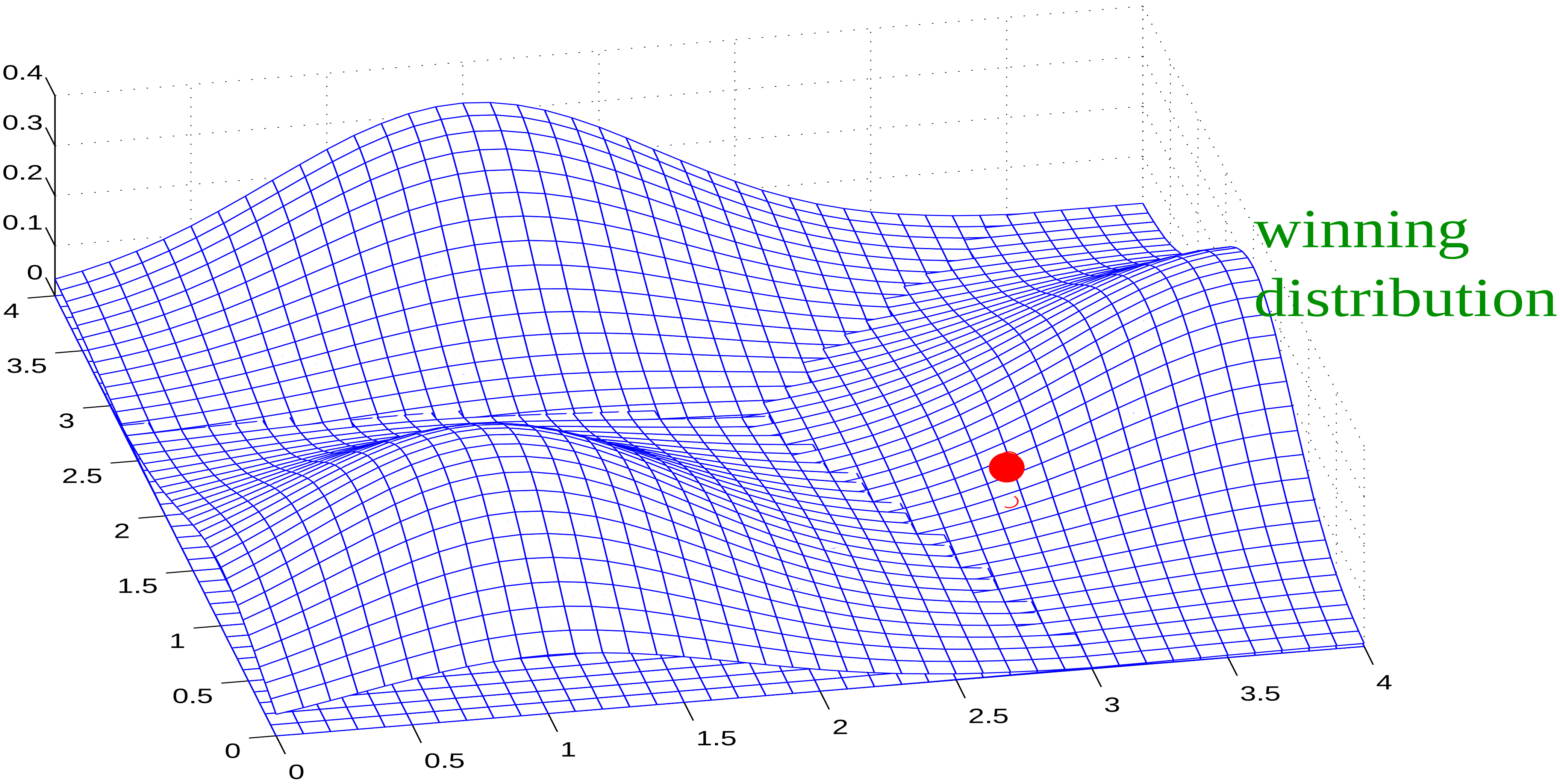
- Recap – variability in speech.
- Statistical approach.
- Model and its parameters.
- Probability of emmiting matrix $\mathbf{O}$ by model M.
- Parameter Estimation.
- Recognition.
- Viterbi a Pivo-passing.

Recap – what with variability ?

parameter space - a humane never say one thing in the same way $\Rightarrow$ parameter vectors **always differ**.

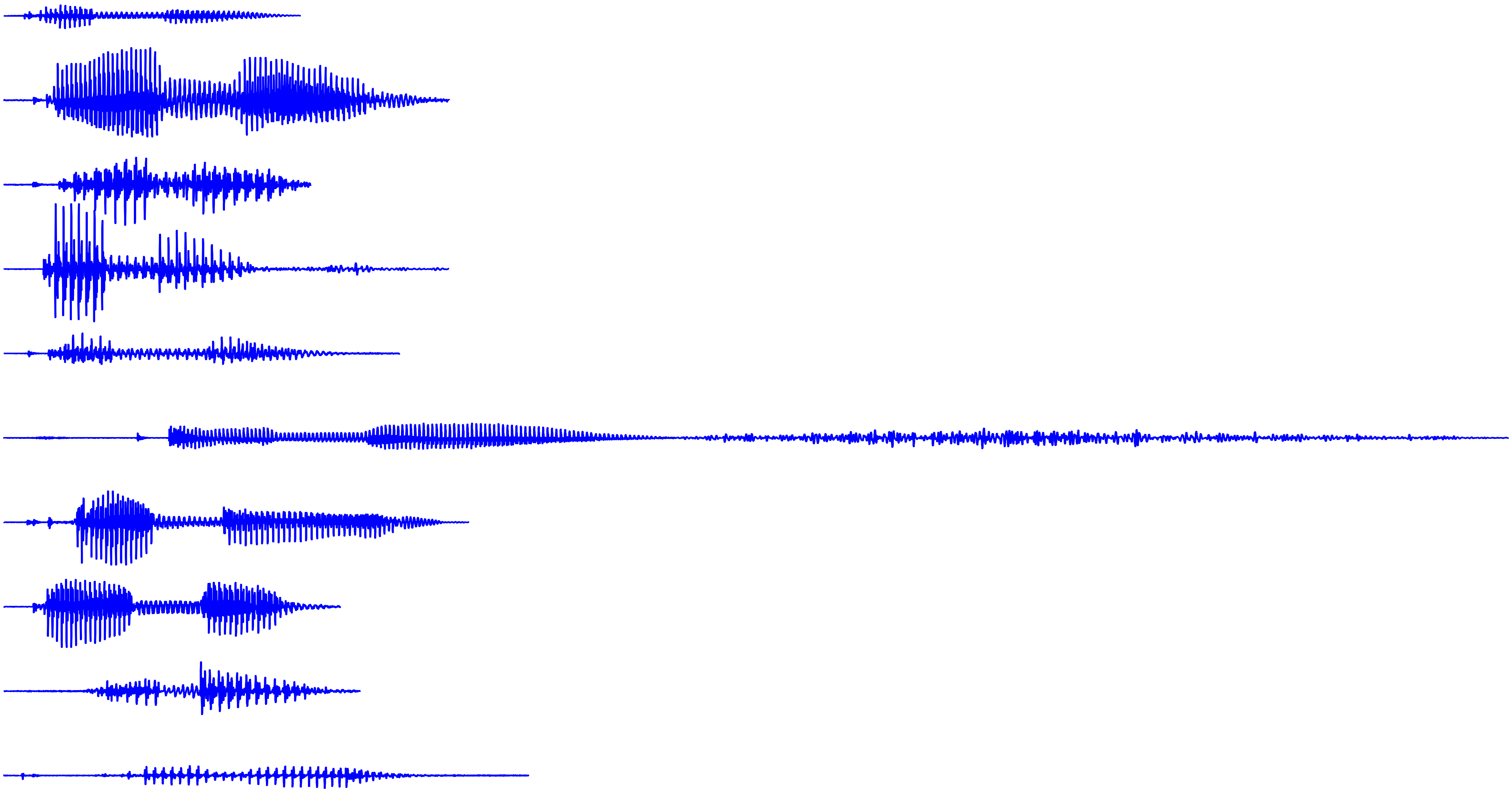
1. Distance measuring between two vectors.
2. Statical modeling.


o2
reference vectors
selected reference vector
test vector
o1

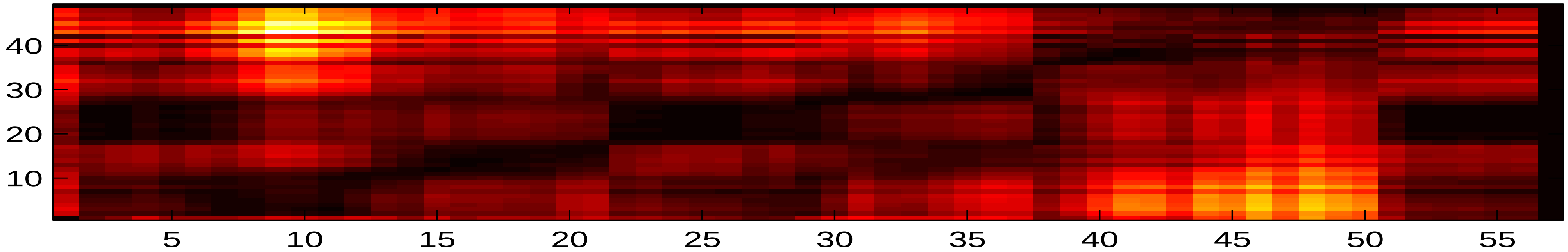


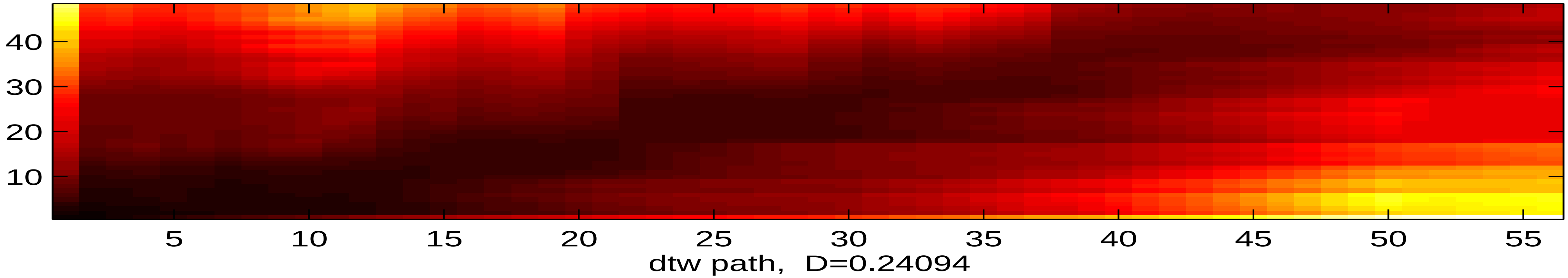
0.4
0.3
0.2
0.1
0
4
3.5
3
2.5
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3
3.5
4
winning distribution

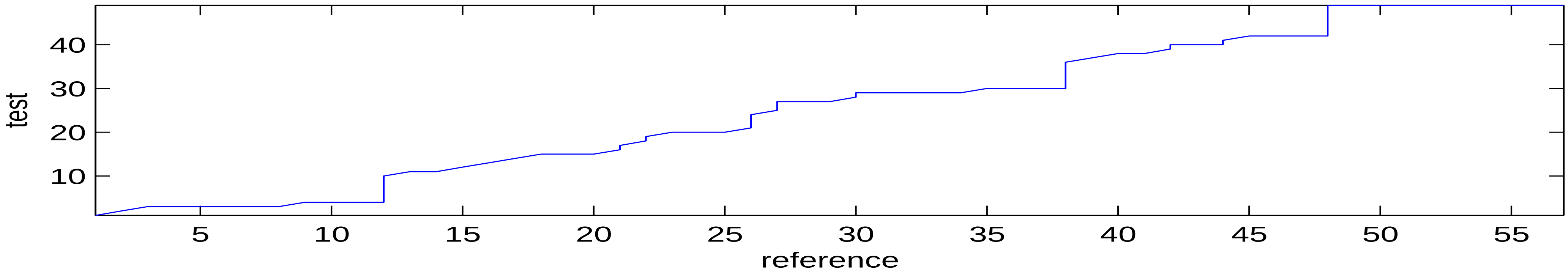
timing – people never say one thing with the same timing.



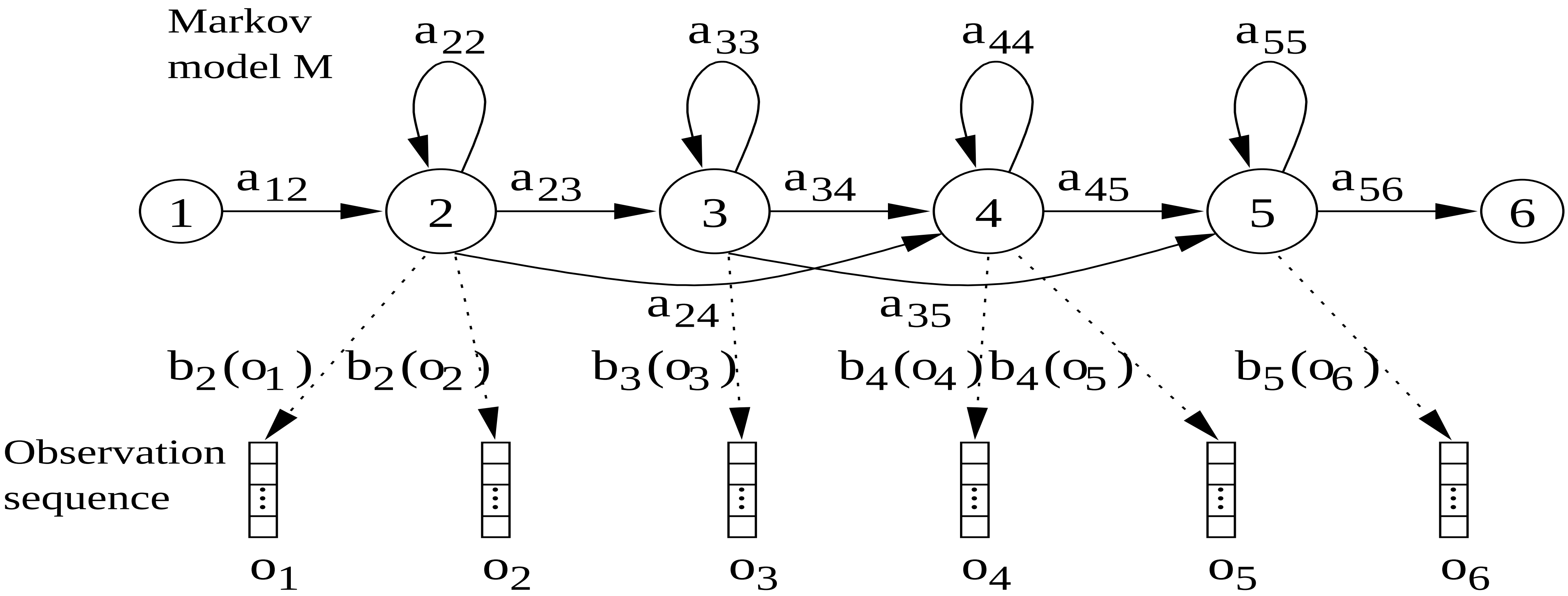
timing n.1 - Dynamic time warping - path





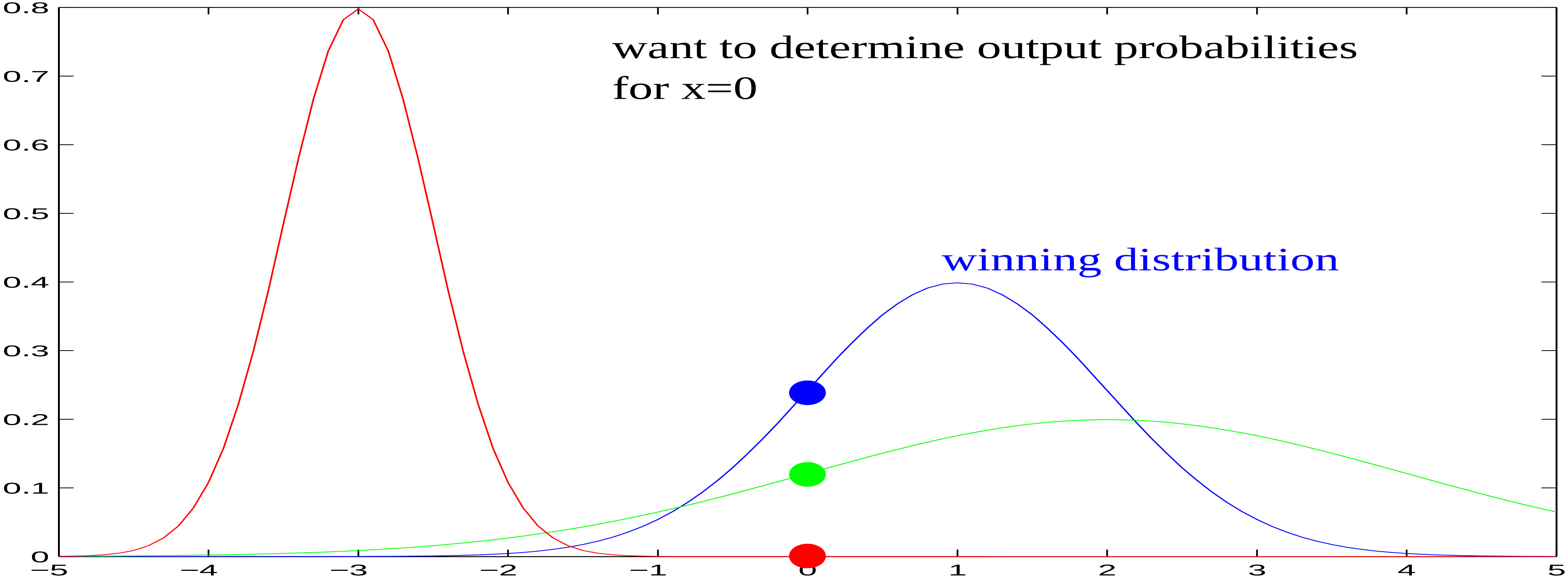


timing n.2 - Hidden Markov Models - state sequence



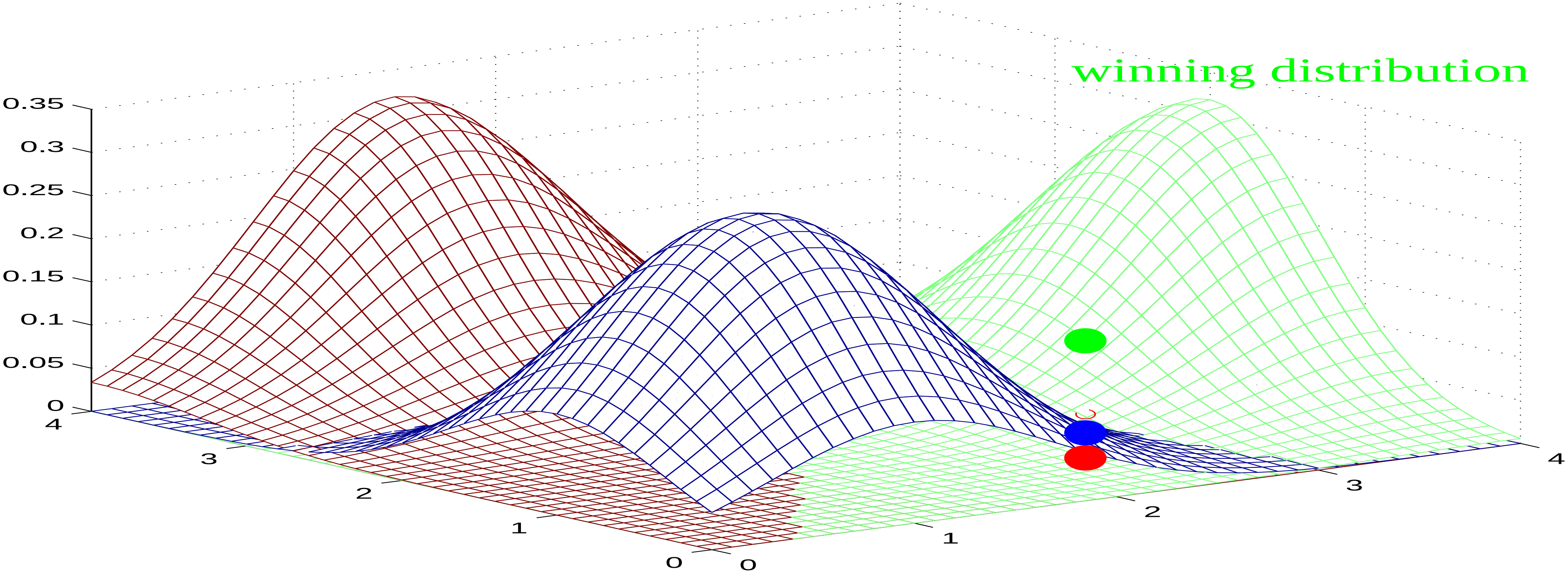
Hidden Markov Models - HMM

If words were represented by a **scalar** we would model them with a Gaussian distribution. . .



- Value $p(x)$ of the probability density function is not a real probability. Only $\int_a^b p(x)dx$ is the correct probability.
- The value $p(x)$ is sometimes called **emission probability**. If the random process would generate values x with probability $p(x)$. Out process does not generate anything literally we just call it emission probability to distinguish from transition probabilities.

. . . however, words are not represented with simple scalars but **vectors** ⇒ multivariate Gaussian distribution.



We use n-dimensional Gaussians (e.g. 39 dimensional). In the examples we re going to work with 1D, 2D or 3D.

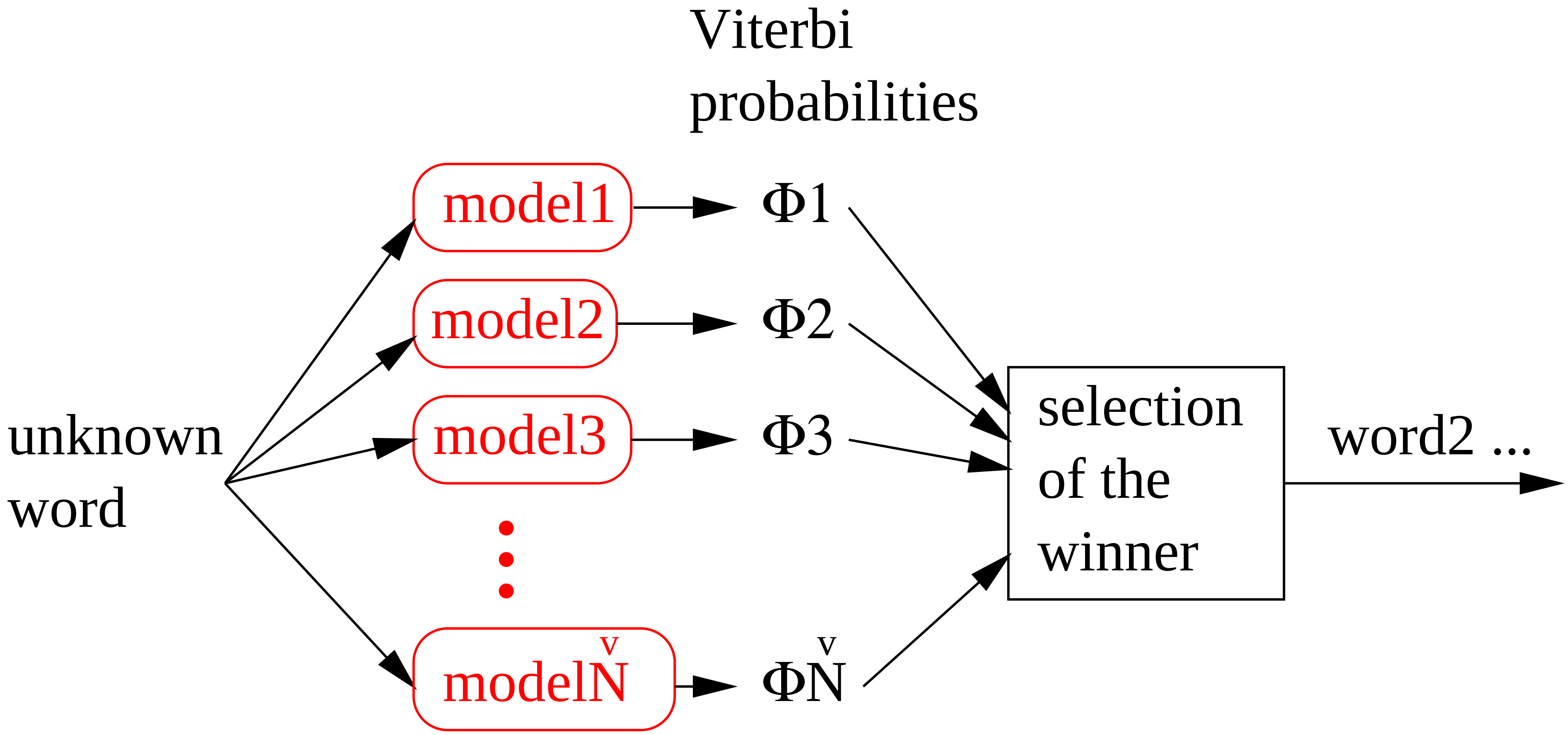
. . . again, words are not represented with just one vector but with a **sequence of vectors**

- Idea 1: 1 Gaussian per word ⇒ BAD IDEA (“paří” = “řípa” ???)
- Idea 2: each word is represented with a **sequence of Gaussians**.
  - 1 Gaussian per frame ? NO, each time a different number of vectors!!!
  - **model, where Gaussians can be repeated!**

Models, where Gaussians can repeat are called HMM

introduction on **isolated words**

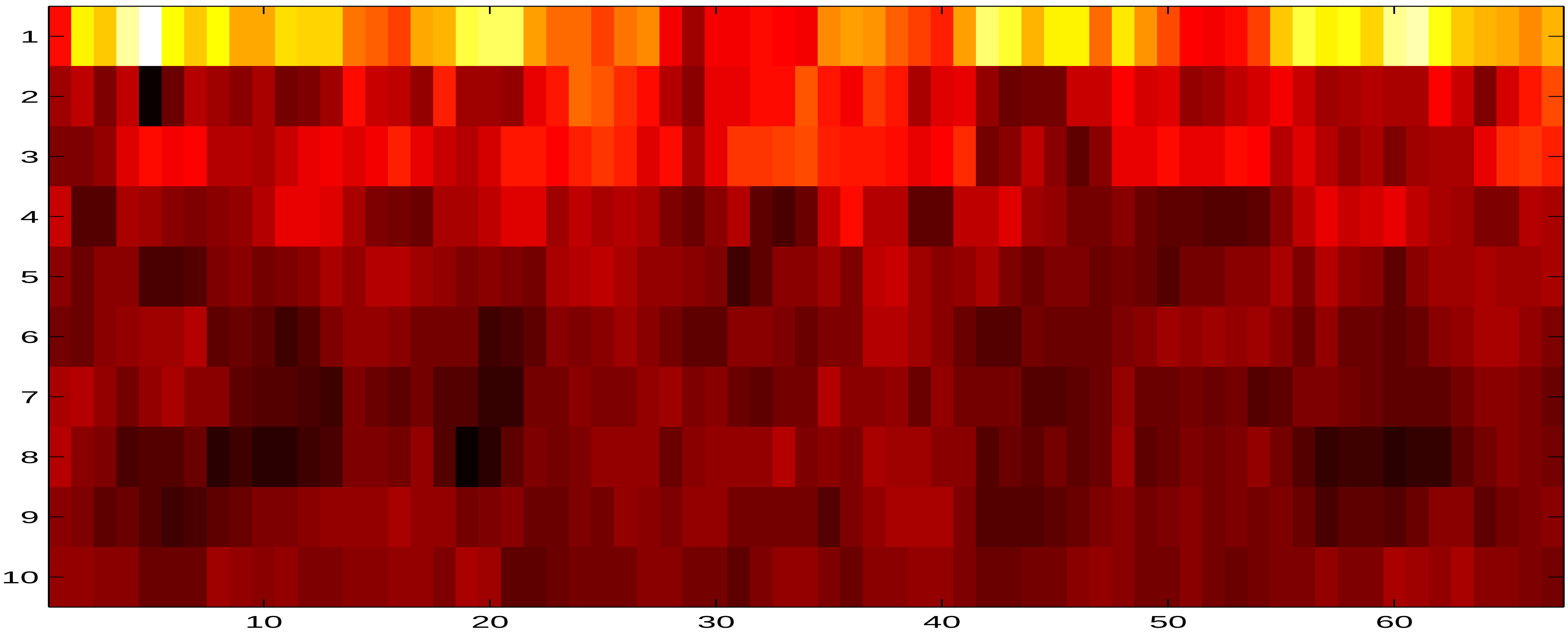
⇒ Each words we want to recognize is represented with one model



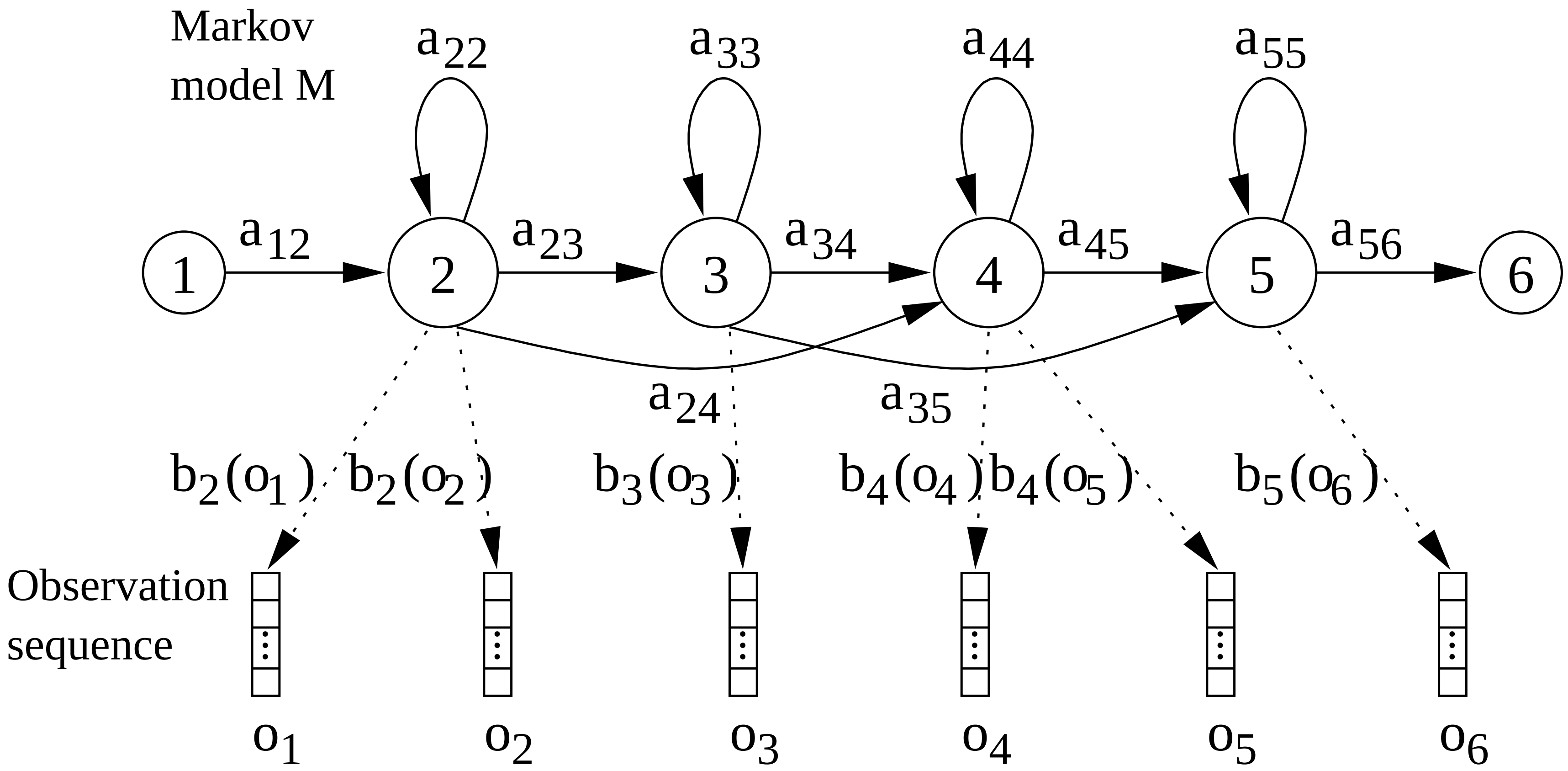
Now some math. Everything is explained on **one model**. Keep in mind, to be able to recognize, we need several models, one per word.

Input Vector Sequence

$$\mathbf{O} = [\mathbf{o}(1), \mathbf{o}(2), \ldots, \mathbf{o}(T)], \tag{1}$$



Configuration of HMM



Transmission probabilities a_{ij}

Here: only three types:

- $a_{i,i}$ probability of remaining in the same state.
- $a_{i,i+1}$ probability of passing to the following state.
- $a_{i,i+2}$ probability of skipping over the approaching state, (*not used*, for short silence models only).

$$\mathbf{A} = \begin{bmatrix} 0 & a_{12} & 0 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & 0 & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2}$$

...the matrix has nonzero item only on the diagonal and on top of it (we stay in the state or go to the following state, in DTW the path could not go back either).

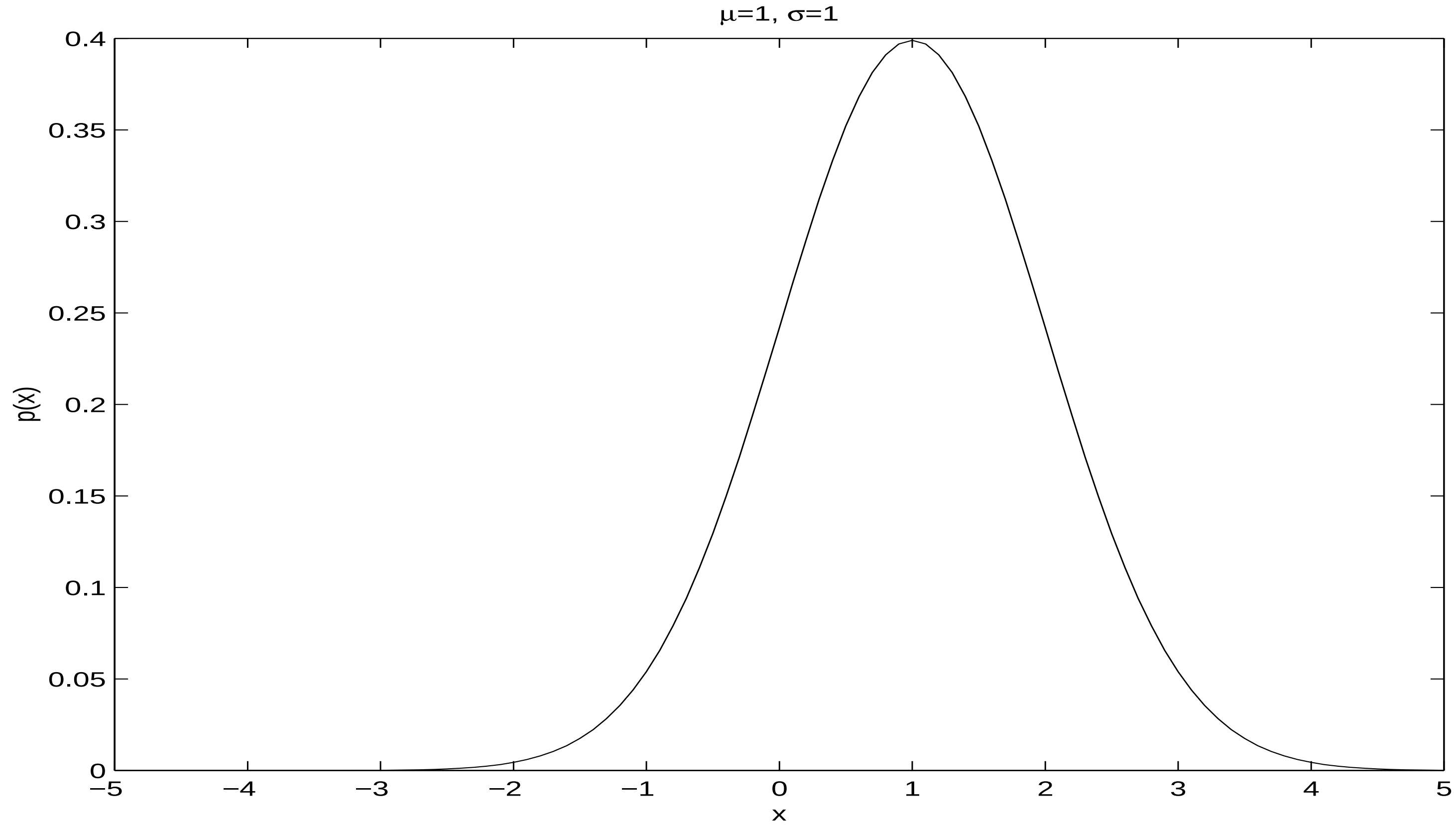
Transmission Probability Density Function (PDF)

transmission "probability" of the vector $\mathbf{o}(t)$ by the i-th state of the model is: $b_i[\mathbf{o}(t)]$.

- **discrete:** vectors are "pre-quantized" by VQ to symbols, states then contain tables with the emission probability of the symbols we will not talk about this one.
- continuous **continuous probability density function (continuous density – CDHMM)** The function is defined as Gaussian probability distribution function or a sum of several distributions.

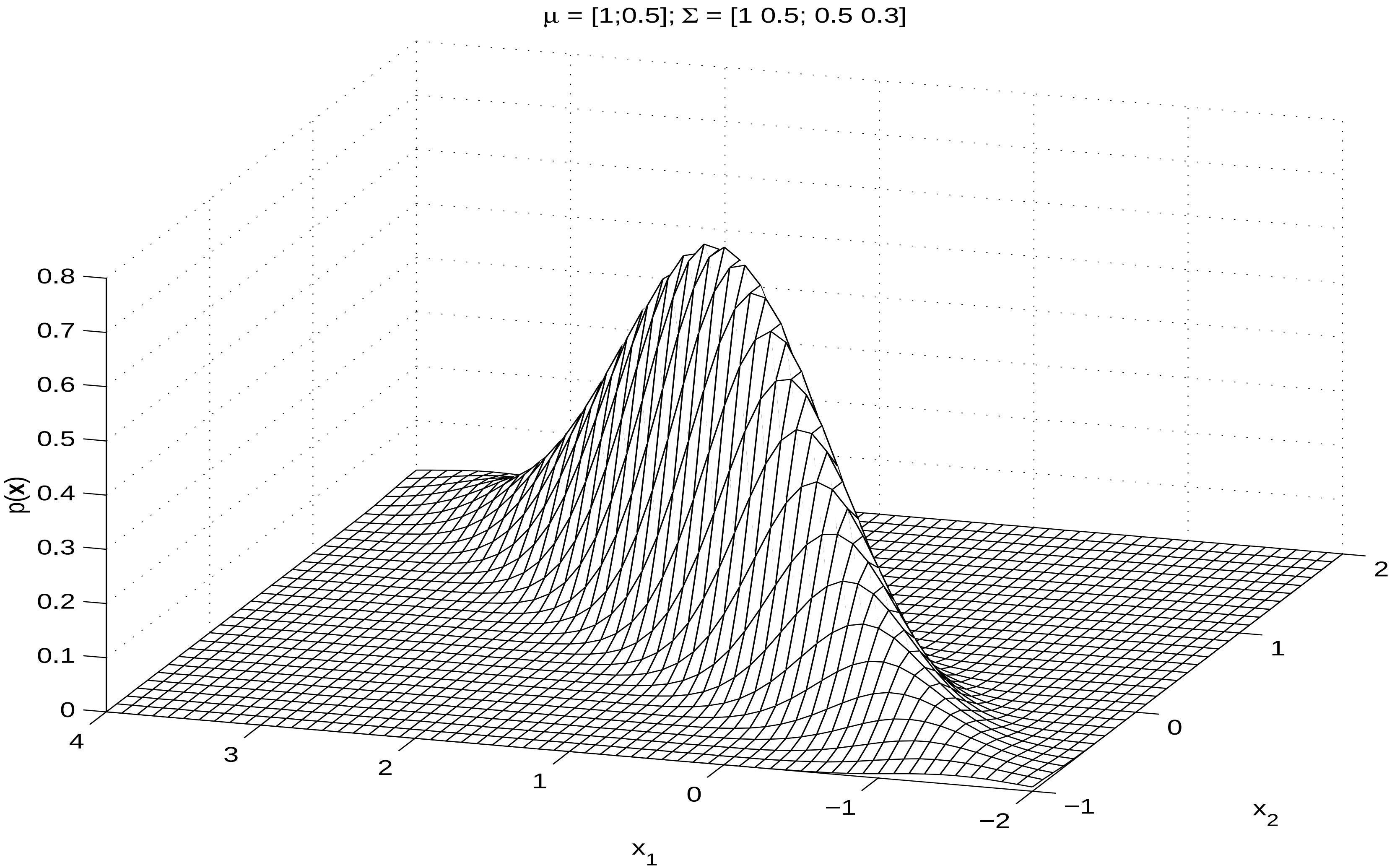
If vector $\mathbf{o}$ contains only one item:

$$b_j[o(t)] = \mathcal{N}(o(t); \mu_j, \sigma_j) = \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{[o(t)-\mu_j]^2}{2\sigma^2}} \quad (3)$$

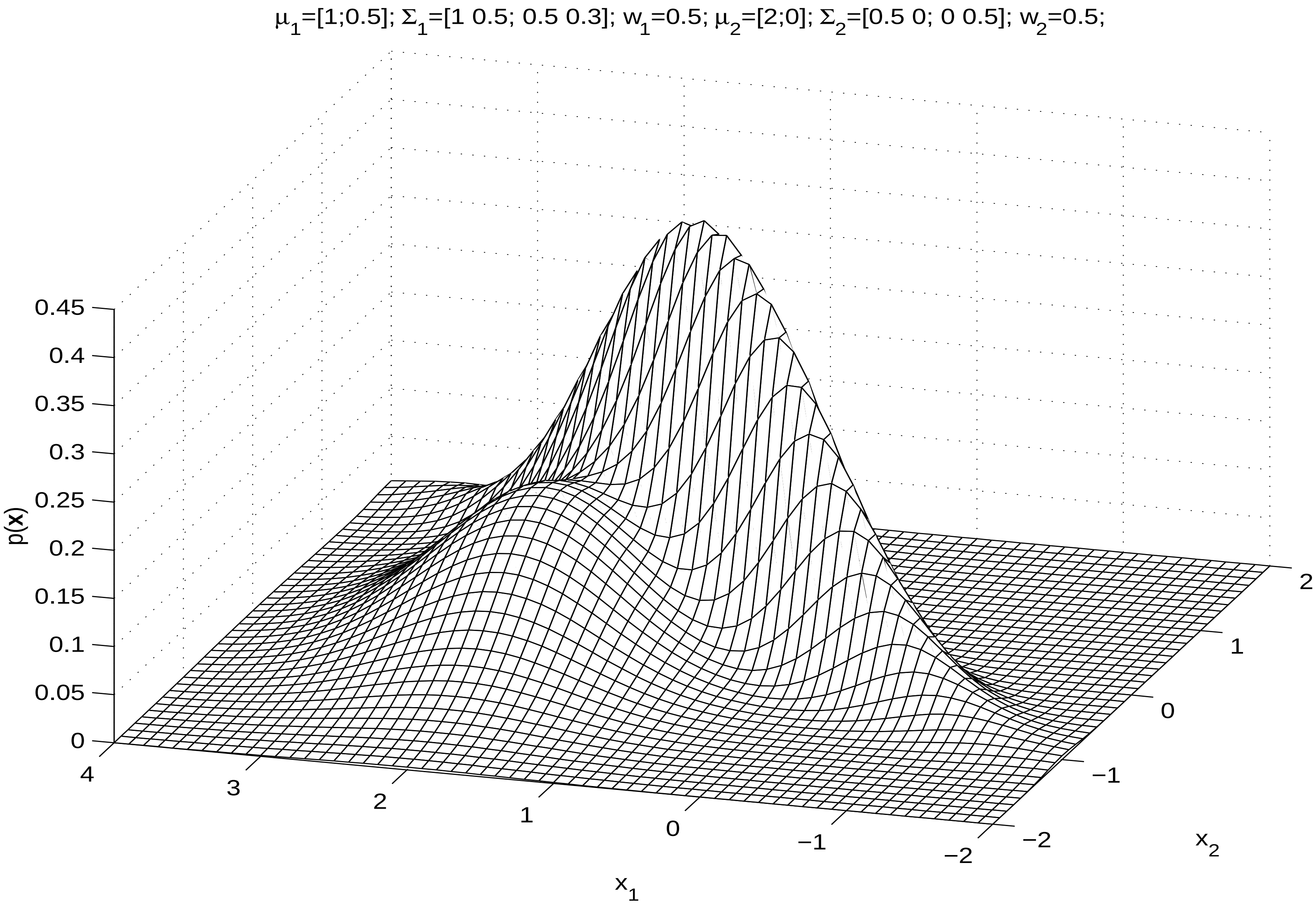


in reality vector $\mathbf{o}(t)$ is P-dimensional (usually $P = 39$):

$$b_j[\mathbf{o}(t)] = \mathcal{N}(\mathbf{o}(t); \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \frac{1}{\sqrt{(2\pi)^P |\boldsymbol{\Sigma}_j|}} e^{-\frac{1}{2}(\mathbf{o}(t)-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\mathbf{o}(t)-\boldsymbol{\mu}_j)}, \tag{4}$$



... definition of the mixture of Gaussians (with no formula):



Simplification of distribution probability:

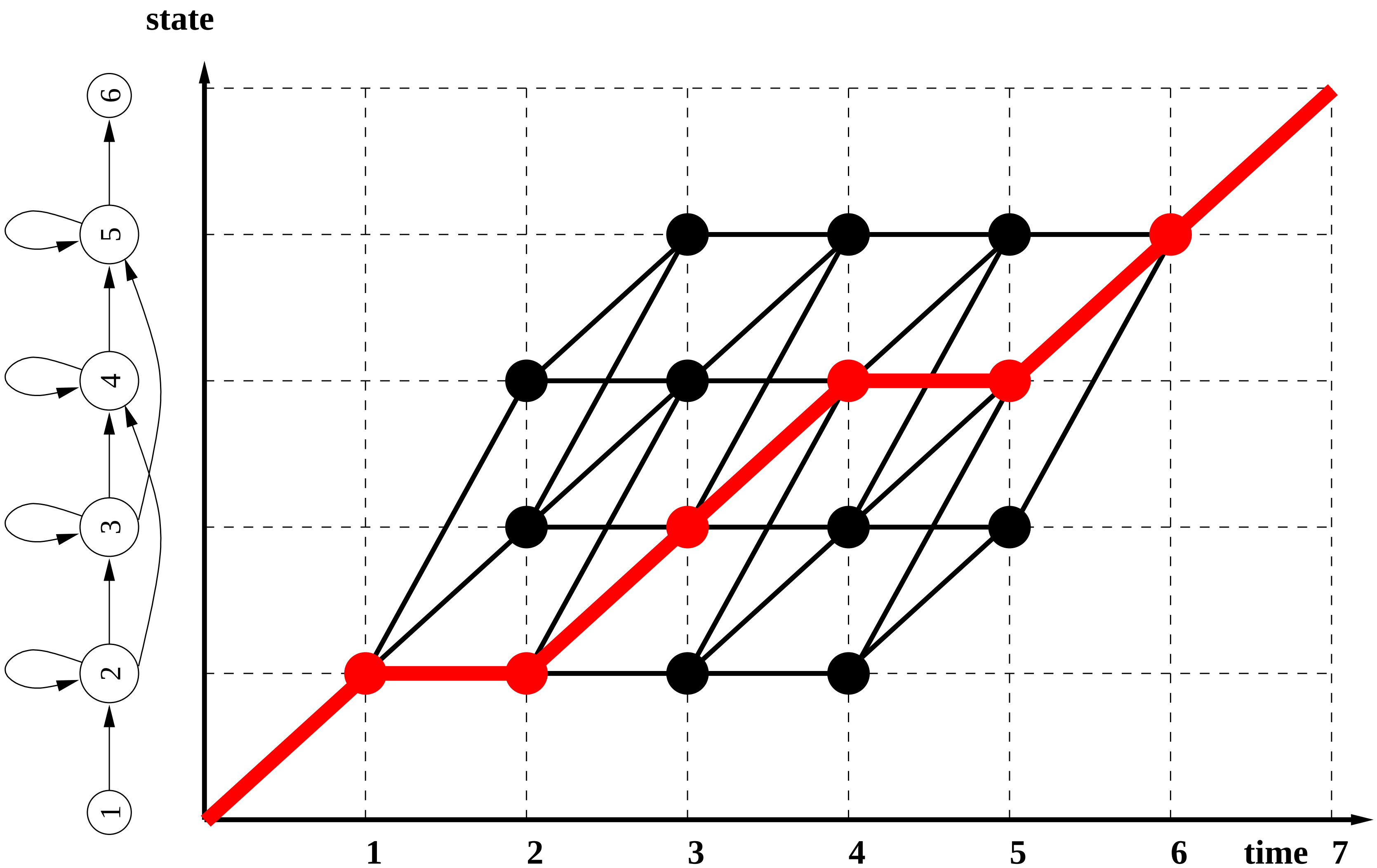
- if the parameters are not correlated (or we hope they are not), the covariance matrix is diagonal ⇒instead of $P \times P$ covariance coefficients we estimate P variances ⇒simpler models, enough data fro good estimation, total value of the probability density is given by a simple sum over the dimensions (no determinant, no inversion).

$$b_j[o(t)] = \prod_{i=1}^{P} \mathcal{N}(o(t); \mu_{ji}, \sigma_{ji}) = \prod_{i=1}^{P} \frac{1}{\sigma_{ji}\sqrt{2\pi}} e^{-\frac{[o(t)-\mu_{ji}]^2}{2\sigma_{ji}^2}} \quad (5)$$

- parameters or states can be shared among models ⇒less parameters, better estimation, less memory.

Probability with which model M generates sequence O

State sequence: states are assigned to vectors, e.g: $X = [1\ 2\ 2\ 3\ 4\ 4\ 5\ 6]$.



generation probability $\mathbf{O}$ over the path X:

$$\mathcal{P}(\mathbf{O}, X|M) = a_{x(o)x(1)} \prod_{t=1}^{T} b_{x(t)}(\mathbf{o}_t) a_{x(t)x(t+1)}, \tag{6}$$

How to define *specific* probability of generating a sequence of vectors?

a) BAUM-WELCH:

$$\mathcal{P}(\mathbf{O}|M) = \sum_{\{X\}} \mathcal{P}(\mathbf{O}, X|M), \tag{7}$$

we take the sum over *all state sequence* of the length $T+2$.

b) VITERBI:

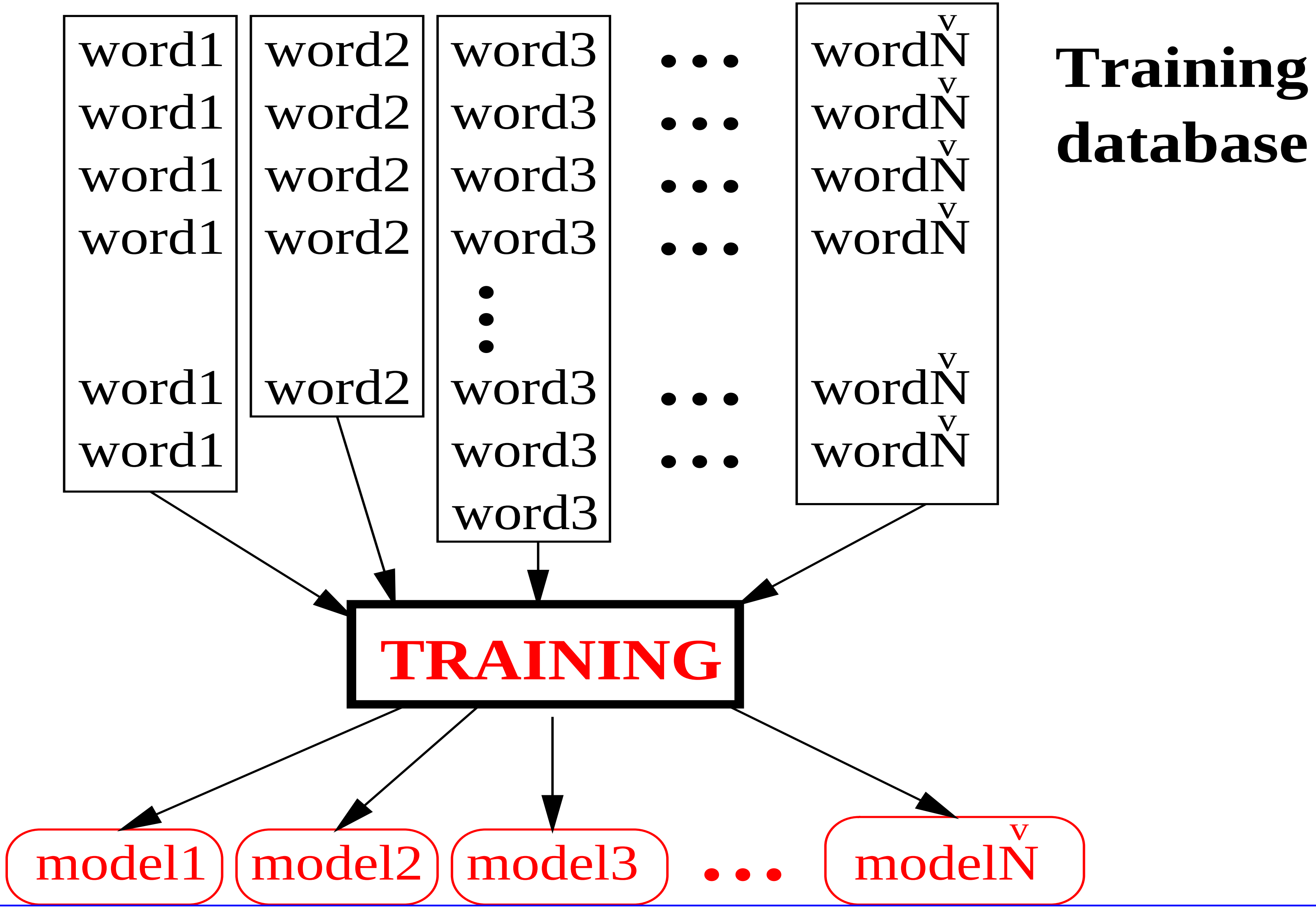
$$\mathcal{P}^{\star}(\mathbf{O}|M) = \max_{\{X\}} \mathcal{P}(\mathbf{O}, X|M), \tag{8}$$

is probability of the optimal path.

Notes

1. In DTW we were looking *minimal distance*. Here we are searching *maximal probability*, sometimes also called *likelihood* $\mathcal{L}$.
2. Fast algorithms for Baum-Welch and Viterbi: we don't evaluate all possible state sequences X.

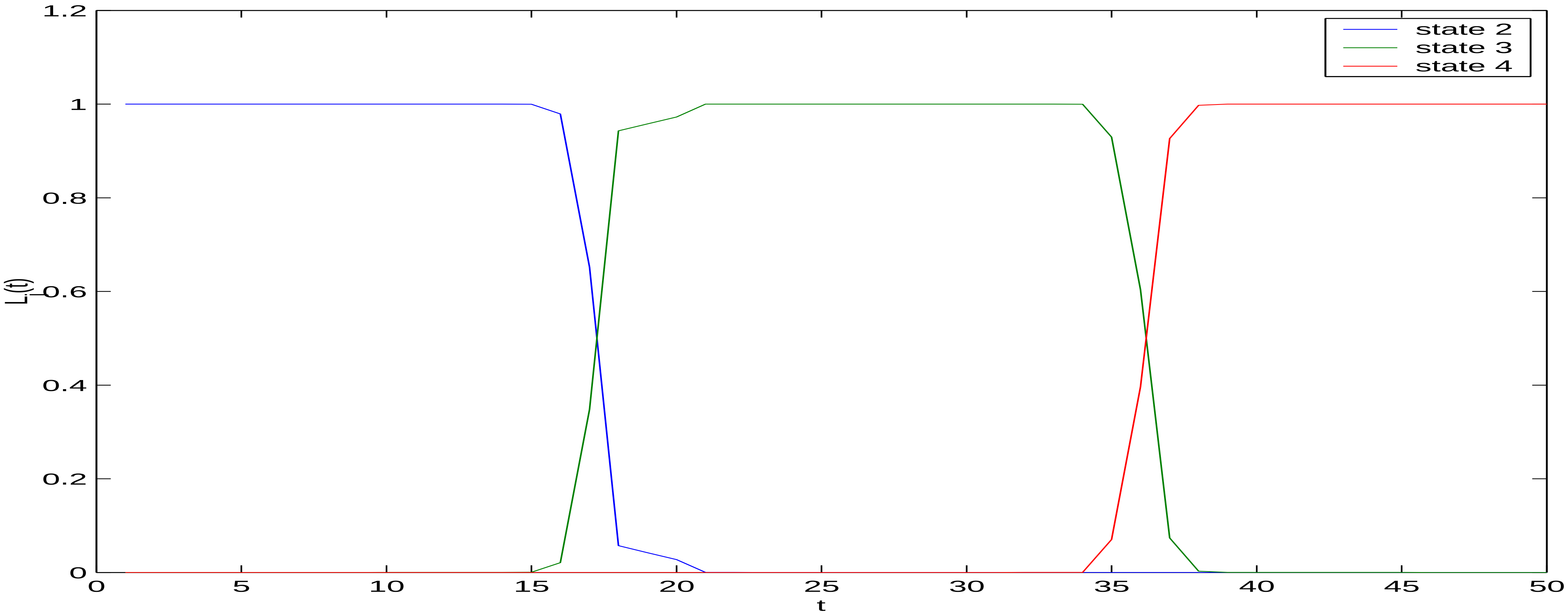
Parameter Estimation (aren't in tables :-) . . .)



1. parameters of a model are **roughly estimated** :

$$\hat{\boldsymbol{\mu}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{o}(t) \quad \hat{\boldsymbol{\Sigma}} = \frac{1}{T}\sum_{t=1}^{T}(\mathbf{o}(t)-\boldsymbol{\mu})(\mathbf{o}(t)-\boldsymbol{\mu})^T \tag{9}$$

2. vectors are **aligned to states**: "hard" or "soft" using function $L_j(t)$ (state occupation function).



3. **new estimation of parameters:**

$$\hat{\boldsymbol{\mu}}_j = \frac{\sum_{t=1}^{T} L_j(t)\mathbf{o}(t)}{\sum_{t=1}^{T} L_j(t)} \quad \hat{\boldsymbol{\Sigma}}_j = \frac{\sum_{t=1}^{T} L_j(t)(\mathbf{o}(t)-\boldsymbol{\mu}_j)(\mathbf{o}(t)-\boldsymbol{\mu}_j)^T}{\sum_{t=1}^{T} L_j(t)}. \tag{10}$$

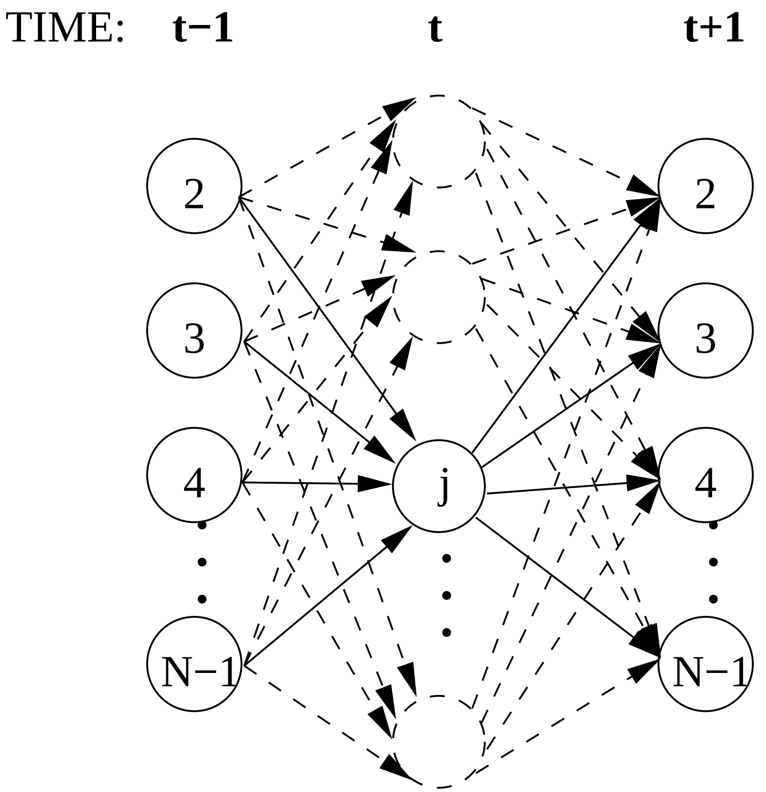
...similar equations for transition probabilities a_{ij}.

Steps 2) and 3) repeat: stop criterion is either a fixed number of iterations or too little improvement in probabilities.

- the algorithm is denoted as EM (Expectation Maximization): dá we compute $\sum p$(each vector|old parameters)$\times \log p$(each vector|new parameters), which is called expectation (average on data) of the overall likelihood. This is the objective function that is maximized $\Rightarrow$ leads to maximizing likelihood.
- maximize likelihood that data "generated" by the model $\Rightarrow$ LM (likelihood maximization) - a little bit of a problem as the models learn to represent words the best rather than **discriminate** between them.

State Occupation Functions

Probability $L_j(t)$ of "being in the state j at time t" can be calculated as a sum of all paths that in time t are in state j : $P(\mathbf{O}, x(t) = j|M)$



We need to ensure real probabilities, thus:

$$\sum_{j=1}^{N} L_j(t) = 1, \tag{11}$$

contribution of a vector to all states has to sum up to 100%. Need to normalize by the sum of likelihoods of all paths in time t

$$L_j(t) = \frac{P(\mathbf{O}, x(t) = j|M)}{\sum_j P(\mathbf{O}, x(t) = j|M)}. \tag{12}$$

... this are all paths through the model so we can use BAUM-WELCH probability:

$$L_j(t) = \frac{P(\mathbf{O}, x(t) = j|M)}{P(\mathbf{O}|M)}. \tag{13}$$

How do we do it $P(\mathbf{O}, x(t) = j|M)$

1. partial forward probabilities (starting at the first state and being at time t in state j):

$$\alpha_j(t) = \mathcal{P}\left(\mathbf{o}(1) \dots \mathbf{o}(t), x(t) = j|M\right) \tag{14}$$

2. partial backward probabilities (starting at the last state, being at time t in state j):

$$\beta_j(t) = \mathcal{P}\left(\mathbf{o}(t+1) \dots \mathbf{o}(T)|x(t) = j, M\right) \tag{15}$$

3. state occupation function is then given as:

$$L_j(t) = \frac{\alpha_j(t)\beta_j(t)}{P(\mathbf{O}|M)} \tag{16}$$

. . . more in HTK Book.

Model training Algorithm

1. Allocate accumulator for each estimated vector/matrix.
2. Calculate forward and backward probabilities. $\alpha_j(t)$ and $\beta_j(t)$ for all times t and all states j. Compute state occupation functions $L_j(t)$.
3. To each accumulator add contribution from vector $\mathbf{o}(t)$ weighted by corresponding $L_j(t)$.
4. Use the resulting accumulator to estimate vector/matrix using equation 10 — dont forget to normalize by $\sum_{t=1}^{T} L_j(t)$.
5. If the value $\mathcal{P}(\mathbf{O}|M)$ doesnt improve comparing to the previous iteration, stop, otherwise Goto 1.

Recognition – Viterbi algorithm

- we have to recognize unknown sequence $\mathbf{O}$.
- in the vocabulary we dispose of $\check{N}$ words $w_1 \dots w_{\check{N}}$.
- for each word we have a trained model $M_1 \dots M_{\check{N}}$.
- The question is: "Which model generates sequence $\mathbf{O}$ with the highest likelihood?"

$$i^\star = \arg\max_i \{\mathcal{P}(\mathbf{O}|M_i)\} \quad (17)$$

we will use VITERBI probability to estimate the most probable state sequence:

$$\mathcal{P}^\star(\mathbf{O}|M) = \max_{\{X\}} \mathcal{P}(\mathbf{O}, X|M). \quad (18)$$

thus:

$$i^\star = \arg\max_i \{\mathcal{P}^\star(\mathbf{O}|M_i)\}. \quad (19)$$

VITERBI probability computation:

– similar to BAUM-WELCH.

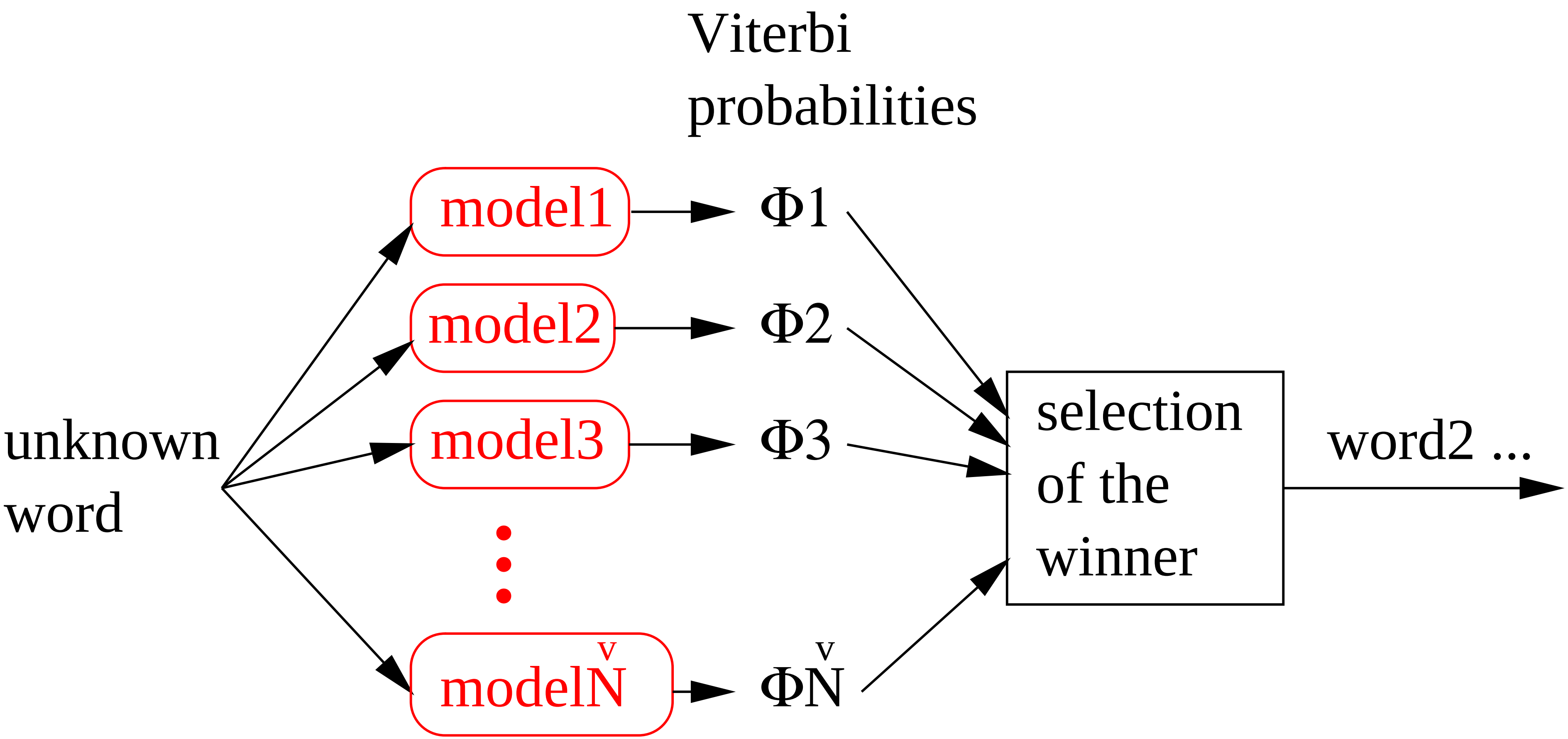
partial VITERBI probability:

$$\Phi_j(t) = \mathcal{P}^\star \left(\mathbf{o}(1) \ldots \mathbf{o}(t), x(t) = j|M\right). \tag{20}$$

For the j-th state and time t. Desired VITERBI probability is:

$$\mathcal{P}^\star(\mathbf{O}|M) = \Phi_N(T) \tag{21}$$

Isolated Words Recognition using HMM



Viterbi: Token passing

each model state has a glass with beer. We work with log-probabilities $\Rightarrow$addition. We will "pour in" probabilities.

Initialization: insert and empty glass to each input state of the model (usually state 1).

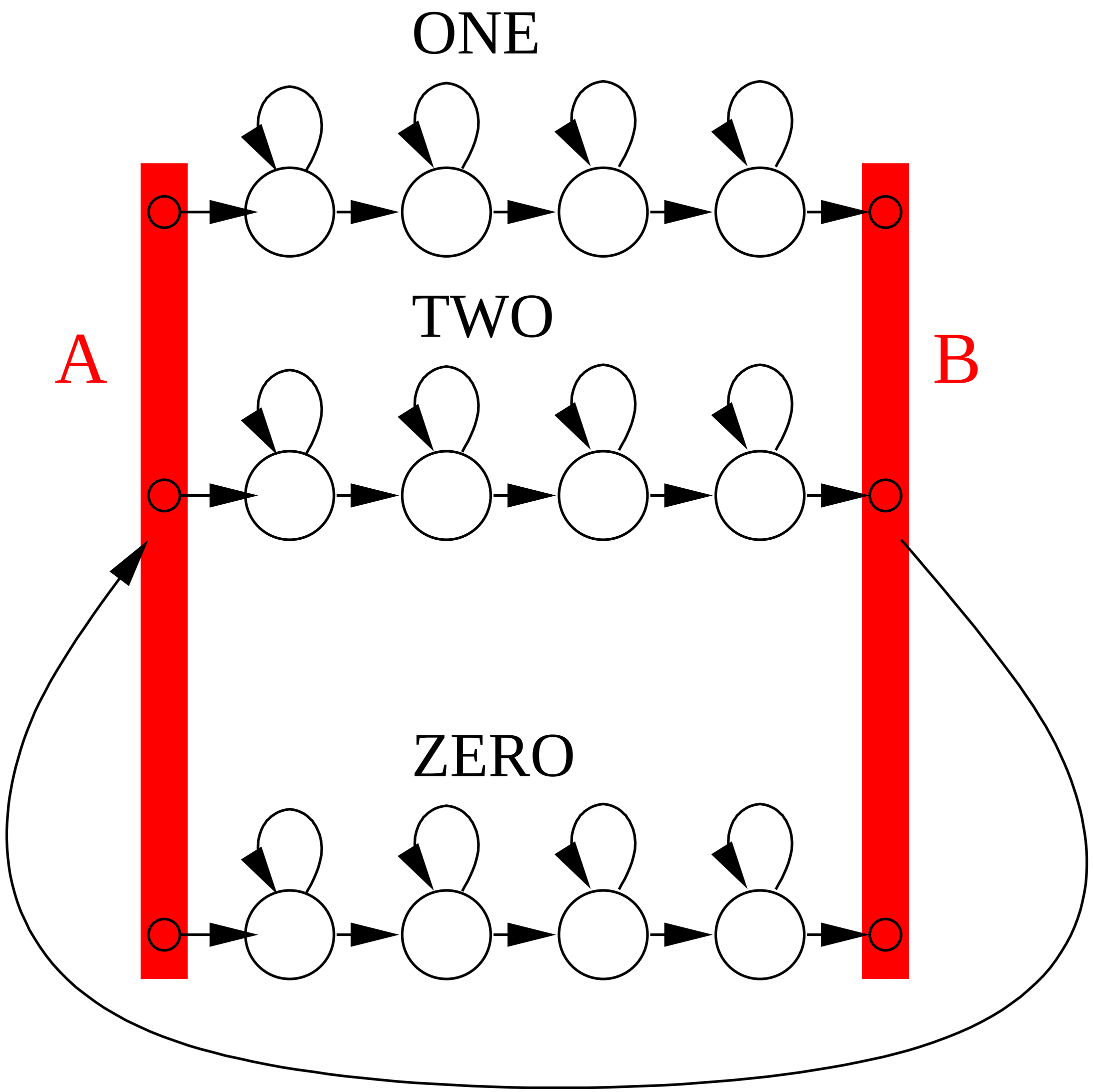
Iteration: for times $t = 1 \dots T$:

- in each state i, that containes the glass, clone the glass and send the copy to all connected states j. On the way add $\log a_{ij} + \log b_j[\mathbf{o}(t)]$.
- if in one state we have more than one glass, the the one that is contains more beer.

Finish: from all states i connected with the finish state N, that contain glass, send them to N and add $\log a_{iN}$. In the last state N, choose the fullest glass and leave out the other ones. The level of beer in state N correspond to the log Viterbi likelihood: $\log \mathcal{P}^{\star}(\mathbf{O}|M)$.

Pivo passing for Connected Words

construct a mega-model from all words, "glue" the first and the last states of two models.



Recognition – similar to isolated words, we need to remember the optimal words the glass was passed through.

Further reading

- HTK Book - `http://htk.eng.cam.ac.uk/`
- Gold and Morgan: in the FIT library.
- Computer labs on HMM-HTK.