

Rozpoznávání řeči – HMM

Jan Černocký ÚPGM FIT VUT Brno, cernocky@fit.vutbr.cz

FIT VUT Brno

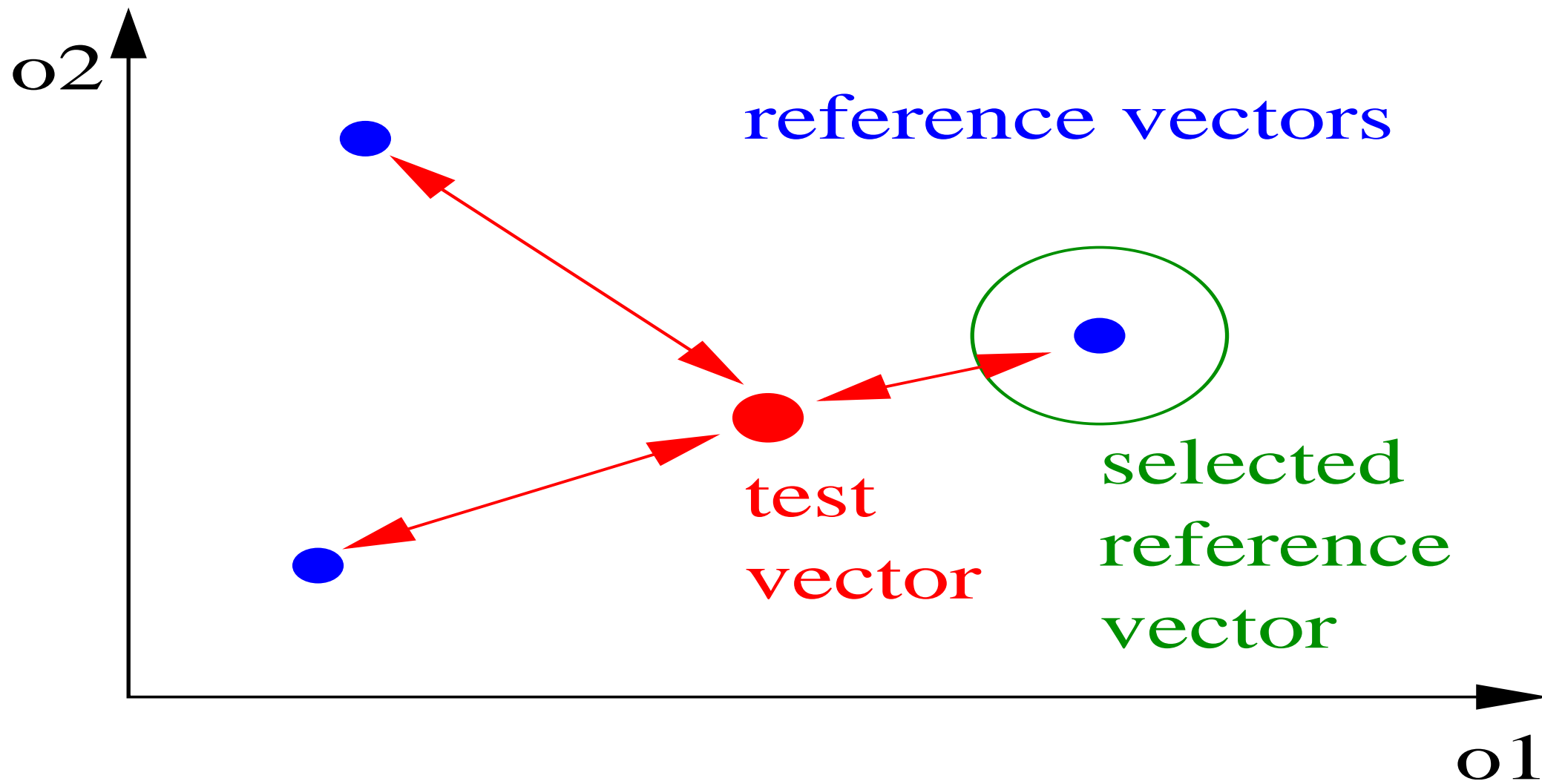
Plán

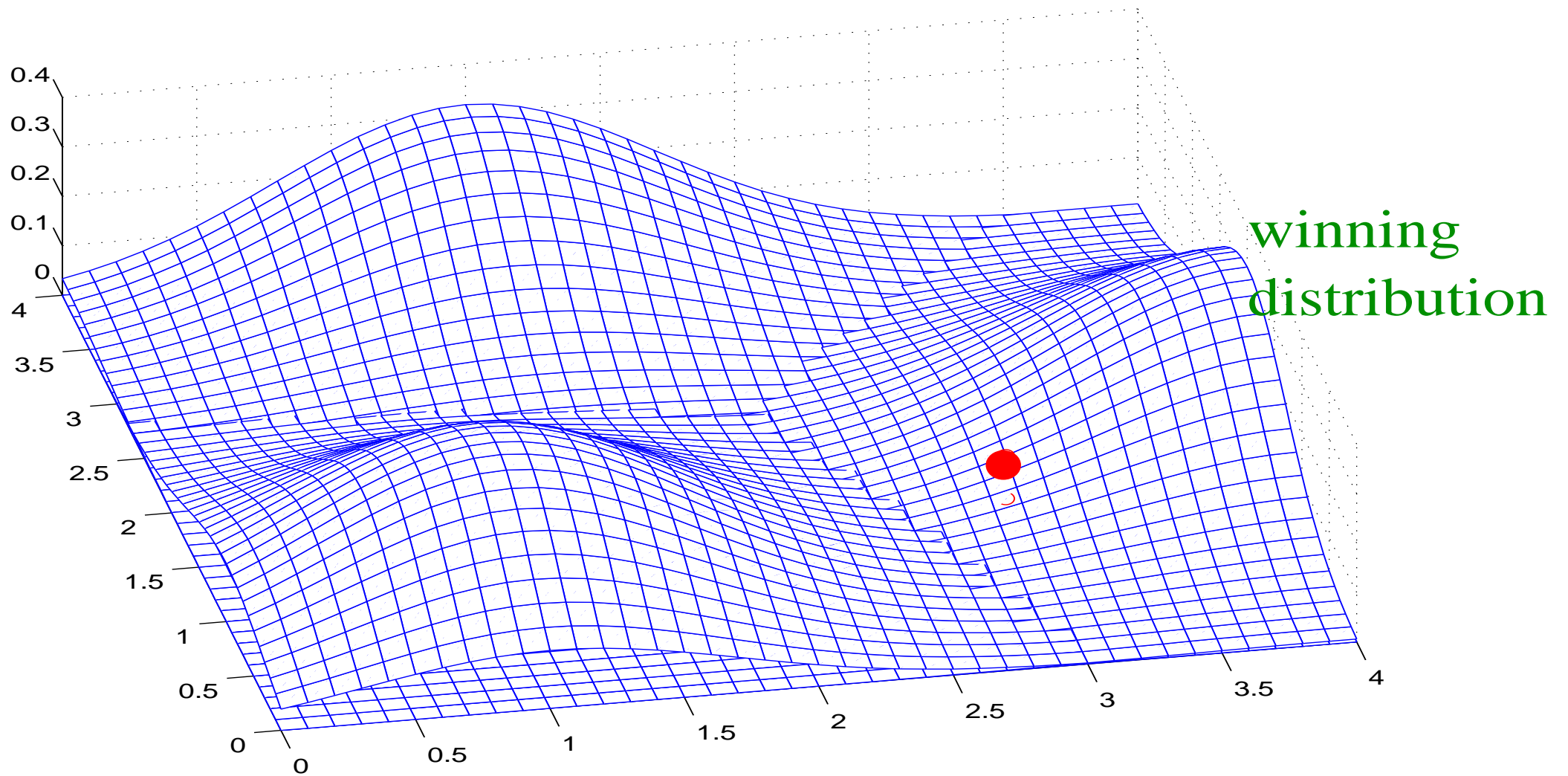
- Opakování – variabilita v řeči.
- Statistický přístup.
- Model a jeho parametry.
- Pravděpodobnost vyslání matice \mathbf{O} modelem M .
- Trénování parametrů.
- Rozpoznávání.
- Viterbi a Pivo-passing.

Opakování – jak na variabilitu v řeči

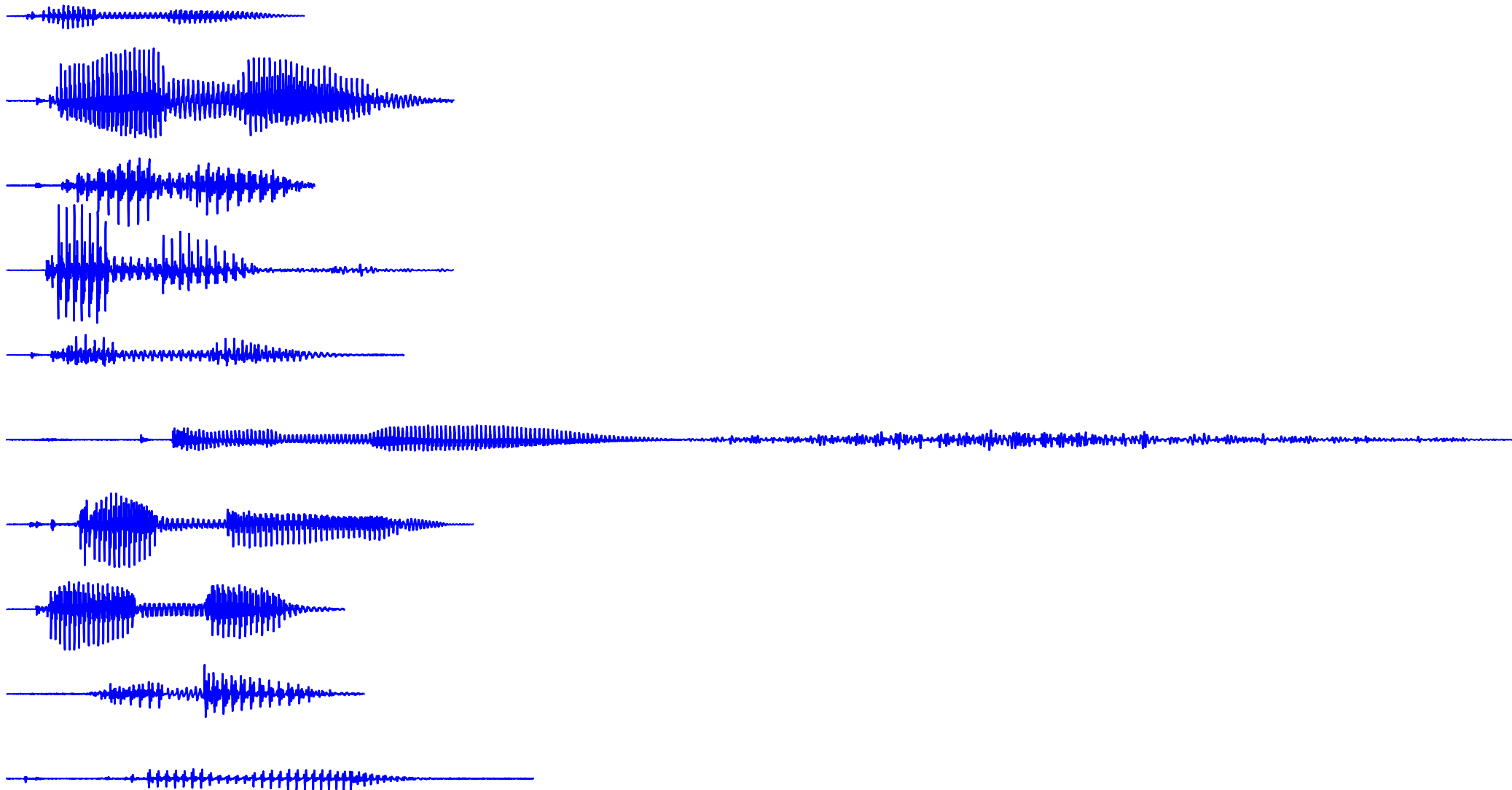
prostor parametrů - člověk nikdy neřekne jednu věc stejně \Rightarrow vektory parametrů se **vždy liší**.

1. Měření vzdálenosti mezi vektory.
2. Statistické modelování.

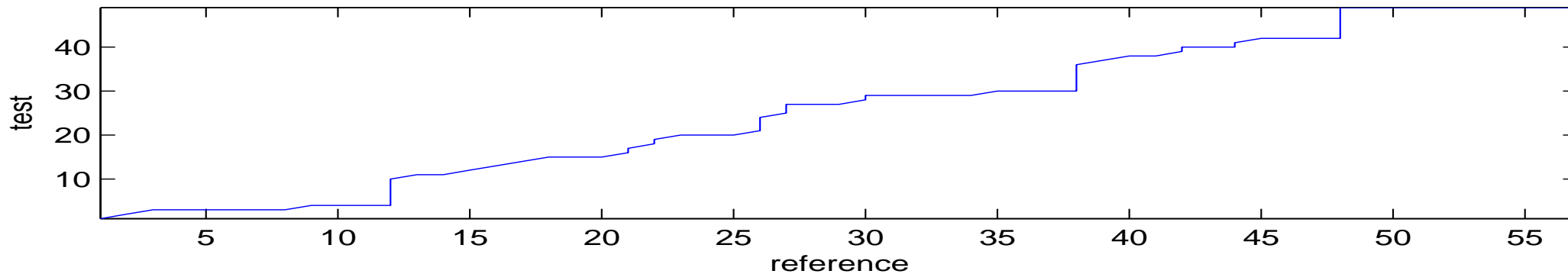
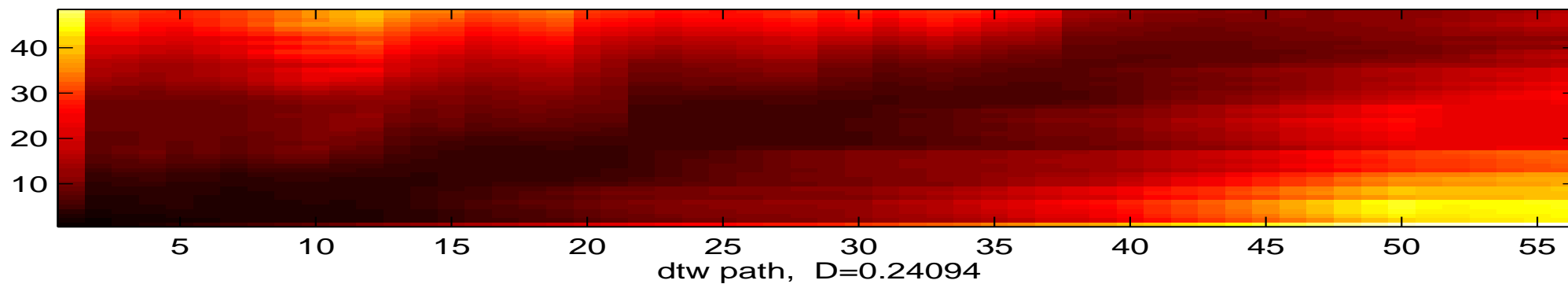
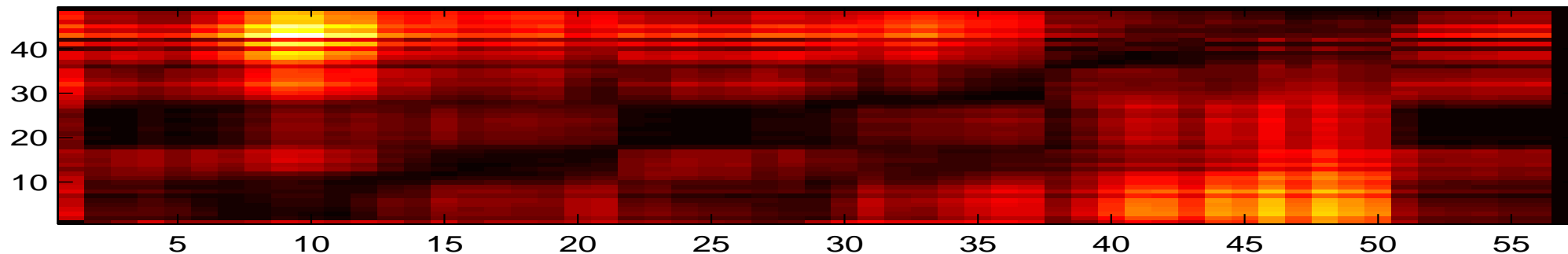




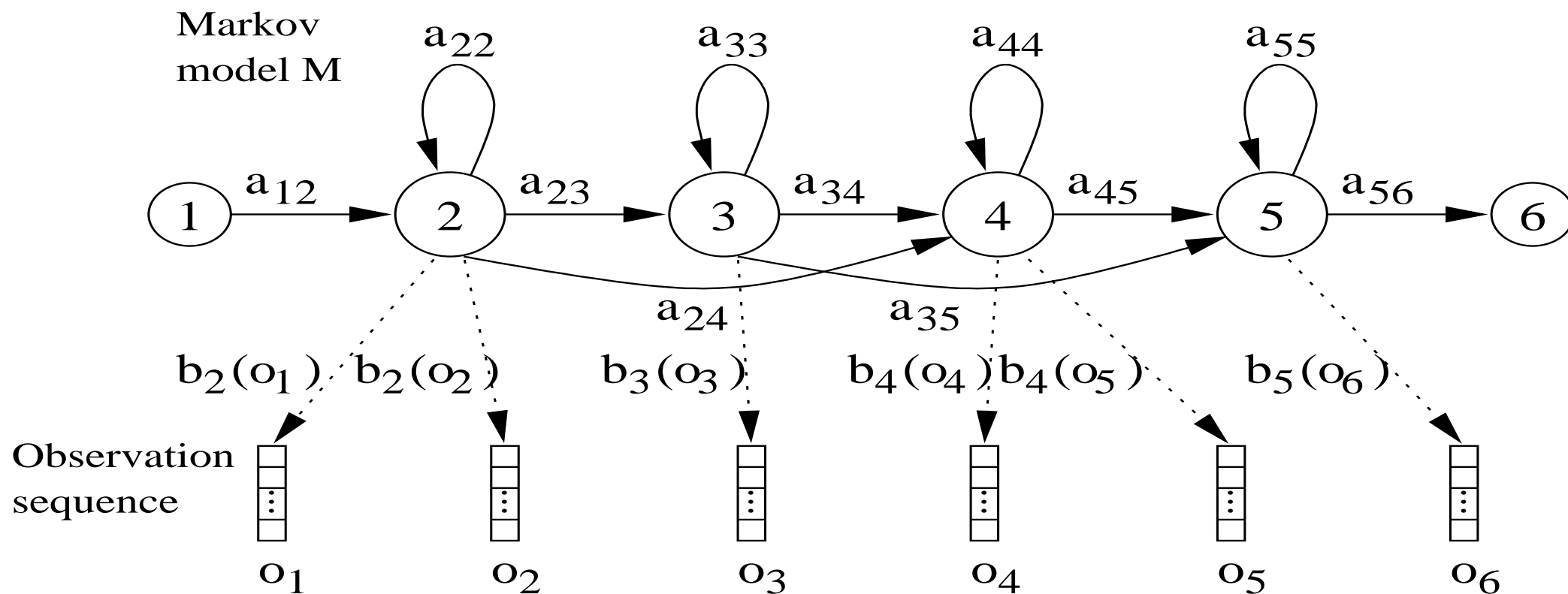
časování – lidé nikdy neřeknou jednu věc se stejným časováním.



Časování č.1 - Dynamické borcení času - cesta

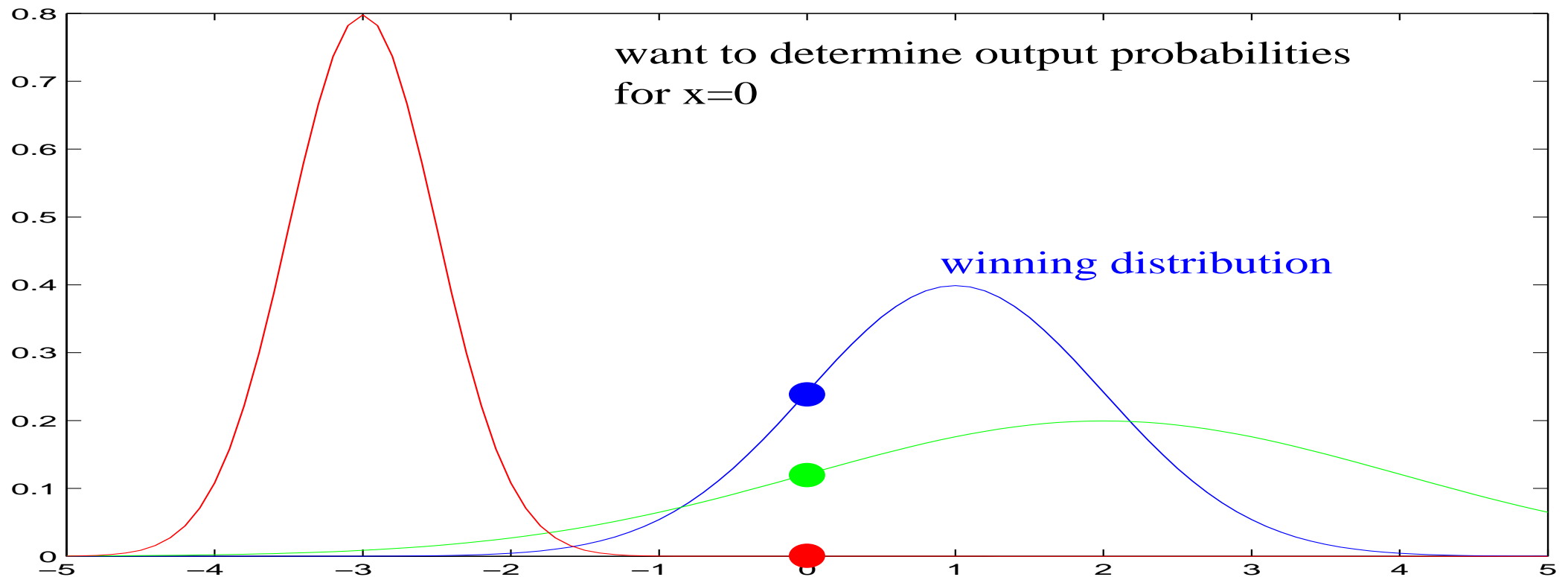


Časování č.2 - Skryté Markovovy modely - sekvence stavů



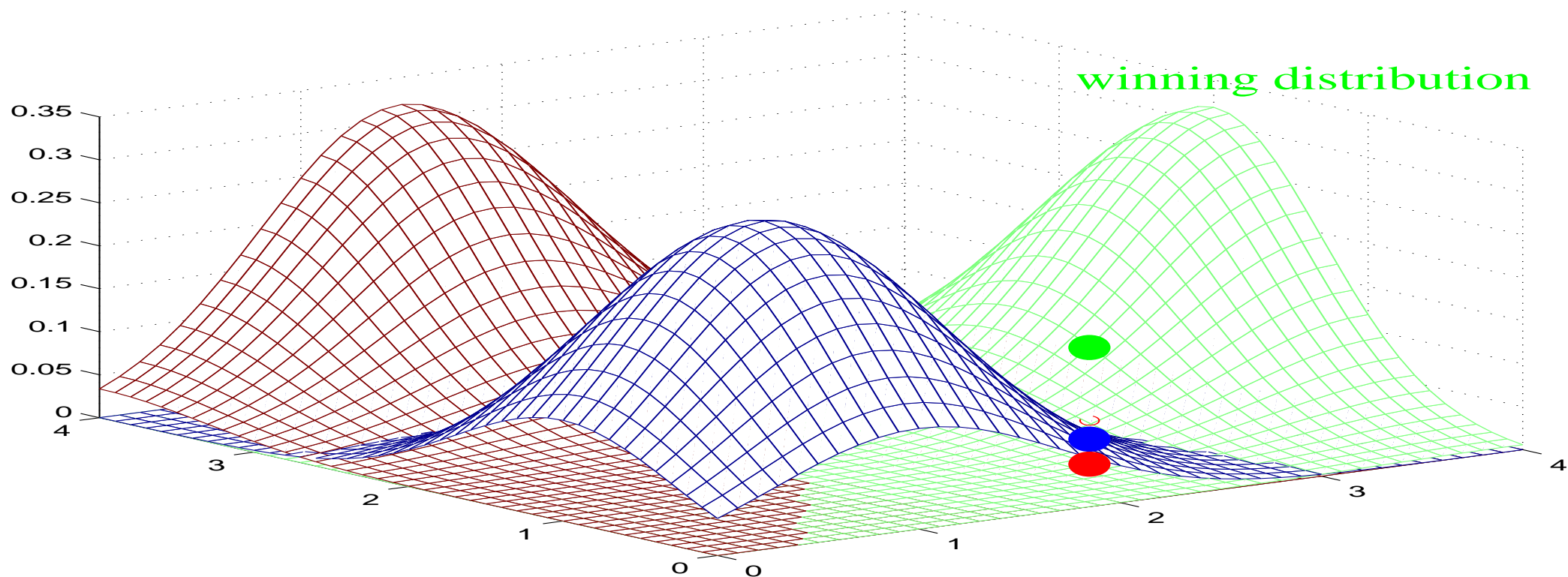
Skryté Markovovy modely (Hidden Markov Models) - HMM

Kdyby byla slova reprezentována jedním **skalárem** a slova byla representována Gaussovskými rozloženími. . .



- Hodnota $p(x)$ funkce hustoty rozdělení pravděpodobnosti **není pravděpodobnost** (pozor na statistiky se sekyrkou ☹). Pravděpodobnost je pouze $\int_a^b p(x)dx$. My tomu ale **budeme říkat pravděpodobnost**.
- Hodnotě $p(x)$ se někdy říká **vysílací pravděpodobnost** (emission probability), protože kdyby náhodný proces dokázal generovat hodnoty, vygeneroval by x s $p(x)$. Naše procesy nic generovat nebudou, ale budeme tomu “vysílací pravděpodobnost” říkat (odlišení od přechodových – viz dále).

...jenže slova nejsou representována skaláry, ale **vektory** \Rightarrow vícerozměrná Gaussovská rozložení.



V reálu jsou gaussovky mnoharozměrné (např. 39-rozměrné), to se špatně představuje a kreslí, ale můžeme si vždy udělat *průmět* do 1D, 2D nebo 3D.

...jenže slova nemají jen jeden vektor, ale více...

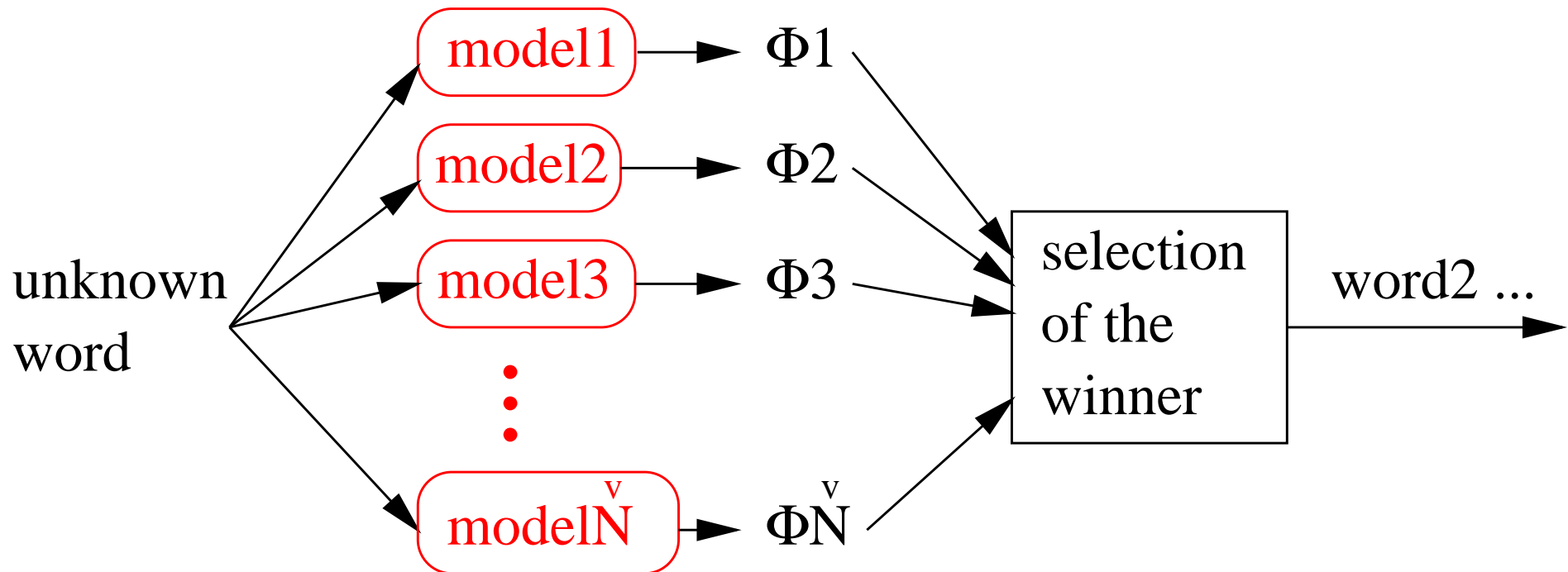
- Nápad 1: mít 1 gaussovku na slovo, postupně jí předkládat vektory a hodnoty sčítat
⇒ BLBÝ NÁPAD (“paří” = “řípa” ???)
- Nápad 2: každé slovo budeme muset representovat **sekvencí více Gaussovek**.
 - jedna nezávislá Gaussovka na každý vektor ? Aj aj, vektorů je pokaždé jiný počet !
 - **model, kde se Gaussovky budou moci opakovat !**

Modely, kde se Gaussovky mohou opakovat, se jmenují HMM

trochu přesněji – výklad pro **izolovaná slova**

⇒ Pro každé slovo, které budeme chtít rozpoznávat, budeme potřebovat jeden model.

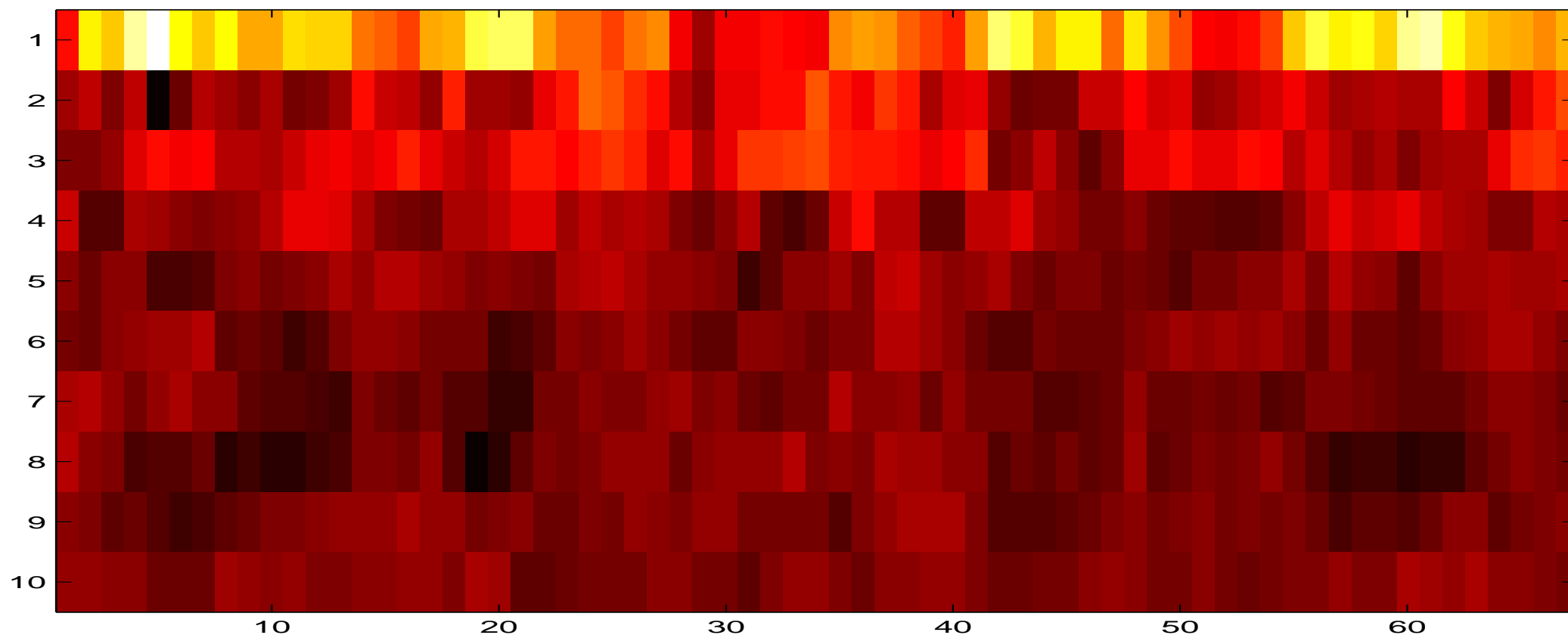
Viterbi
probabilities



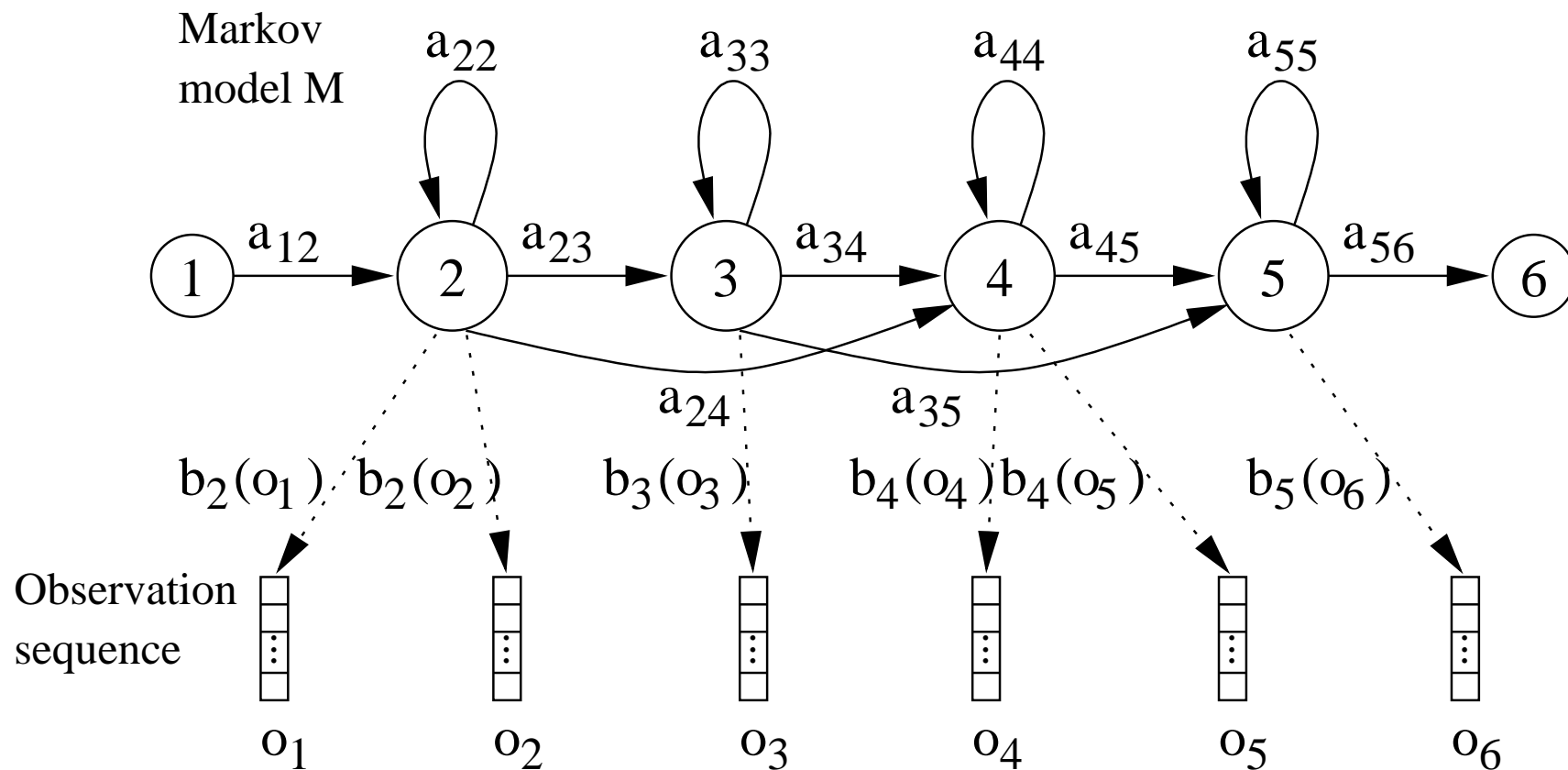
Teď už musí nastoupit trocha matematiky. Vše budeme ukazovat na **jednom modelu**, ale uvědomme si, že aby rozpoznávač rozpoznával, musí jich v něm být **několik** – pro každé slovo.

Vstupní sekvence vektorů

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



Konfigurace HMM



Přechodové pravděpodobnosti a_{ij}

Zde: pouze tři typy:

- $a_{i,i}$ pravděpodobnost setrvání ve stavu.
- $a_{i,i+1}$ pravděpodobnost přechodu do následujícího stavu.
- $a_{i,i+2}$ pravděpodobnost přeskočení následujícího stavu (*nepoužívá se*, pouze někdy pro modely “krátkého ticha”).

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

... matice má prvky pouze na diagonále a nad ní (musíme se v modelu zastavit nebo jít dál, nelze jít dozadu. DTW: cesta také nesměla jít dozadu!)

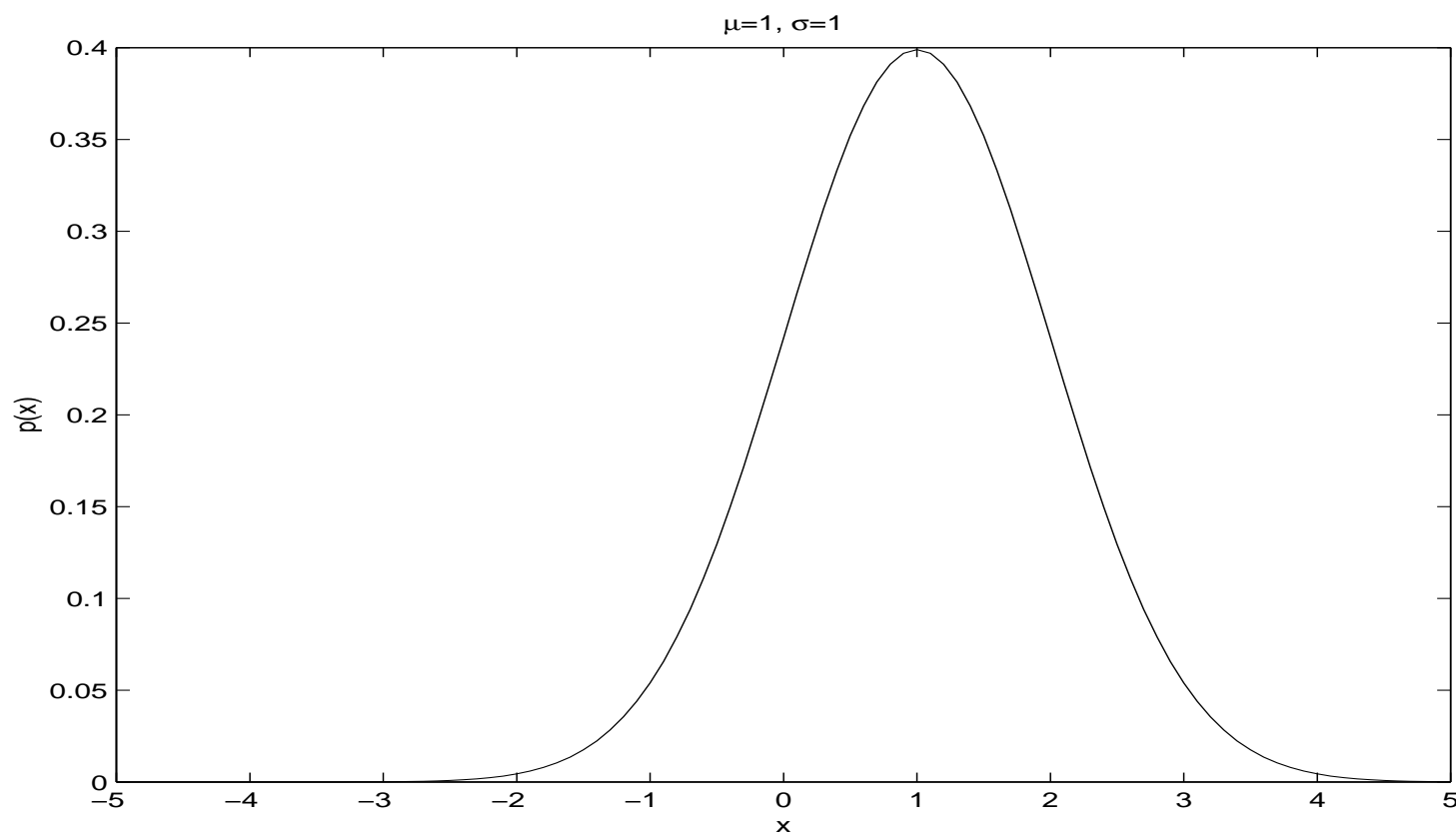
Funkce hustoty rozložení vysílacích pravděpodobností (pdfs)

“pravděpodobnost” (pozor na statistiky!) vyslání vektoru $\mathbf{o}(t)$ i -tým stavem modelu: $b_i[\mathbf{o}(t)]$.

- **diskrétní:** vektory jsou pomocí VQ “předkvantovány” na symboly, stavy pak obsahují tabulky vysílacích pravděpodobností jednotlivých symbolů – nebudeme probírat.
- spojité **funkce hustoty rozložení vysílacích pravděpodobností (continuous density – CDHMM)** Funkce je definována gaussovským rozdělením pravděpodobností nebo sumou několika rozdělení.

Kdyby měl vektor \mathbf{o} jen jeden prvek:

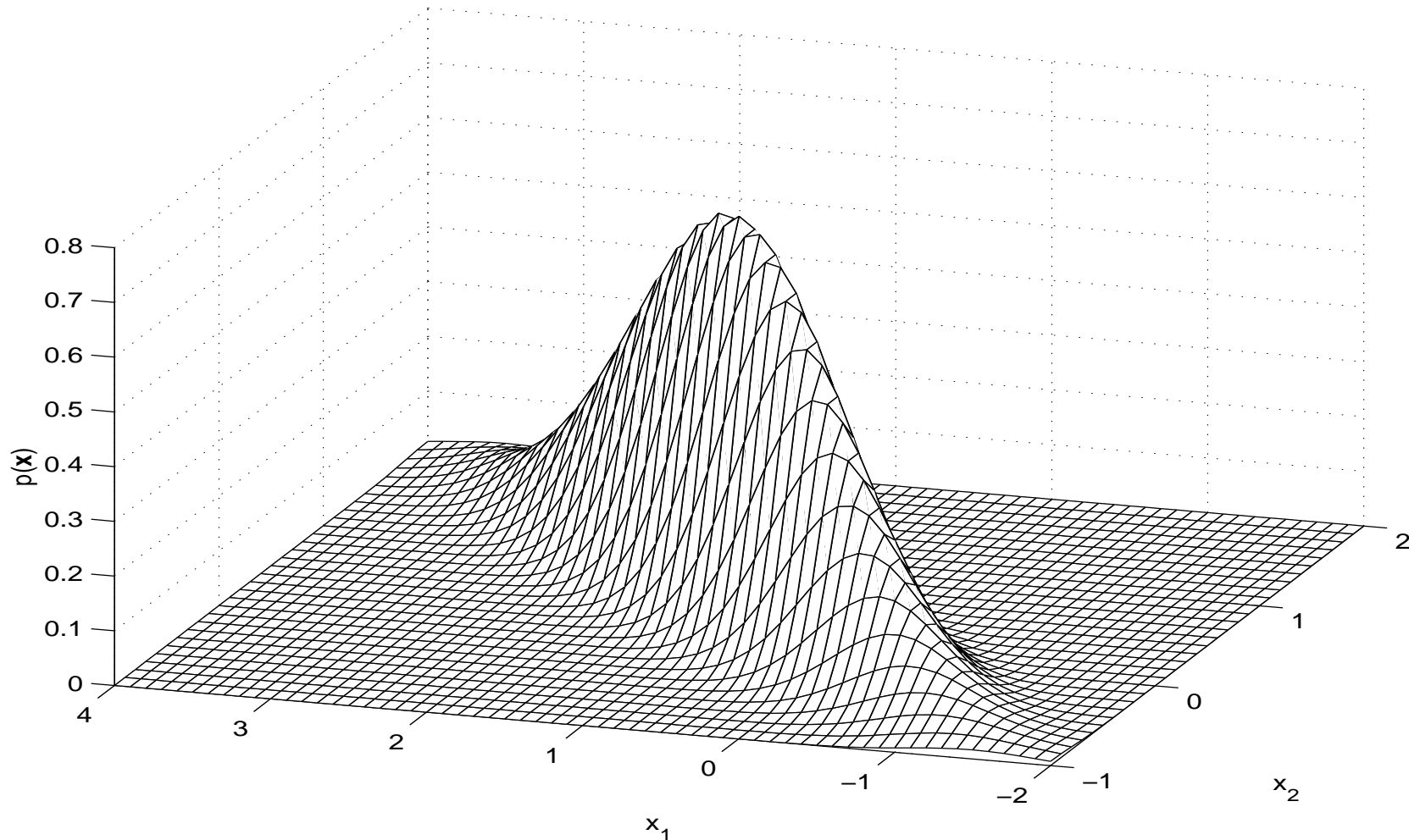
$$b_j[o(t)] = \mathcal{N}(o(t); \mu_j, \sigma_j) = \frac{1}{\sigma_j \sqrt{2\pi}} e^{-\frac{[o(t) - \mu_j]^2}{2\sigma_j^2}} \quad (3)$$



jenže vektor $\mathbf{o}(t)$ je P -rozměrný (klasicky 39):

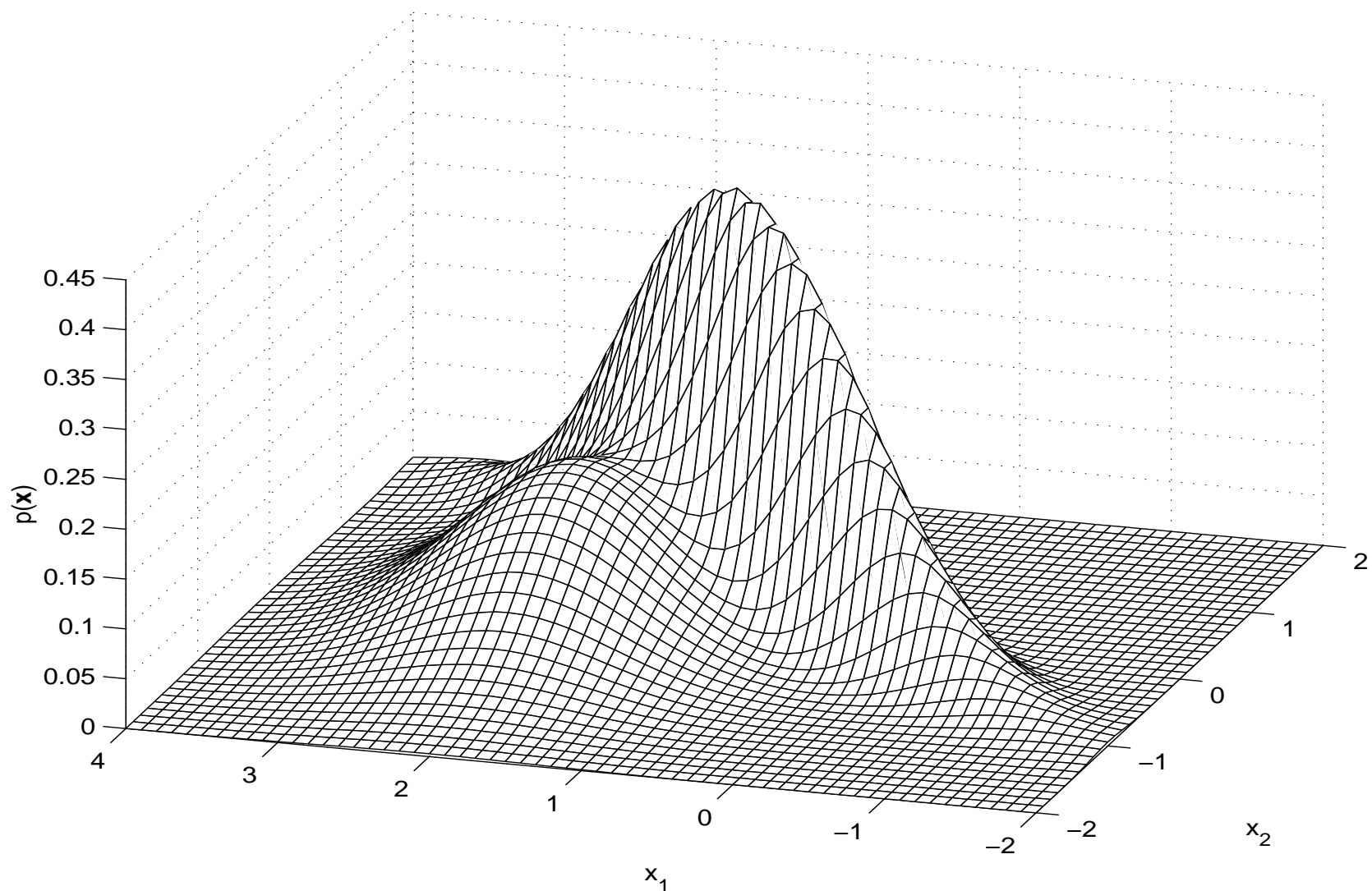
$$b_j[\mathbf{o}(t)] = \mathcal{N}(\mathbf{o}(t); \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \frac{1}{\sqrt{(2\pi)^P |\boldsymbol{\Sigma}_j|}} e^{-\frac{1}{2}(\mathbf{o}(t) - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{o}(t) - \boldsymbol{\mu}_j)}, \quad (4)$$

$$\boldsymbol{\mu} = [1; 0.5]; \boldsymbol{\Sigma} = [1 \ 0.5; 0.5 \ 0.3]$$



... definice směsí Gaussových rozdělení (bez rovnice):

$$\mu_1=[1;0.5]; \Sigma_1=[1 \ 0.5; 0.5 \ 0.3]; w_1=0.5; \mu_2=[2;0]; \Sigma_2=[0.5 \ 0; 0 \ 0.5]; w_2=0.5;$$



Zjednodušení rozdělení pravděpodobnosti:

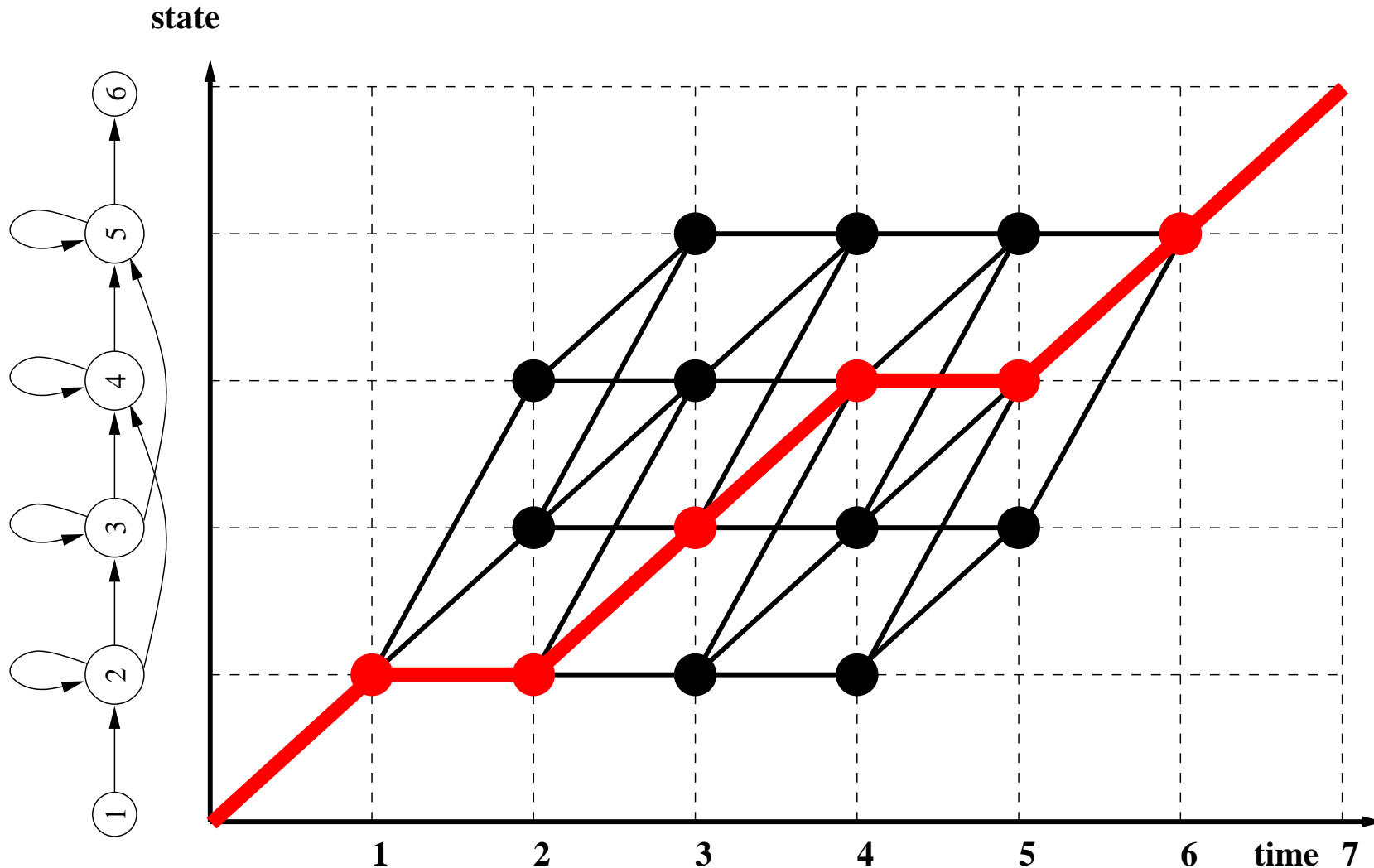
- pokud nejsou parametry korelovány (nebo doufáme, že nejsou), jejich kovarianční matice je diagonální \Rightarrow namísto $P \times P$ kovariančních koeficientů stačí odhadnout P směrodatných odchylek (rozptylů) \Rightarrow jednodušší modely, dostatek dat k odhadu, hodnota rozdělení dána pouze součinem 1-rozměrných Gauss. rozdělení (bez determinantu a inverze):

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

- sady parametrů nebo celé *stavy* mohou být sdílené mezi modely \Rightarrow méně parametrů, spolehlivější odhad, méně paměti.

Pravděpodobnosti, že model M generuje sekvenci O

Stavová sekvence: stavy přiřadíme vektorům, např: $X = [1\ 2\ 2\ 3\ 4\ 4\ 5\ 6]$.



pravděpodobnost generování \mathbf{O} po cestě X :

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

Jak definovat *jednu* pravděpodobnost generování sekvence vektorů modelem ?

a) BAUM-WELCH:

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

bereme sumu přes *všechny stavové sekvence* délky $T + 2$.

b) VITERBI:

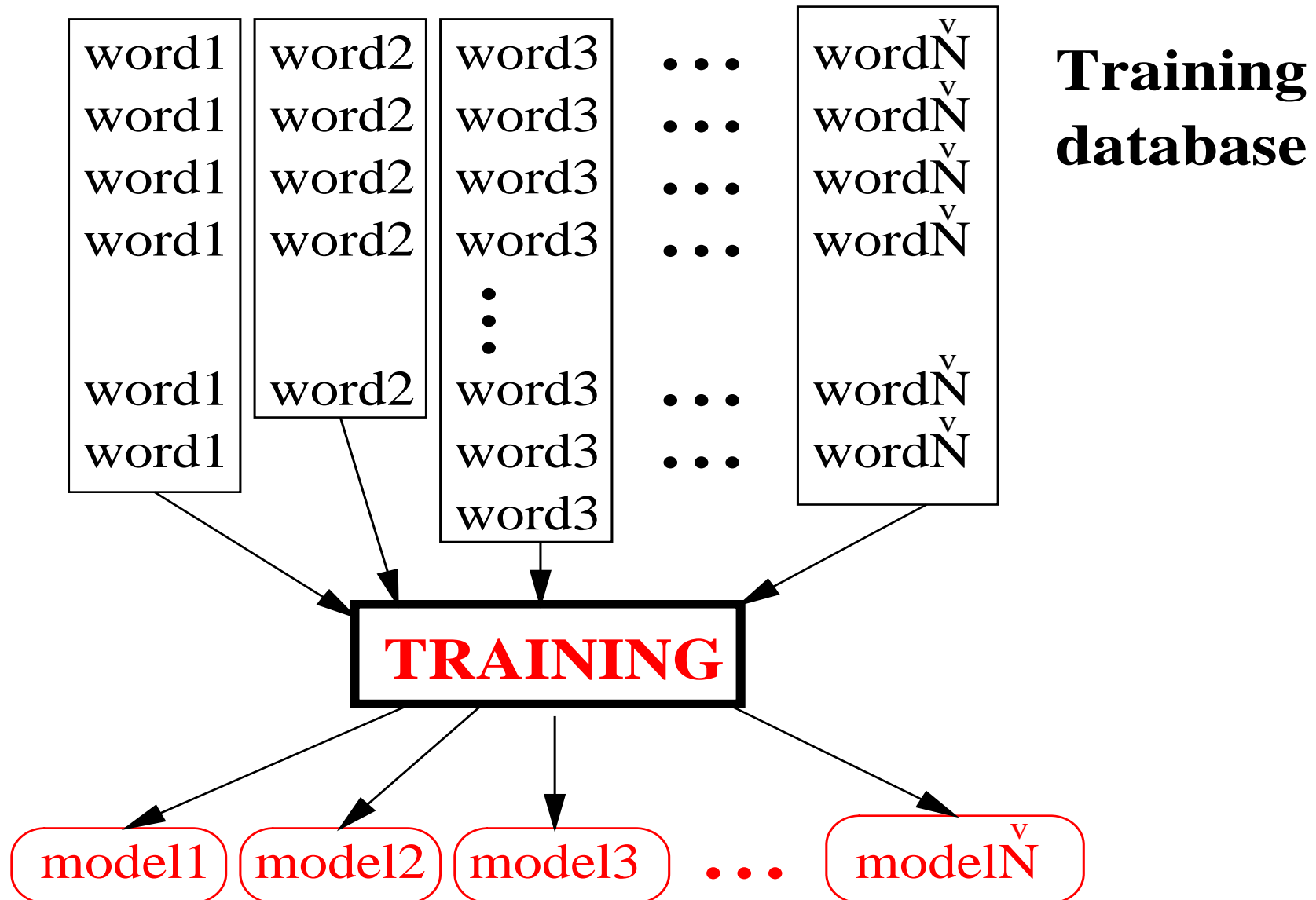
$$\mathcal{P}^*(\mathbf{O}|M) = \max_{\{X\}} \mathcal{P}(\mathbf{O}, X|M), \quad (8)$$

je pravděpodobnost optimální cesty.

Poznámky

1. U DTW jsme hledali *minimální vzdálenost*. Zde hledáme *maximální pravděpodobnost*, někdy též *věrohodnost* \mathcal{L} .
2. Rychlé algoritmy pro implementaci výpočtu jak BAUM-WELCHE, tak VITERBIHO: není nutné vyhodnocovat všechny možné stavové sekvence X .

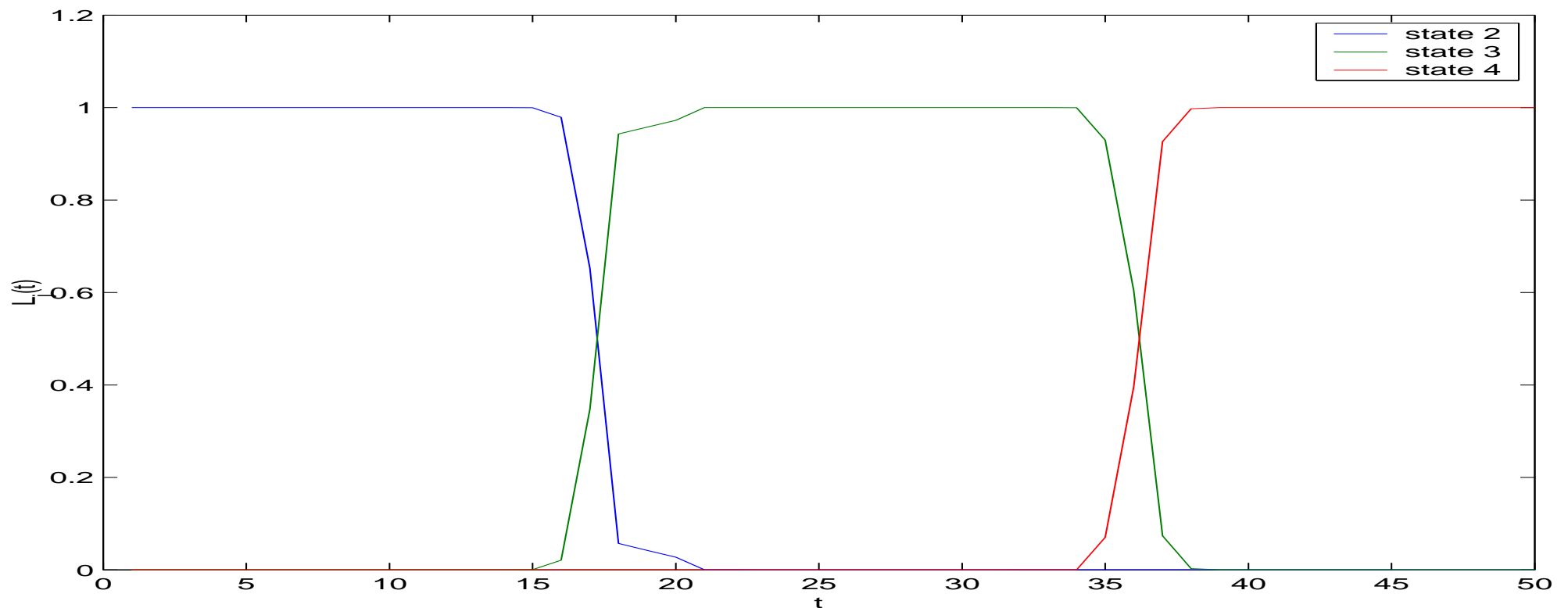
Trénování parametrů (nejsou v tabulkách :-) ...)



1. parametry modelu jsou **zhruba odhadnuty**:

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

2. vektory jsou **přiřazeny stavům**: “natvrdo” nebo “měkce” pomocí funkce $L_j(t)$ (state occupation function).



3. nové odhady parametrů:

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

... podobné vzorce pro odhad přechodových pravděpodobností a_{ij} .

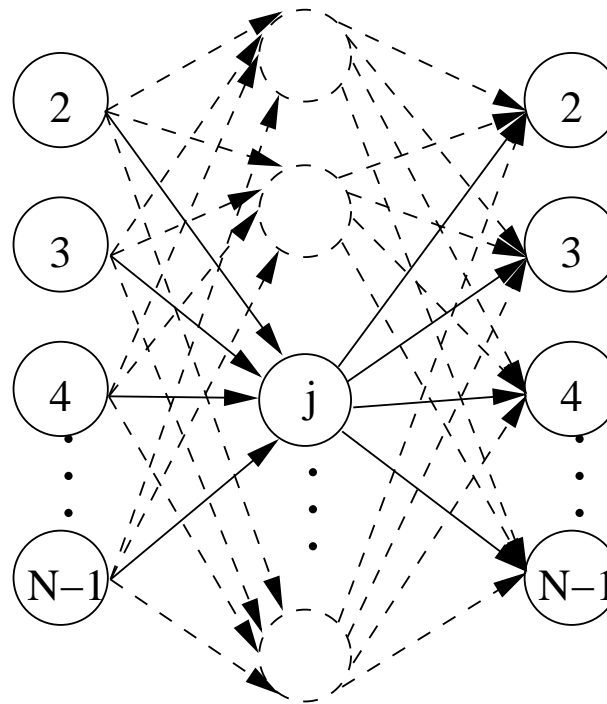
Kroky 2) a 3) se opakují: kritérium pro stop je fixní počet iterací nebo moment, kdy už se nezvětšují pravděpodobnosti.

- algoritmus se označuje jako EM (Expectation Maximization): dá se odvodit tak, že počítáme $\sum p(\text{každého vektoru} | \text{staré parametry}) \times \log p(\text{každého vektoru} | \text{nové parametry})$, což je vlastně očekávání (průměr na datech) celkové likelihood. Toto je kritériální funkce, kterou maximalizujeme \Rightarrow vede ke zvyšování likelihood.
- maximalizujeme věrohodnost, že data budou “vyslána” modelem \Rightarrow LM (likelihood maximization) - trochu problém, protože modely učíme tak, aby nejlépe reprezentovaly jednotlivá slova, ale ne aby mezi němi **diskriminovaly**.

Jak na state occupation functions

Pravděpodobnost $L_j(t)$ “bytí ve stavu j v čase t ” můžeme spočítat jako sumu všech cest, které v čase t projdou stavem j : $P(\mathbf{O}, x(t) = j | M)$

TIME: $t-1$ t $t+1$



Potřebujeme ale zajistit, aby to byly opravdové pravděpodobnosti, tedy:

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

jinými slovy: příspěvek jednoho vektoru všem stavům musí být přesně 100%. Musíme normalizovat sumou pravděpodobností (ne ! správně věrohodností !) všech cest, které projdou čímkoliv v čase t

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

... jenže to už jsou opravdu všechny cesty modelem, takže můžeme normalizovat BAUM-WELCHOVOU pravděpodobností:

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

Jak na $P(\mathbf{O}, x(t) = j | M)$

1. Výpočet částečných dopředných pravděpodobností (začal jsem na začátku modelu a v čase t jsem ve stavu j):

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

2. Výpočet částečných zpětných pravděpodobností (začal jsem na konci modelu a v čase t jsem ve stavu j):

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

3. state occupation function je pak definována:

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

... více v HTK Book.

Algoritmus trénování modelů

1. Alokuj akumulátor pro každý odhadovaný vektor/matice.
2. Spočítej dopředné a zpětné pravdě. $\alpha_j(t)$ a $\beta_j(t)$ pro všechny časy t a všechny stavy j .
Spočítej state occupation functions $L_j(t)$.
3. Ke každému akumulátoru přidej příspěvek od vektoru $\mathbf{o}(t)$ vážený příslušnou $L_j(t)$.
4. Použij konečnou hodnotu akumulátoru pro odhad vektoru/matice pomocí rovnice 10
— nesmíme zapomenout na normování výrazem $\sum_{t=1}^T L_j(t)$.
5. Pokud se hodnota $\mathcal{P}(\mathbf{O}|M)$ od minulé iterace podstatně nezměnila, stop, jinak Goto 1.

Rozpoznávání – Viterbiho algoritmus

- máme rozpoznat neznámou sekvenci \mathbf{O} .
- ve slovníku máme \check{N} slov $w_1 \dots w_{\check{N}}$.
- pro každé máme natrénovaný model $M_1 \dots M_{\check{N}}$.
- Otázka zní: “Který model by generoval \mathbf{O} s největší pravděpodobností ?”

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

použijeme VITERBIHO pravd. pro nejpravděpodobnější stavovou sekvenci:

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

takže:

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

Výpočet VITERBIHO pravdě.:

– podobné jako BAUM-WELCH.

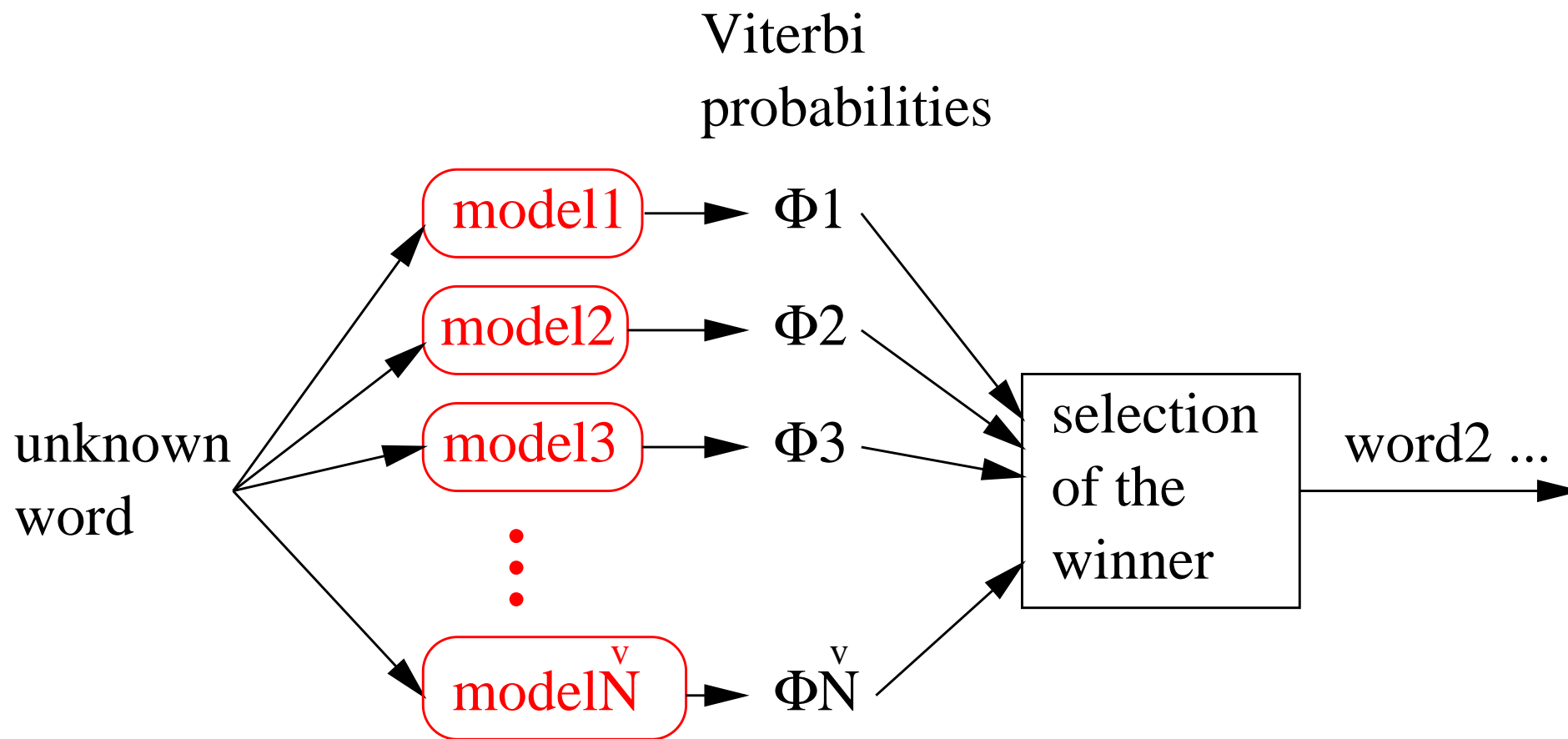
Částečná VITERBIHO pravdě.:

$$\Phi_j(t) = \mathcal{P}^*(\mathbf{o}(1) \dots \mathbf{o}(t), x(t) = j | M). \quad (20)$$

Pro j -tý stav a čas t . Žádaná VITERBIHO pravdě je:

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

Rozpoznávání izolovaných slov pomocí HMM



Viterbi jinak: Token passing – předávání püllitrů

každý stav modelu může obsahovat püllitr s pivem. Pracujeme s log pravděpodobnostmi, součiny \Rightarrow součty. Budeme dolévat pravděpodobnosti.

Inicialisace: vlož prázdný püllitr do každého vstupního stavu modelu (běžně pouze stav 1.).

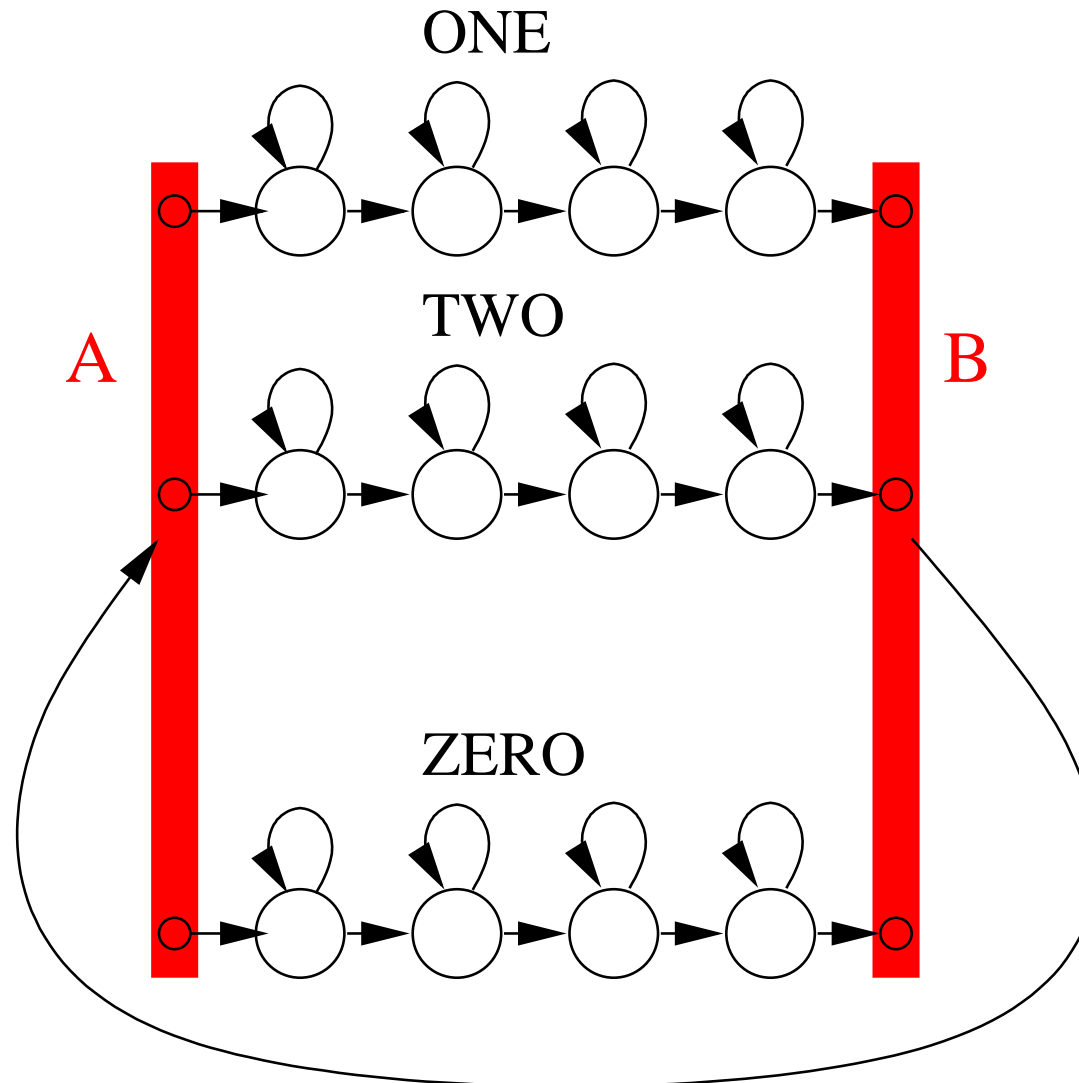
Iterace: pro časy $t = 1 \dots T$:

- v každém stavu i , který obsahuje püllitr, tento naklonuj a pošli kopii do všech napojených stavů j . Po cestě dolejš $\log a_{ij} + \log b_j[\mathbf{o}(t)]$.
- pokud se v nějakém stavu nachází více než jeden püllitr, nechej jen ten nejplnějš, zahodě ostatní.

Konec: ze všech stavů i spojených s výstupním stavem N , které obsahují püllitr, pošli ješ do N a dolejš $\log a_{iN}$. V posledním stavu N , vyber jen nejplnějš püllitr a zahodě ostatní. Hladinka piva ve stavu N odpovídá log Viterbiho pravděpodobnosti:
 $\log \mathcal{P}^*(\mathbf{O}|M)$.

Pivo passing pro rozpoznávání spojených slov

postavíme mega-model ze všech slov, “slepíme” první a poslední stavy.



Rozpoznávání – podobně jako pro izolovaná slova, musíme si však pamatovat slova, kterými optimální půllitr prošel.



Další čtení

- HTK Book - <http://htk.eng.cam.ac.uk/>
- Gold and Morgan: v knihovně FIT.
- Černocký - staré texty o HMM na <http://www.fit.vutbr.cz/~cernocky/oldspeech/>
- Počítačové cvičení HMM-HTK, staré poč. cvičení HMM-Matlab (na oldspeech).