Rozpoznávání řeči – úvod a DTW

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

FIT VUT Brno

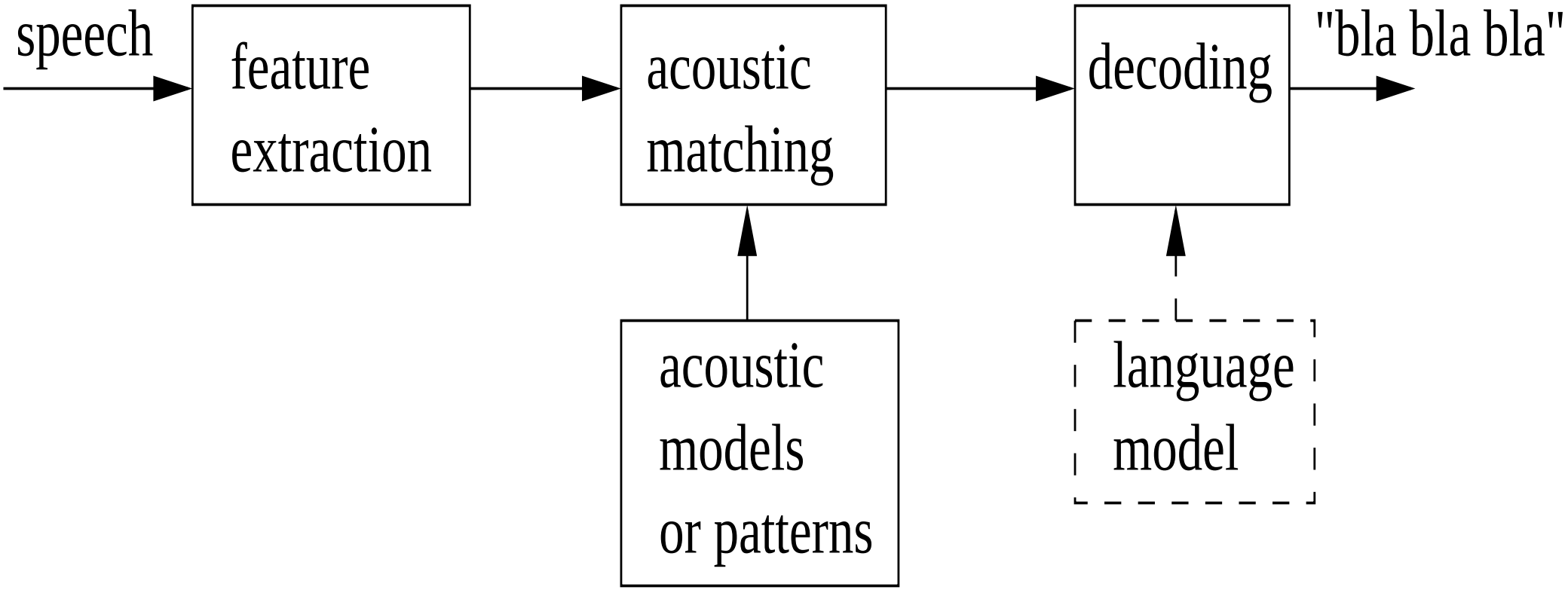
Rozpoznávání řeči (Speech Recognition)

Úkol: pro neznámý signál je nutné určit, co bylo řečeno

Klasifikace:

- izolovaná slova – ovládání mobilních telefonů hlasem. Potřebují voice activity detector nebo push-to-talk.
- spojená slova (omezený slovník) – např. číslovky při zadávání tlf. čísla nebo čísla kreditní karty. Rozpoznávání je většinou řízeno nějakou sítí nebo jednoduchou gramatikou.
- plynulá řeč s velkým slovníkem (large vocabulary continuous speech recognition LVCSR) – nejtěžší úkol, potřebuje informace o akustice, ale také o struktuře jazyka (jazykový model - language model) a výslovnostní slovník (pronunciation dictionary). Pracují s menšími jednotkami než se slovy (60 tisíc slov se nedá naučit...) - fonémy, kontextově závislé fonémy.

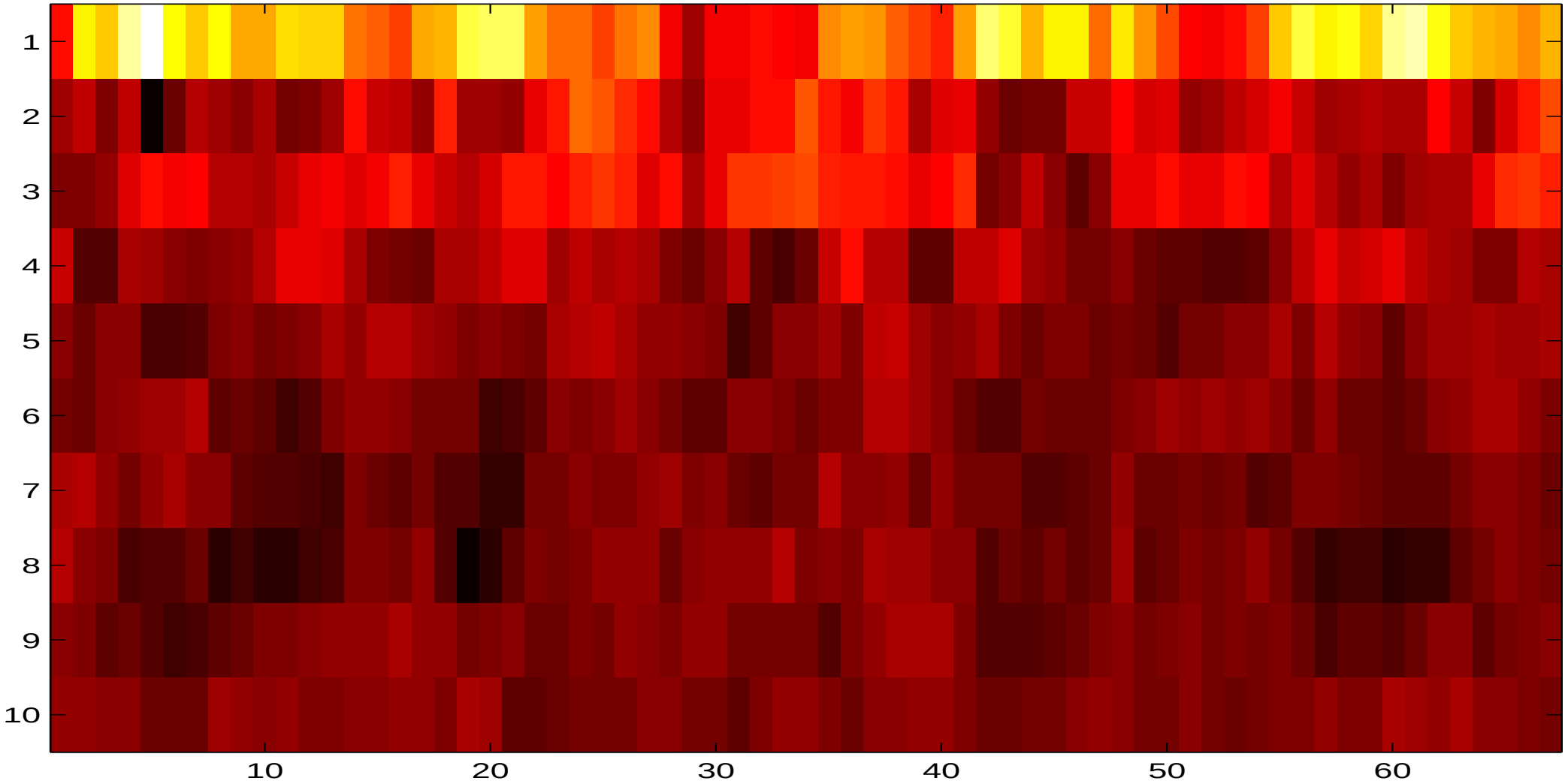
Struktura rozpoznávače



Parametrizace

- omezení množství dat.
- "odrušení" složek, které nás nezajímají (pitch, střední hodnota, fáze)
- většinou využívají spektrální analýzu (Mel-frekvenční cepstrální koeficienty) nebo LPC analýzu (LPC-cepstrum)
- v rámcích (kvazistacionarita)
- parametry musí být dobré pro rozpoznávač (např. rozdíl dvou vektorů by měl mít nějaký smysl, nebo nekorelovanost).
- samostatná přednáška !

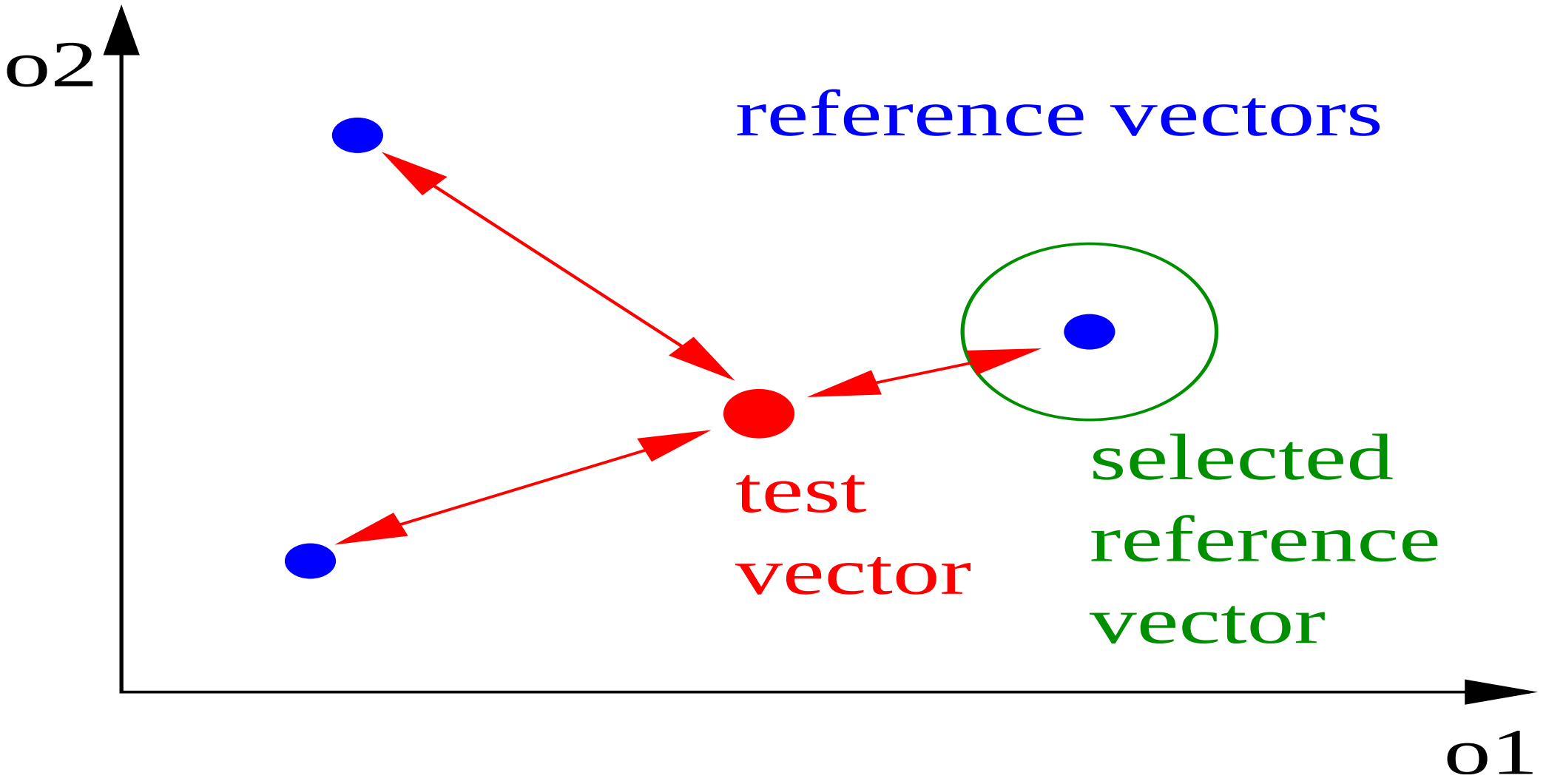
Výsledkem parametrizace je sekvence vektorů: $\mathbf{O} = [\mathbf{o}(1), \mathbf{o}(2), \ldots, \mathbf{o}(T)]$

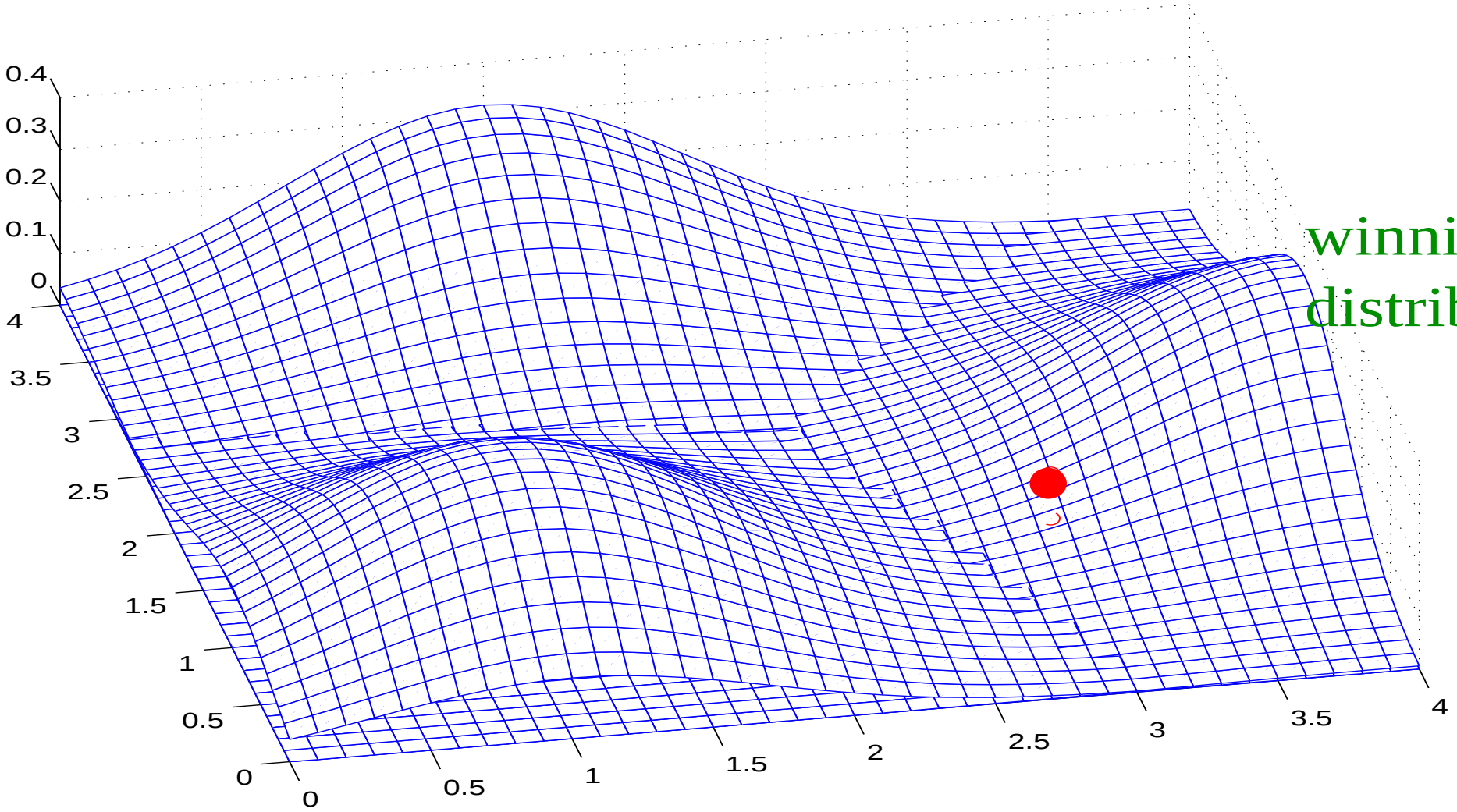


Acoustic matching — variabilita všude !!!

prostor parametrů - člověk nikdy neřekne jednu věc stejně $\Rightarrow$ vektory parametrů se **vždy liší**. Metody fungující pro text $\Rightarrow$ ne... Jak na to ?

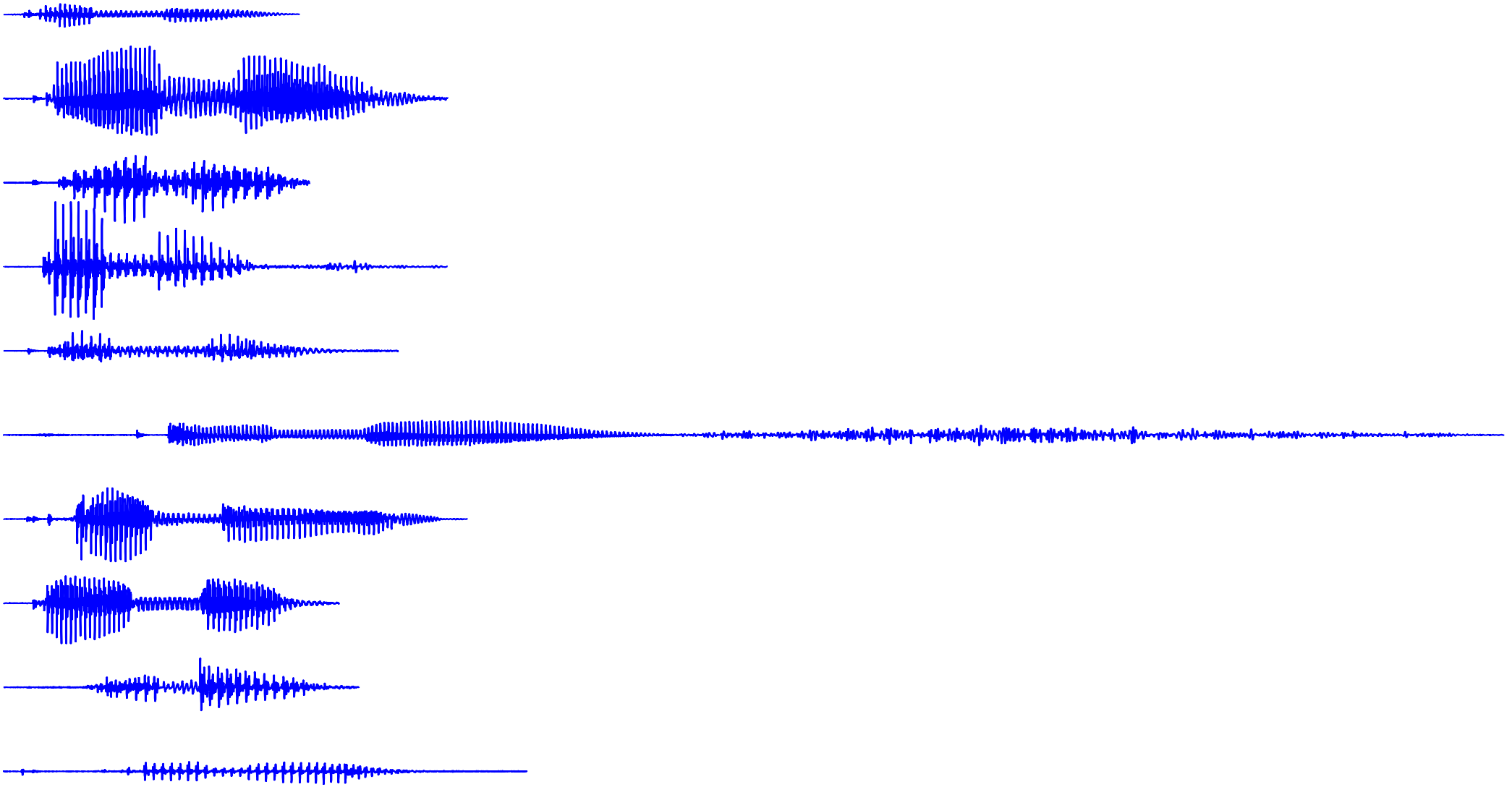
1. Měření vzdálenosti mezi vektory.
2. Statistické modelování.


o2
reference vectors
test
vector
selected
reference
vector
o1

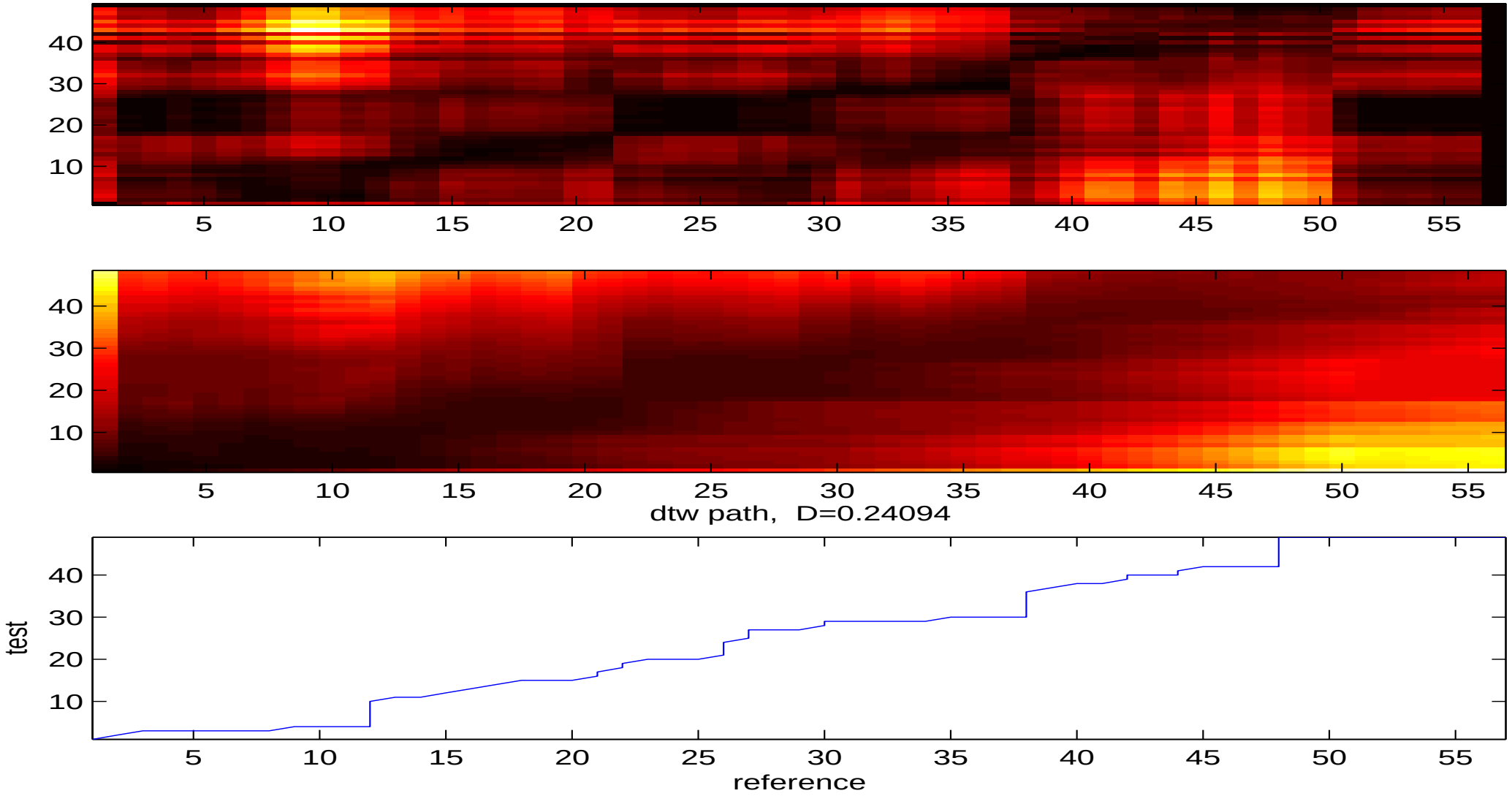


winning distribution

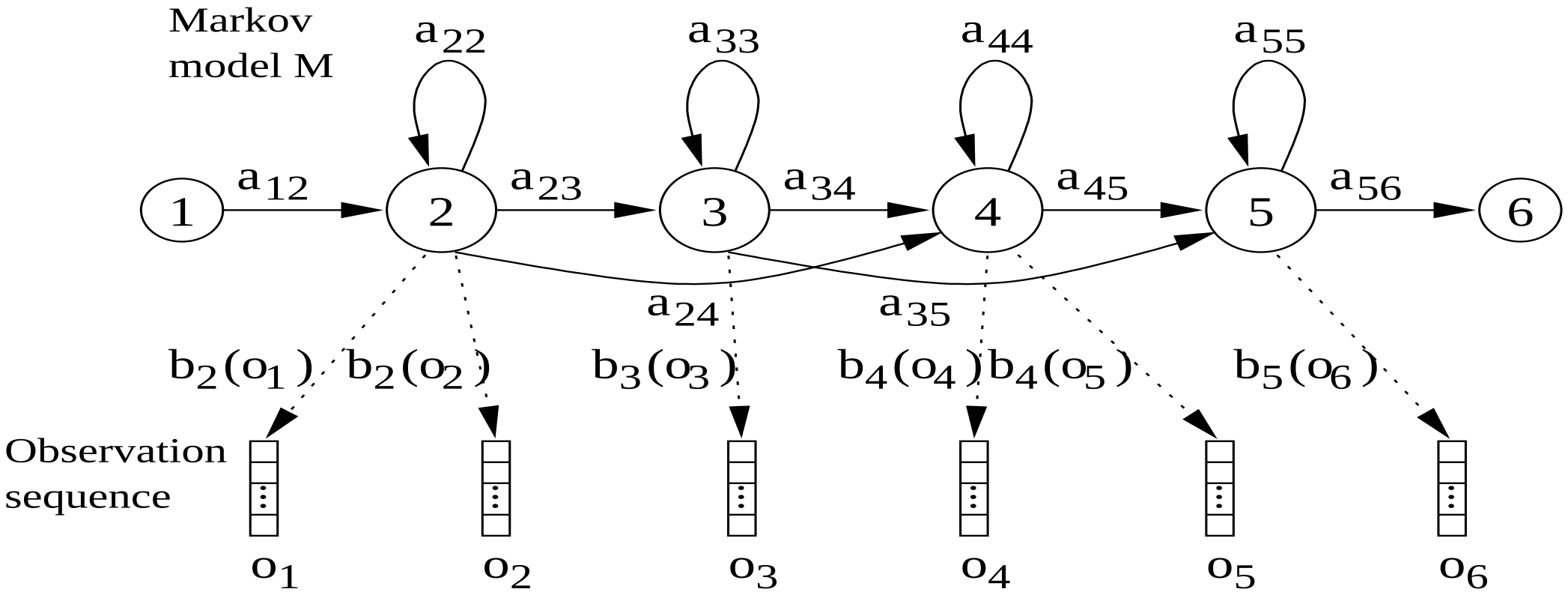
časování – lidé nikdy neřeknou jednu věc se stejným časováním.



Časování č.1 - Dynamické borcení času - cesta



Časování č.2 - Sktyré Markovovy modely - sekvence stavů



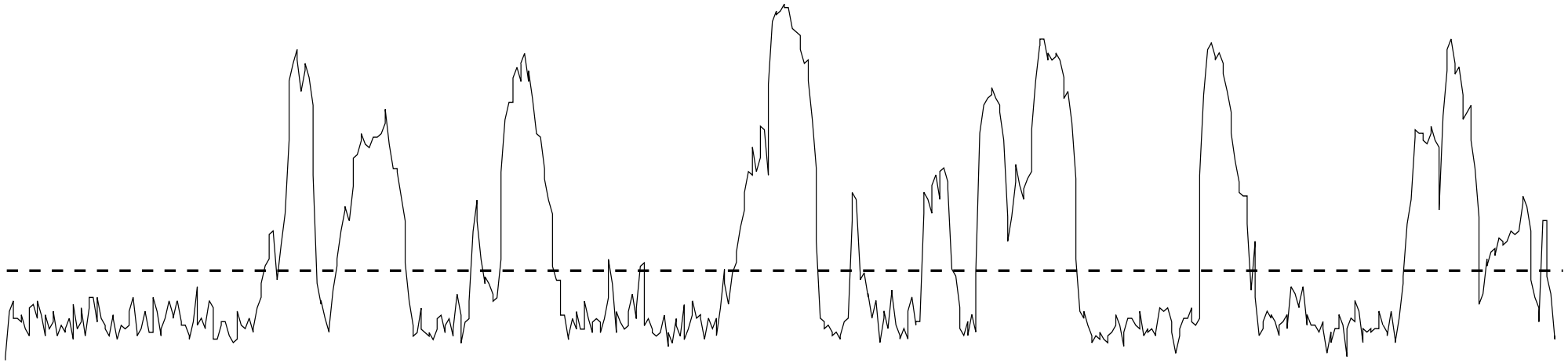
Dekódování

- izolovaná slova: velmi jednoduché (jen výběr maxima pravděpodobnosti nebo minima vzdálenosti).
- LVCSR: velmi složité (akustické modely (trifóny), language model, výslovnostní slovník) - Viterbiho algoritmus, A*-search, best-first decoding, konečné stavové automaty (!), omezení prohledávacího prostoru (beam-search), atd atd.

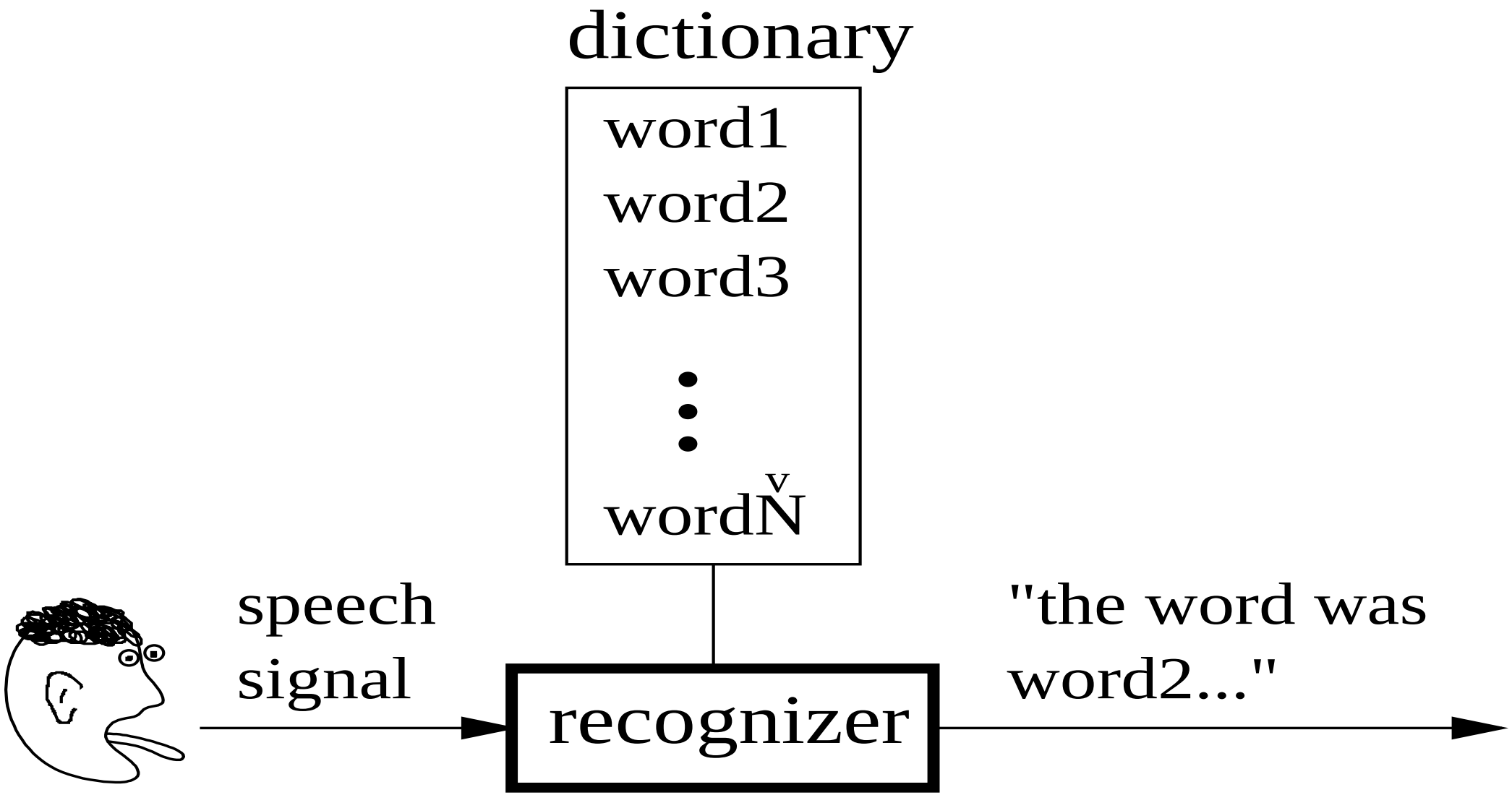
ROZPOZNÁVÁNÍ IZOLOVANÝCH SLOV POMOCÍ DTW

Určení izolovaných slov:

- push-to-talk...
- detekce řečové aktivity – např. detektor založený na energii:



Jak to má fungovat:



- ve slovníku jsou **referenční** matice parametrů pro slova, která chceme rozpoznávat:

$$\mathbf{R}_1 \dots \mathbf{R}_{\check{N}}$$

- na vstup rozpoznávače přijde **testovací** matice parametrů $\mathbf{O}$
- chceme určit, ke kterému referenčnímu slovu test patří.

Kdyby měla slova jen jeden vektor, bylo by to jednoduché:

$$d(\mathbf{o}, \mathbf{r}_i) = \sqrt{\sum_{k=1}^{P} |o(k) - r_i(k)|^2}.$$

Pak by se vybrala minimální vzdálenost.

Jenže slova jeden vektor **nemají**: Potřebujeme určit *vzdálenost* (či *podobnost*) referenční sekvence vektorů o délce R:

$$\mathbf{R} = [\mathbf{r}(1), \ldots, \mathbf{r}(R)] \tag{1}$$

a testovací sekvence vektorů o délce T:

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

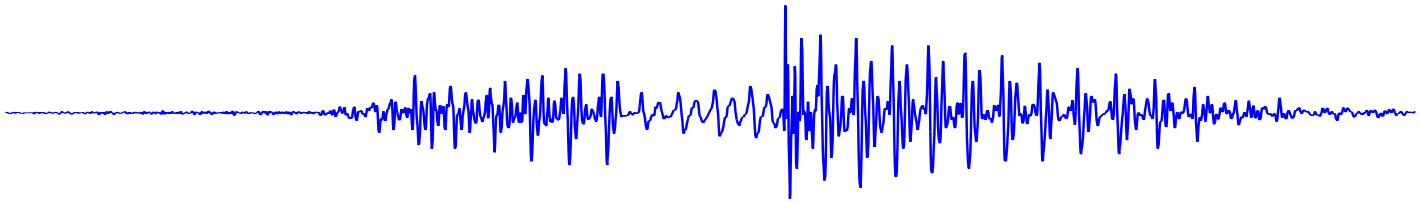
Sečítání vzdáleností jednotlivých vektorů přes celé slovo ? Jakých ? Slova **nejsou nikdy stejně dlouhá** $R \neq T$.

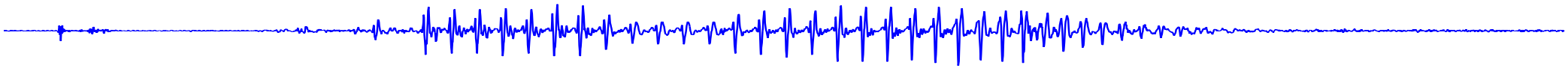
Lineární srovnání

$$D(\mathbf{O}, \mathbf{R}) = \sum_{i=1}^{R} d[\mathbf{o}(w(i)), \mathbf{r}(i)] \tag{3}$$

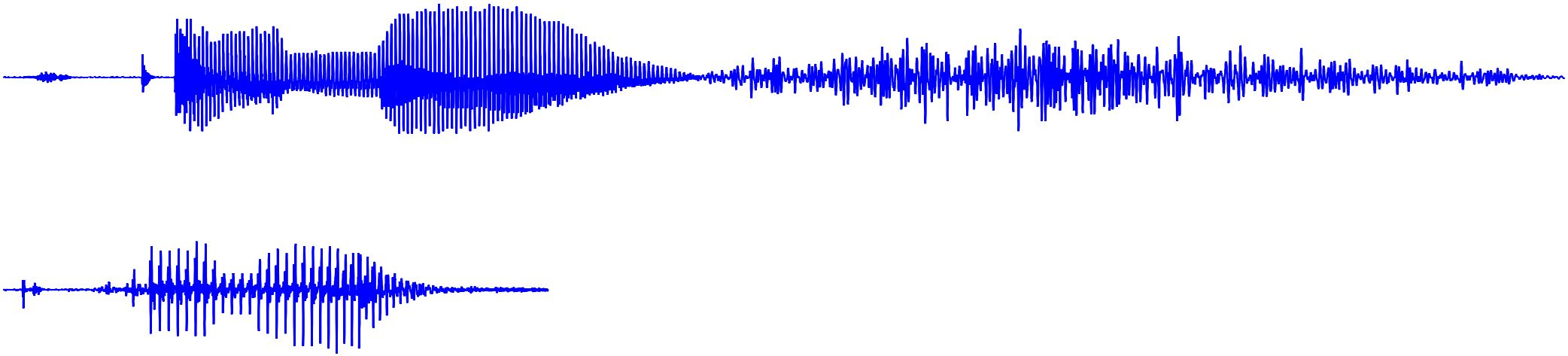
kde $w(i)$ je definována tak, aby srovnání bylo lineární.

Jenže to nebude fungovat. . . tady ještě ano:





. . . ale tady už ne (chyba VAD):

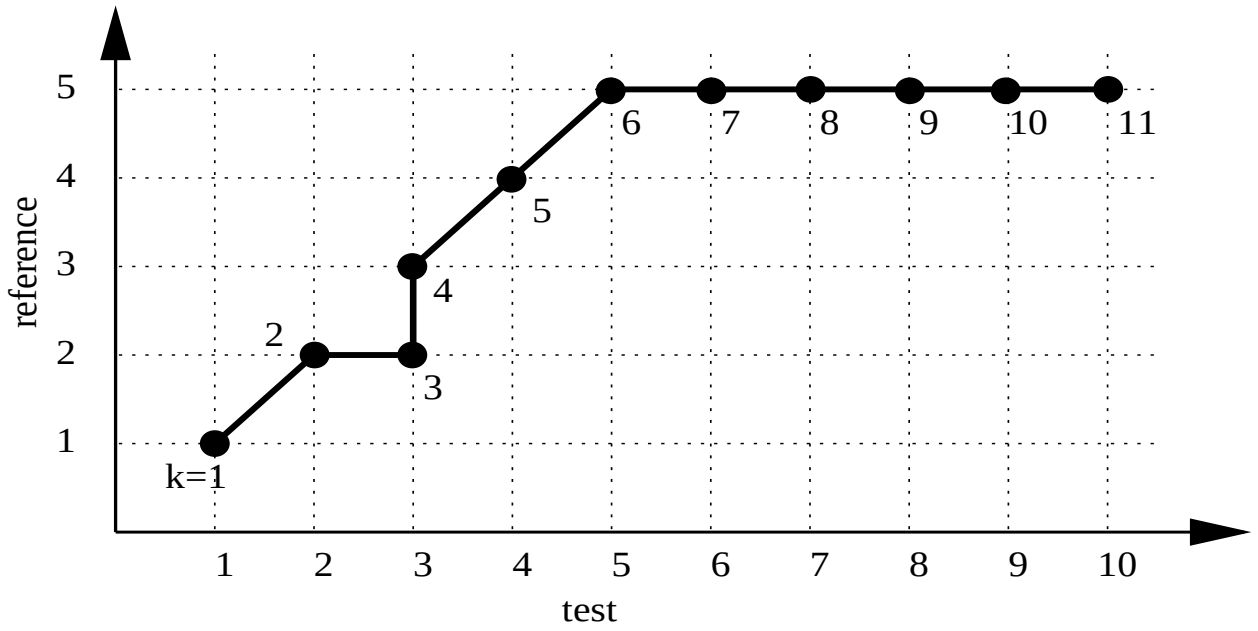


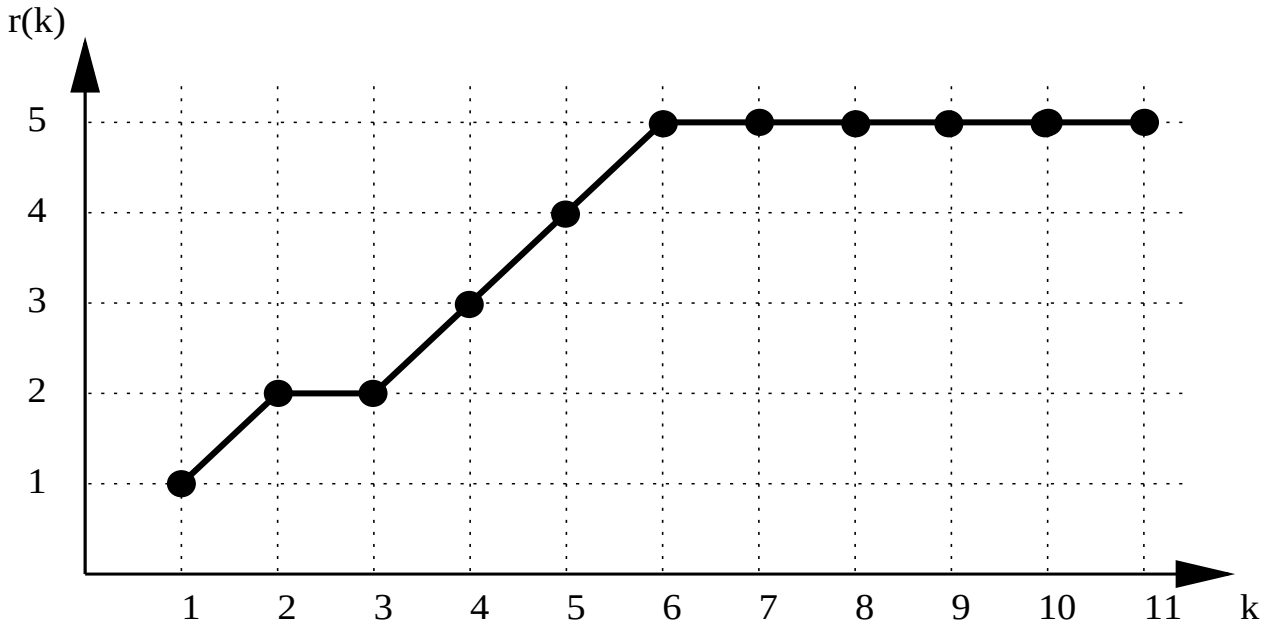
Je ovšem podstatně lepší když je srovnání *řízeno* přímo vzdálenostmi jednotlivých vektorů $\Rightarrow$ **Dynamické borcení času**.

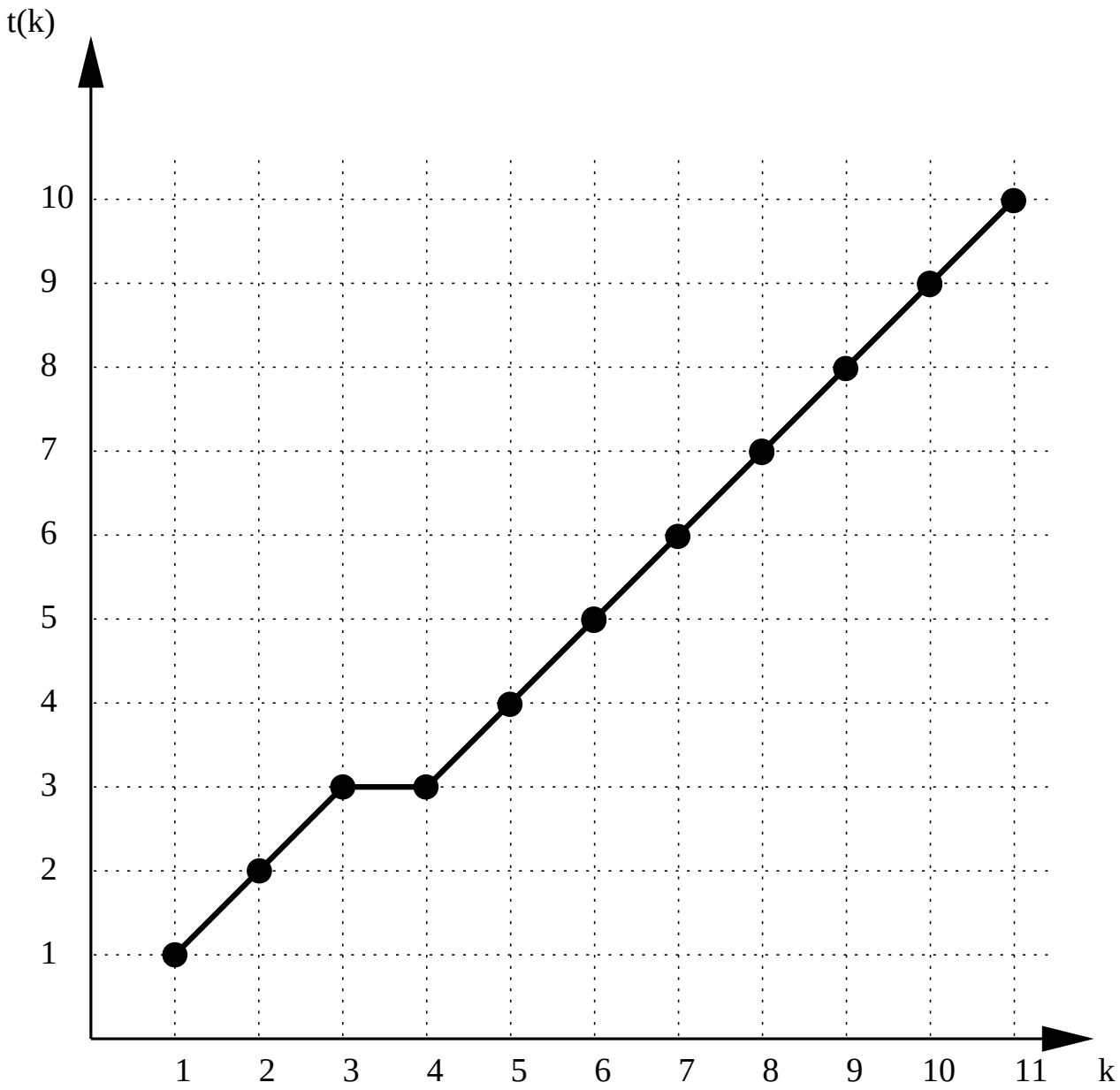
Definujeme obecnou časovou proměnnou k a a zavedeme *dvě* transformační funkce:

- $r(k)$ pro referenční sekvenci.
- $t(k)$ pro testovací sekvenci.

Pak můžeme srovnání jednotlivých vektorů zakreslit pomocí **cesty**: Počet kroků cesty označíme jako K. Referenci budeme zobrazovat na svislé ose, test na vodorovné. Z této cesty můžeme odvodit průběh funkcí $r(k)$ a $t(k)$, které "krokují" jednotlivé sekvence


reference
test
k=1
1
2
3
4
5
6
7
8
9
10
11


r(k)
k
1
2
3
4
5
6
7
8
9
10
11


t(k)
10
9
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
9
10
11
k

Cesta C je jednoznačně dána svou délkou K_C a průběhem funkcí $r_C(k)$ a $t_C(k)$. Pro tuto cestu je vzdálenost sekvencí $\mathbf{O}$ a $\mathbf{R}$ dána jako:

$$D_C(\mathbf{O}, \mathbf{R}) = \frac{\sum_{k=1}^{K_C} d[\mathbf{o}(t_C(k)), \mathbf{r}(r_C(k))] W_C(k)}{N_C} \tag{4}$$

kde $d[\mathbf{o}(\cdot), \mathbf{r}(\cdot)]$ je vzdálenost dvou vektorů, $W_C(k)$ je váha odpovídající k-tému kroku cesty a N_C je normalisační faktor závislý na vahách.

Vzdálenost sekvencí $\mathbf{O}$ a $\mathbf{R}$ je dána jako *minimální vzdálenost* přes soubor všech možných cest (všechny možné délky, všechny možné průběhy):

$$D(\mathbf{O}, \mathbf{R}) = \min_{\{C\}} D_C(\mathbf{O}, \mathbf{R}). \tag{5}$$

Je třeba vyřešit 3 věci:

1. přípustné průběhy funkcí $r(k)$ a $t(k)$. Není možné, aby se cesta vracela zpět, "skákala" přes několik vektorů, atd.
2. definovat váhovací funkci a normalisační faktor.
3. vyrobit algoritmus, který vypočítá $D(\mathbf{O}, \mathbf{R})$ co nejrychleji.

Omezení cesty

1. **Počáteční a koncové body**

$$\left.\begin{array}{l} r(1)=1 \\ t(1)=1 \end{array}\right\} \text{začátek} \qquad \left.\begin{array}{l} r(K)=R \\ t(K)=T \end{array}\right\} \text{konec} \tag{6}$$

2. **Lokální souvislost a lokální strmost**

$$\begin{aligned} 0 \leq r(k) - r(k-1) \leq R^\star \\ 0 \leq t(k) - t(k-1) \leq T^\star \end{aligned} \tag{7}$$

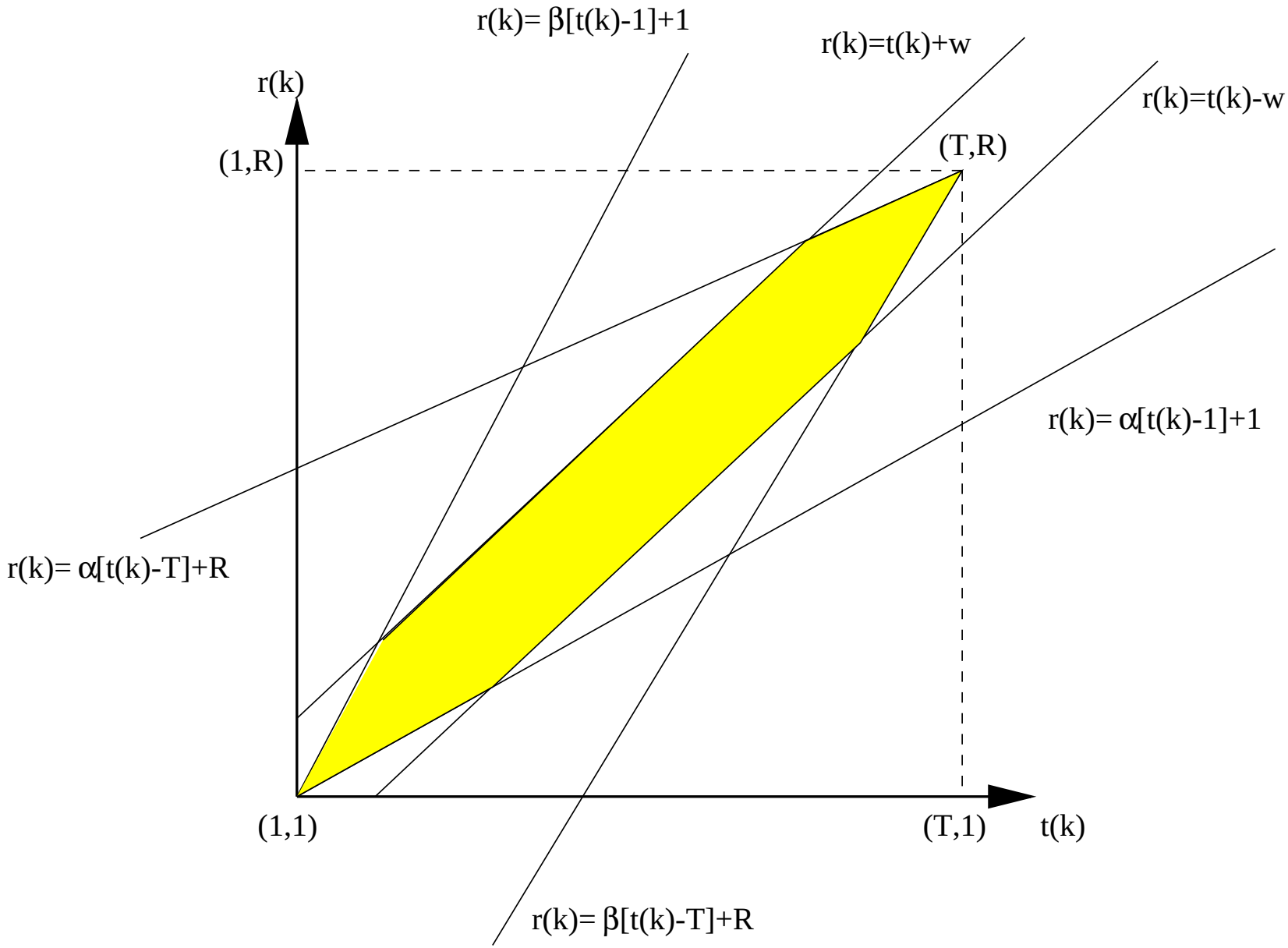
v praxi se volí $R^\star, T^\star$=1,2,3.

- $R^\star, T^\star$=1: Každý vektor se musí vzít alespoň jedenkrát. $r(k) = r(k-1)$ znamená, že se opakuje.
- $R^\star, T^\star > 1$: Vektor(y) se mohou přeskočit.

3. **Globální vymezení cesty DTW:** vymezení přípustné oblasti pomocí přímek:

$$\begin{aligned} 1 + \alpha[t(k) - 1] \leq r(k) \leq 1 + \beta[t(k) - 1] \\ R + \beta[t(k) - T] \leq r(k) \leq R + \alpha[t(k) - T] \end{aligned} \tag{8}$$

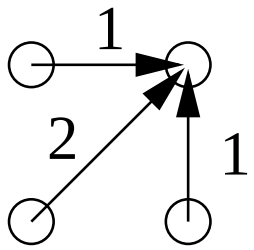
Tyto podmínky vymezují maximální "strmost" nebo "placatost" cesty DTW


r(k)= β[t(k)-1]+1
r(k)=t(k)+w
r(k)
r(k)=t(k)-w
(1,R)
(T,R)
r(k)= α[t(k)-1]+1
r(k)= α[t(k)-T]+R
(1,1)
(T,1)
t(k)
r(k)= β[t(k)-T]+R

Definice váhových funkcí

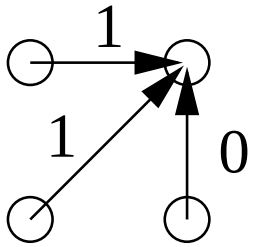
Váhová funkce $W(k)$ závisí na lokálním "posunu" cesty. 4 typy:

- **typ a)** symetrická: $W_a(k) = [t(k) - t(k-1)] + [r(k) - r(k-1)]$.

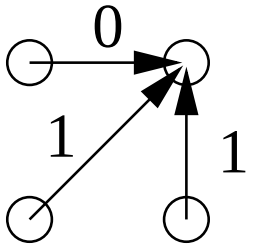


- **typ b)** asymetrická:

b1) $W_{b1}(k) = t(k) - t(k-1)$



b2) $W_{b2} = r(k) - r(k-1)$



- **typ c)** $W_c(k) = \min\{t(k) - t(k-1), r(k) - r(k-1)\}$
- **typ d)** $W_d(k) = \max\{t(k) - t(k-1), r(k) - r(k-1)\}$

Normalisační faktor

$$N = \sum_{k=1}^{K} W(k) \tag{9}$$

Pro váhovací funkci typu a) je normalisační faktor:

$$N_a = \sum_{k=1}^{K} [t(k) - t(k-1) + r(k) - r(k-1)] = t(K) - t(0) + r(K) - r(0) = T + R \tag{10}$$

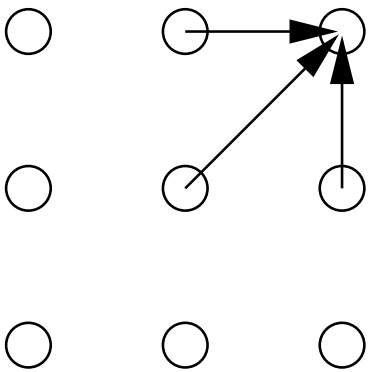
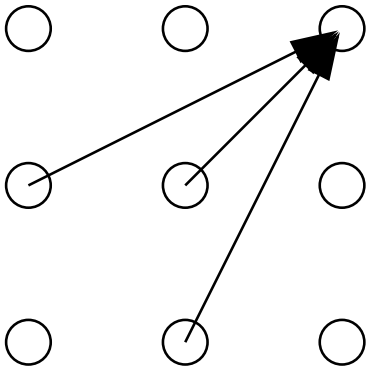
Pro váhovací funkci typu b1) je normalisační faktor $N = T$.

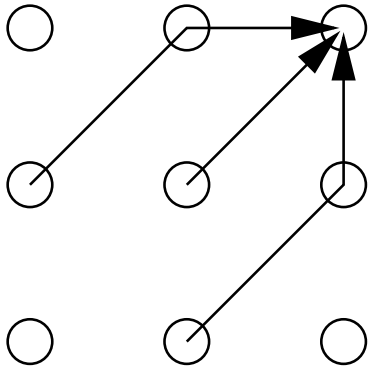
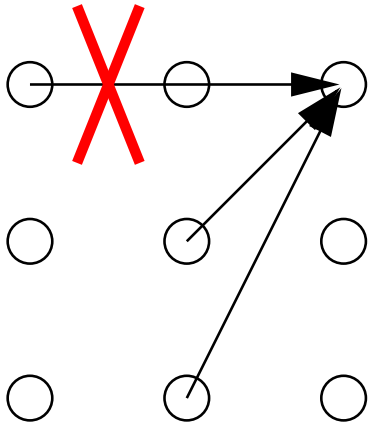
Pro váhovací funkci typu b1) je normalisační faktor $N = R$.

Pro typy c), d) faktor silně závislý na průběhu cesty, je lepší použít konstantu: $N = T$.

Lokální omezení cesty

Tabulka udává typy lokálních omezení cesty a z nich vyplývající faktory α a β. Význam veličiny $g(n, m)$ bude vysvětlen později.

Typ		α	β	Typ $w(k)$	$g(n,m)$
I.		0	∞	a	$\min \left\{ \begin{array}{l} g(n,m-1)+d(n,m) \\ g(n-1,m-1)+2d(n,m) \\ g(n-1,m)+d(n,m) \end{array} \right\}$
				d	$\min \left\{ \begin{array}{l} g(n,m-1)+d(n,m) \\ g(n-1,m-1)+d(n,m) \\ g(n-1,m)+d(n,m) \end{array} \right\}$
II.		$\frac{1}{2}$	2	a	$\min \left\{ \begin{array}{l} g(n-1,m-2)+3d(n,m) \\ g(n-1,m-1)+2d(n,m) \\ g(n-2,m-1)+3d(n,m) \end{array} \right\}$
				d	$\min \left\{ \begin{array}{l} g(n-1,m-2)+d(n,m) \\ g(n-1,m-1)+d(n,m) \\ g(n-2,m-1)+d(n,m) \end{array} \right\}$

III.		$\frac{1}{2}$	2	a	$\min \left\{ \begin{array}{l} g(n-1,m-2)+2d(n,m-1)+d(n,m) \\ g(n-1,m-1)+2d(n,m) \\ g(n-2,m-1)+2d(n-1,m)+d(n,m) \end{array} \right\}$
IV.		$\frac{1}{2}$	2	b1	$\min \left\{ \begin{array}{l} g(n-1,m)+kd(n,m) \\ g(n-1,m-1)+d(n,m) \\ g(n-1,m-2)+d(n,m) \end{array} \right\}$ kde $k=1$ pro $r(k-1) \neq r(k-2)$ $k=\infty$ pro $r(k-1)=r(k-2)$

Efektivní výpočet $D(\mathbf{O}, \mathbf{R})$

Výpočet minimální vzdálenosti

$$D(\mathbf{O}, \mathbf{R}) = \min_{\{C\}} D_C(\mathbf{O}, \mathbf{R}). \tag{11}$$

je jednoduchý, pokud normalisační faktor N_C není funkcí cesty a můžeme psát:

$$N_C = N \quad \text{pro} \quad \forall C$$

Toto naštěstí většinou platí a tedy:

$$D(\mathbf{O}, \mathbf{R}) = \frac{1}{N} \min_{\{C\}} \sum_{k=1}^{K_C} d[\mathbf{o}(t_C(k)), \mathbf{r}(r_C(k))] W_C(k) \tag{12}$$

Postup je následující:

1. do mřížky $\mathbf{d}$ o velikosti $T \times R$ si zapíšeme vzdálenosti referenčních a testovacích vektorů, každý s každým, viz Příklad.
2. definujeme mřížku $\mathbf{g}$ s *částečnou kumulovanou vzdáleností*. Oproti mřížce $\mathbf{d}$ má $\mathbf{g}$

navíc ještě nultý řádek a nultý sloupec, které inicialisujeme na:

$$g(0,0) = 0, \quad \text{a} \quad g(0, m \neq 0) = g(n \neq 0, 0) = \infty.$$

3. Částečnou kumulovanou vzdálenost spočítáme pro každý bod takto:

$$g(m,n) = \min_{\forall \text{předchůdci}} [g(\text{předchůdce}) + d(m,n)w(k)] \tag{13}$$

   - možní předchůdci jsou dáni pomocí tabulky lokálních omezení cesty.
   - váha $w(k)$ odpovídá pohybu z předchůdce do bodu $[m,n]$.
   - vztahy pro výpočet Částečné kumulované vzdálenosti jsou tabelovány v Tabulce

4. Konečná minimální normovaná vzdálenost je pak dána:

$$D(\mathbf{O}, \mathbf{R}) = \frac{1}{N} g(T,R) \tag{14}$$

Příklad

d

ref.			
	4	3	2
	2	3	1
	4	2	3
	0	1	1
		test	

g

ref.			
inf	10	9	7
inf	6	6	5
inf	4	3	5
inf	0	1	2
0	inf	inf	inf
		test	

Výsledek:

- máme vzdálenost $D = \frac{1}{3+4}7 = 1$.
- můžeme zpětně "odkrokovat optimální cestu" (cesta má 5 kroků): $t(k) = [1\ 2\ 2\ 3\ 3]$, $r(k) = [1\ 1\ 2\ 3\ 4]$.

Realisace rozpoznávače založeného na DTW

Opakování: co vlastně chci ? Přijde slovo $\mathbf{O}$; chci ho zatřídit do třídy ω_r. Mám $\check{N}$ tříd, které odpovídají slovům (např. “jedna”, “dvě”, “křížek”, atd.).

Trénování – tvorba referencí neboli vzorů

pří trénování mám sekvence od jednoho nebo více řečníků, a vím, kam která patří.

1. **nejjednodušší:** ve třídě ω_r mám jen jednu referenci $\mathbf{R}_r$.
2. **složitější:** pro každou třídu ω_r mám více referencí: $\mathbf{R}_{r,1} \ldots \mathbf{R}_{r,\check{N}_r}$. Tyto mohu mít ve slovníku uložené tak, jak byly vytvořeny, nebo je lineárně normalisovat na jednotnou délku:

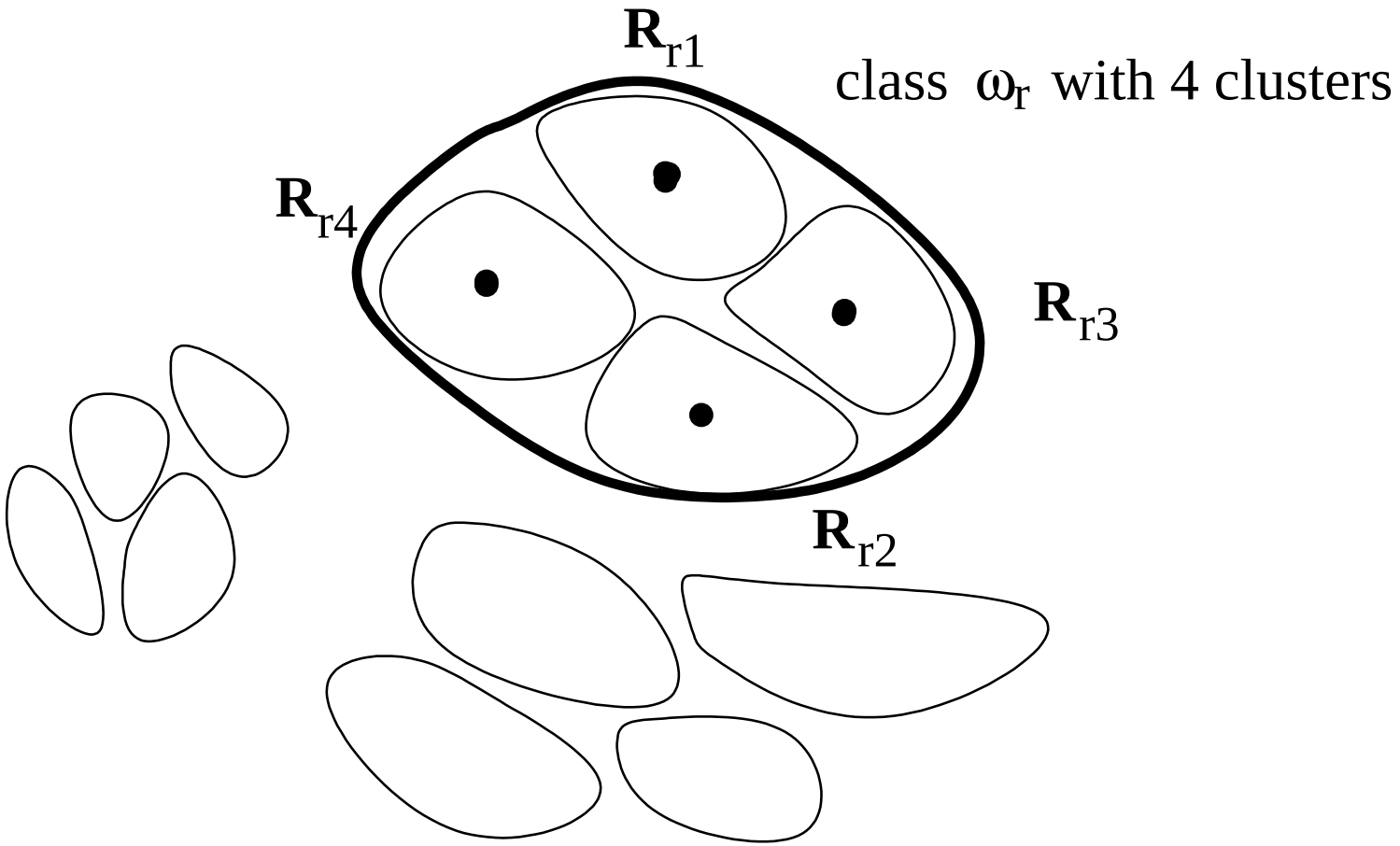
$$\bar{R} = \frac{1}{N} \sum_{r=1}^{N} \left[\frac{1}{\check{N}_r} \sum_{i=1}^{\check{N}_r} R_{r_i} \right], \tag{15}$$

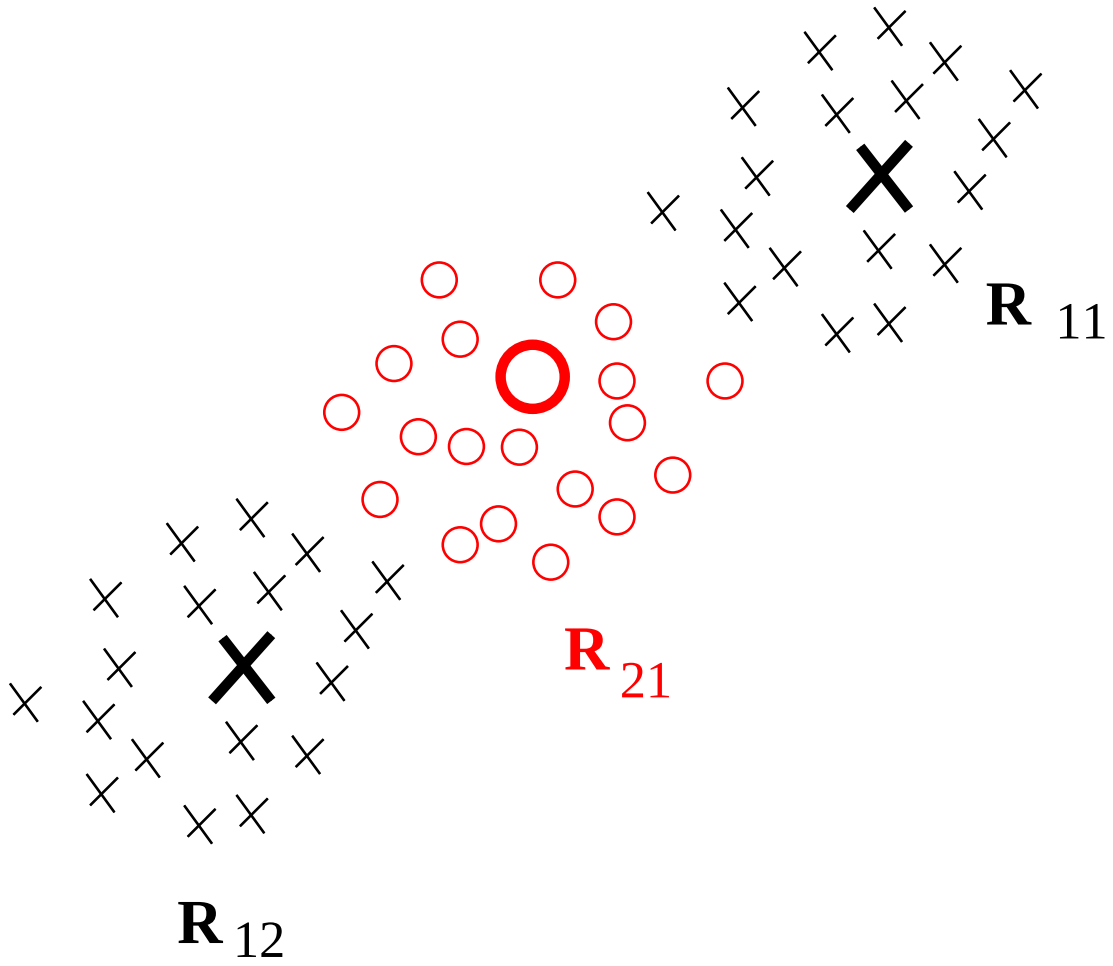
   kde R_{r_i} je délka i-tého vzoru třídy ω_r.
3. **vytvoření průměrného vzoru pro každou třídu** ω_r:
   - lineární průměrování – bereme průměr vektorů lineárně srovnaných sekvencí. Nebezpečí: mohu dostat zcela nesmyslný vzor ...
   - dynamické průměrování:

   (a) najdu vzor s o délkou, která se mi líbí.

   (b) průměrování se děje z vektorů přiřazených k tomuto prvnímu vzoru pomocí cesty DTW.

4. vytváření vzorů *shlukováním*. Shluky jsou tvořeny tak, aby si byly reference uvnitř shluků co nejvíce podobné, mezi shluky co nejméně podobné. Existují automatické algoritmy pro shlukování, např. Mac Queenův: nejprve se všechny reference přiřadí do jednoho shluku. Pak se odštěpí nejvzdálenější od středu, "přeshlukuje se", atd. (analogie s VQ). Shluky jsou representovány *centroidy* $\mathbf{R}_{ri}$. Výhoda shlukování oproti průměrování spočívá v tom, že třídy mohou mít složitější tvar.




R 11
R 21
R 12

Rozpoznávání (klasifikace)

Je-li každá třída representována jen jednou referencí, je to jednoduché:

$$\omega_r^\star = \arg\min_r D(\mathbf{O}, \mathbf{R}_r) \quad \text{pro} \quad r = 1, \ldots, N \tag{16}$$

Mám-li pro každou třídu více referencí, je možné použít dvě metody:

1. **1-NN** (nearest neighbor) neboli nejbližší soused:

$$\omega_r^\star = \arg\min_{r,i} D(\mathbf{O}, \mathbf{R}_{r_i}) \quad \text{pro} \quad \begin{matrix} r = 1, \ldots, N \\ i = 1, \ldots, N_r \end{matrix} \tag{17}$$

2. k-**NN** (k nearest neighbors) neboli k nejbližších sousedů:

   - pro každou třídu nejprve spočítám všechny vzdálenosti $D(\mathbf{O}, \mathbf{R}_{ri})$ a seřadím je podle velikosti od nejlepší po nejhorší:

$$D(\mathbf{O}, \mathbf{R}_{r_{(1)}}) \leq D(\mathbf{O}, \mathbf{R}_{r_{(2)}}) \leq \ldots \leq D(\mathbf{O}, \mathbf{R}_{r_{(N_r)}}) \tag{18}$$

   - o klasifikaci $\mathbf{O}$ do třídy ω_r se pak rozhodnu podle průměrné vzdálenosti k nejbližších sousedů:

$$\omega_r^\star = \arg\min_r \frac{1}{k} \sum_{i=1}^{k} D(\mathbf{O}, \mathbf{R}_{r_{(i)}}) \tag{19}$$