

Počítačové zpracování řeči

Přednáška 7

Od DTW k HMM aneb
od referencí k pravděpodobnostním modelům

Diskuse k minulé úloze

Ukázka nejlepších dosažených výsledků (Martin Motejlek):

Speaker dependent			Speaker independent		
KNN: K = 1; příznaky: MFCC			KNN: K = 13; příznaky: MFCC + delta MFCC		
Sady	Čas (parfor)	Přesnost	Sady	Čas (parfor)	Přesnost
1	8	96,3	1	29	85,3
1-2	9	97,5	1-2	43	90,0
1-3	10	98,8	1-3	60	92,2
			1-4	89	92,2
			1-5	101	92,5

Metoda DTW – pro a proti

Výhody DTW:

- **snadno pochopitelná metoda**
- **snadná příprava systému** (stačí pouze nahrát referenci pro každé slovo)
- **snadná implementace** i na nevýkonnému HW (např. embedded systems)
- **zvládnutí DTW usnadní pochopení modernějších metod**

Nevýhody:

- reference jsou **závislé na mluvčím (SD)**
- **SI system** vyžaduje **reference od mnoha osob**
- **SI systém s velkým slovníkem** by byl zákonitě **příliš pomalý**
(smyčky pro každé slovo, pro každou jeho referenci, pro každý frame)

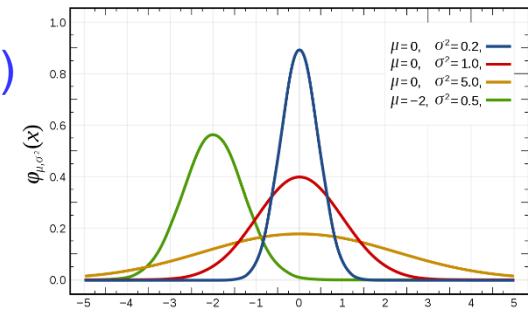
„Průměrná“ reference?

Idea: Lze nahradit reference od mnoha mluvčích jedinou „průměrnou“ referencí (a zrychlit + zlepšit rozpoznávání) ?

- Jakou by měla mít délku?
- Jak určit „průměrné“ příznaky pro každý frame?
- Zachová průměrování (mnoha dat) charakteristické rysy slova?

Co si představit jako průměrování (po framech)?

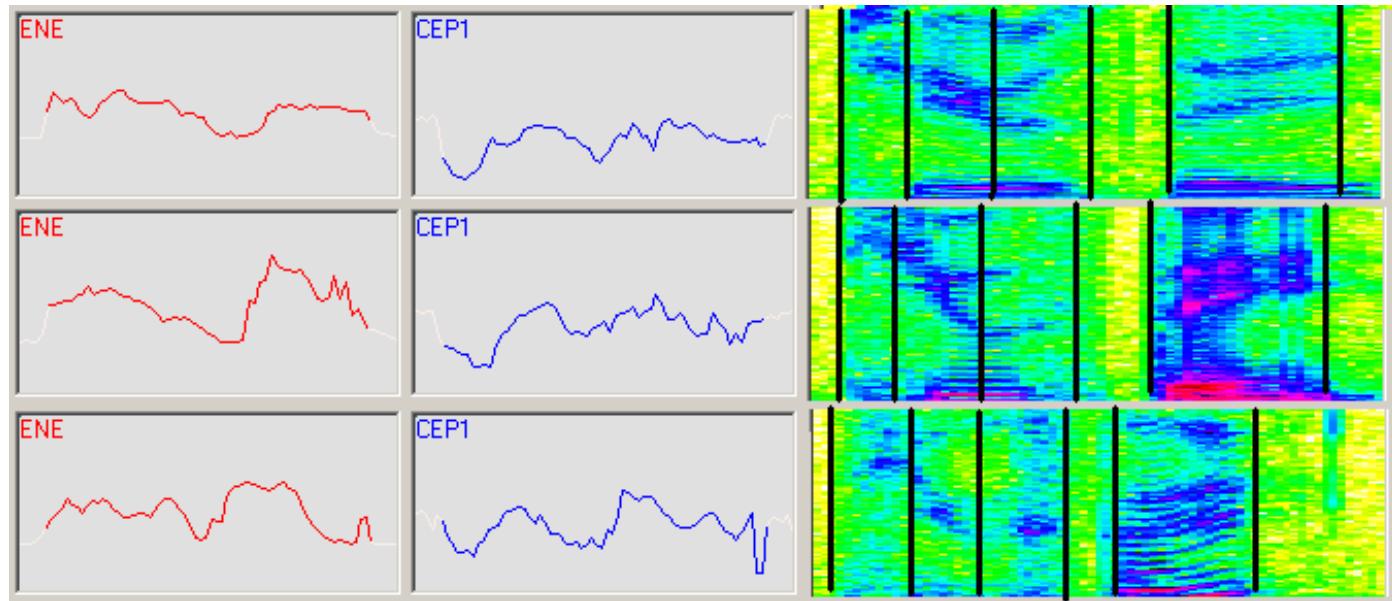
- Náhrada mnoha hodnot jedinou – střední hodnotou $\bar{x} = \frac{1}{N} \sum_{n=1}^N x_n$
- Příklad dat: 4, 2, 1, 7, 6, 9, 3, 8 stř. hodnota = ...
 4, 5, 4, 6, 5, 6, 5, 5 stř. hodnota = ...
- Samotná stř. hodnota nestačí, důležitý je i rozptyl $\sigma^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \bar{x})^2$
- „Chytré průměrování“ spočívá v přiřazení vhodného pravděpodobnostního rozložení (nejčastěji gaussovského) k daným datům a nalezení jeho parametrů (u gauss. – střed. hodnota a rozptyl)



Přechod od framů ke stavům

Podívejme se na příznaky stejného slova ("Tuesday") řečeného 3 osobami

Osoba1

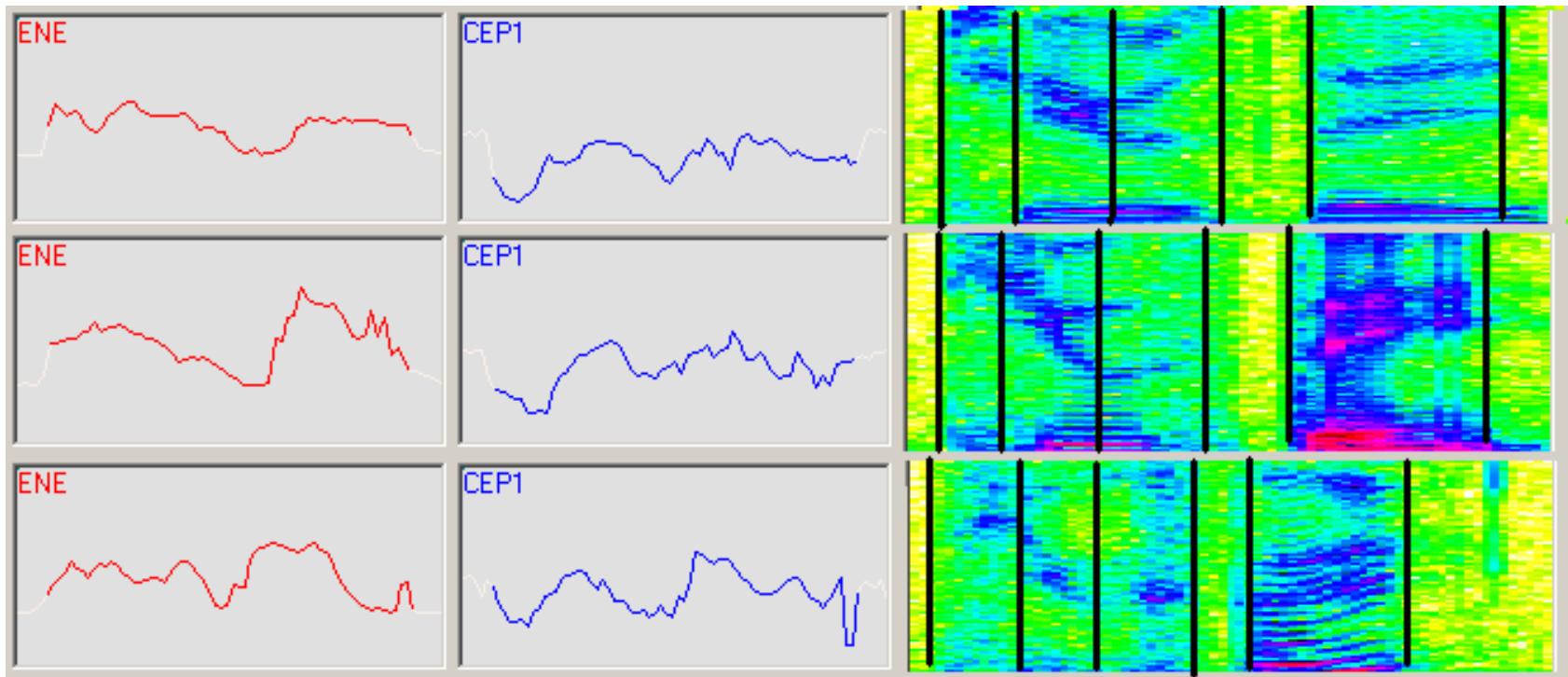


Osoba2

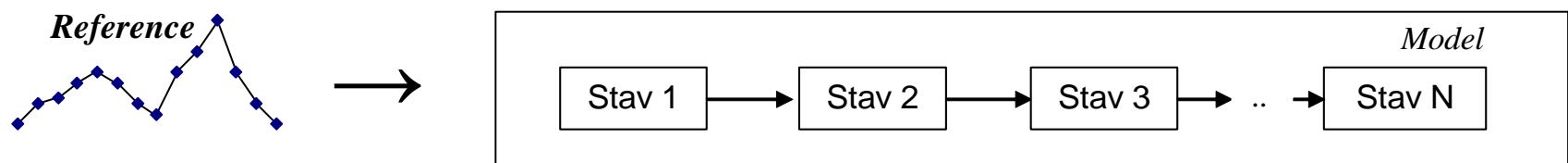
Vidíme, že **charakter signálu** (a stejně tak i průběh příznaků) **se nemění skokem** z framu na frame. Spíše lze pozorovat určité **oblasti** s podobným charakterem signálu (tyto oblasti souvisí s hláskami)

Idea: Zkusme reprezentovat slovo sekvencí stavů (ne tedy už sekvencí jednotlivých framů), kdy každý stav charakterizuje jednu oblast

Od referencí k modelům



Rozpoznávané slovo budeme i dál reprezentovat příznakovými vektory, ale pro porovnání místo **slovních referencí** použijme **slovní modely** reprezentované (několika málo, 6 - 20) stavů popsanými pravděp. rozloženími příznaků



Od DTW k HMM

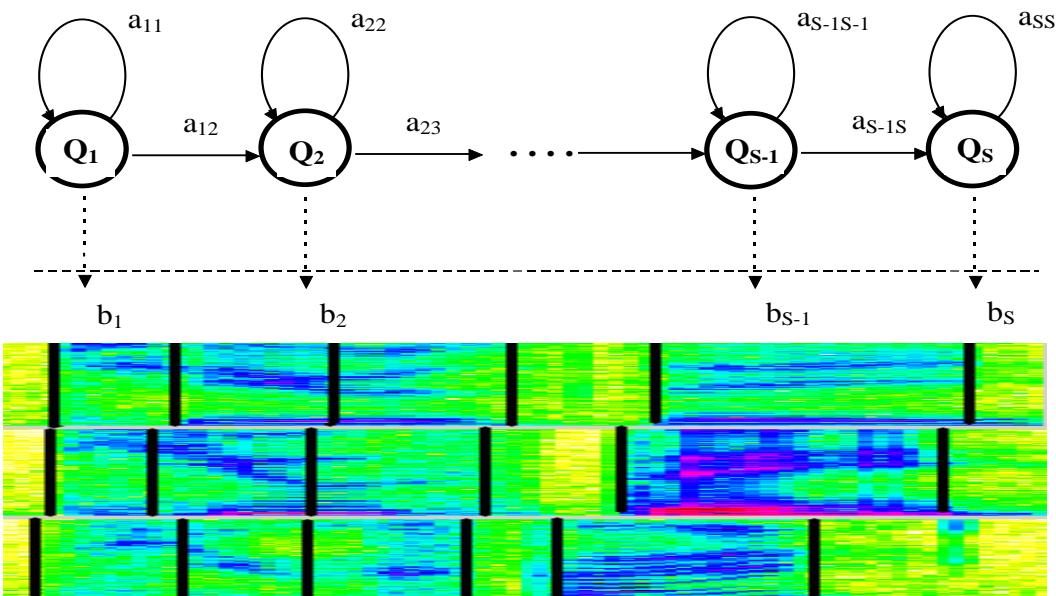
Metoda HMM (Hidden Markov Model – skryté Markovovy modely)
reprezentuje slovo **stavovým modelem**

s pravděpodobnostními parametry

Typická struktura slovního HMM

– takzvaný *levo-pravý model*

Můžeme si představit, že každý stav reprezentuje jednu z vyznačených oblastí



$Q_s \dots$ **stavy** (šipky naznačují možné přechody mezi nimi – buď setrvání nebo přechod doprava)

$a_{ij} \dots$ **přechodová pravděpodobnost** – pravděpodobnost, že (v aktuálním framu) model přejde ze stavu i do stavu j (a_{ii} je tedy pravděpodobnost setrvání ve stavu i)

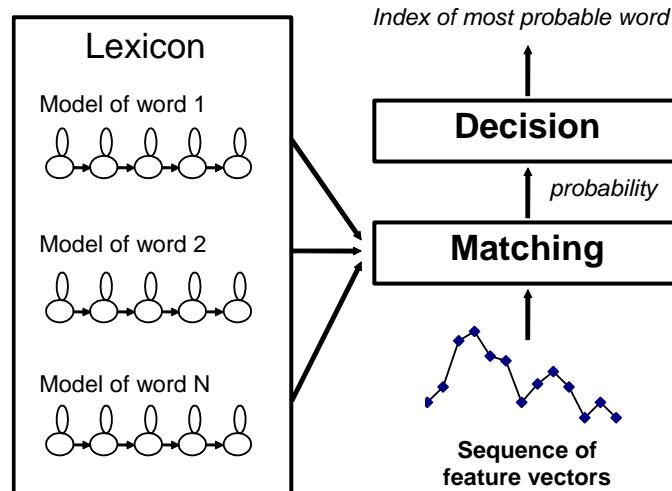
$b_s(x) \dots$ **výstupní pravděpodobnostní rozložení** – funkce určující míru pravděpodobnosti, že příznakový vektor x patří ke stavu s – v praxi se používá normální (Gaussovo) rozložení

Jak rozpoznávat pomocí HMM?

Základní princip:

Všechna slova ve slovníku jsou reprezentována modely se **stejnou strukturou** (a stejným počtem stavů) ale **různými parametry** a_{ij} a b_s

Při rozpoznávání slovo **reprezentujeme framy a příznaky úplně stejně jako dosud**. Při klasifikaci nyní hledáme **nejpravděpodobnější model**.



Oproti DTW schématu se tedy změnila pouze klasifikace. Zde spočívá ve vyhodnocení pravděpodobnosti modelů všech slov a nalezení modelu s nejvyšší z nich. Výhoda oproti DTW – každá třída je zastoupena jediným natrénovaným modelem.

Jak určit (natreňovat) parametry HMM?

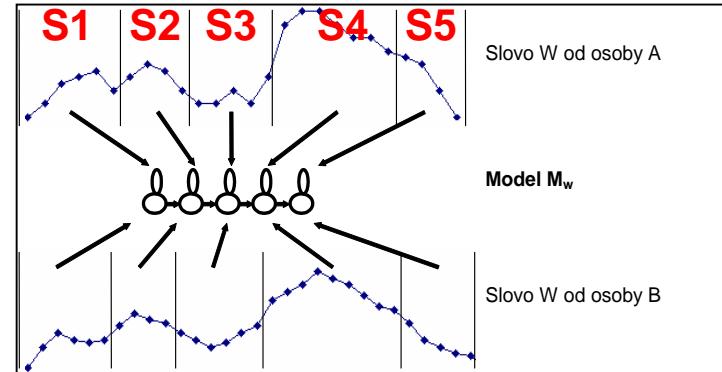
Pro zjednodušení uvažujme 1-rozměrný příznakový vektor \mathbf{x} (např. ENE)

Předpokládáme, že $b_s(x)$ má gauss. rozdělení se dvěma parametry μ_s, σ_s

$$b_s(x) = \frac{1}{\sqrt{2\pi} \cdot \sigma_s} \exp\left[-\frac{(x - \mu_s)^2}{2\sigma_s^2}\right]$$

Mějme alespoň 2 nahrávky pro každé slovo
(čím více nahrávek, čím lepší model získáme)

a pro každé slovo sekvenci jeho příznaků



Pokud víme, které framy patří k jednotlivým stavům

(jak je ukázáno na obrázku, kde svislé čáry vymezují 5 stavů)

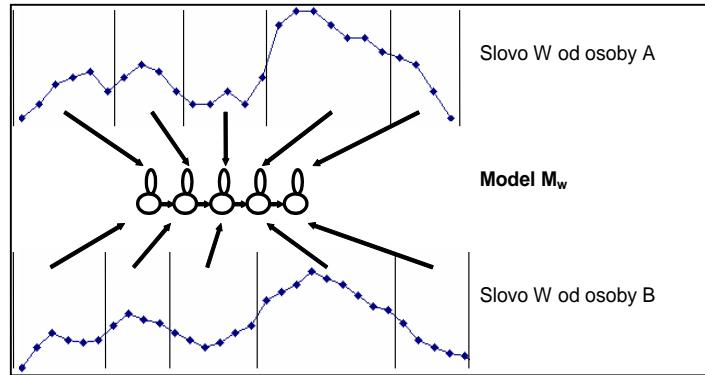
pak stř. hodnotu určíme jako

$$\mu_s = \frac{1}{N_s} \sum_{n=1}^{N_s} \mathbf{x}_n \quad \text{a rozptyl jako} \quad \sigma_s^2 = \frac{1}{N_s} \sum_{n=1}^{N_s} (\mathbf{x}_n - \mu_s)^2$$

kde N_s je počet framů přiřazených stavu s

Jak natrénovat parametry HMM (2)?

Stejně jednoduše se dají určit i přechodové pravděpodobnosti



Pravděpodobnost přechodu ze stavu s do $s+1$

$$a_{ss+1} = \frac{K}{N_S}$$

Proč? U každého slova dojde k přechodu ze stavu s do $s+1$ právě jednou, u K slov tedy K -krát, a to vydělíme souhrnným počtem framů přiřazených ke stavu s .

Pravděpodobnost setrvání v tomtéž stavu s

$$a_{ss} = 1 - a_{ss+1}$$

Jedná se o pravděpodobnost opačného jevu

Model je natrénovaný, když pro každý jeho stav známe všechny parametry, tj. hodnoty $\mu_s, \sigma_s, a_{ss}, a_{ss+1}$

Jak trénovat HMM (3)?

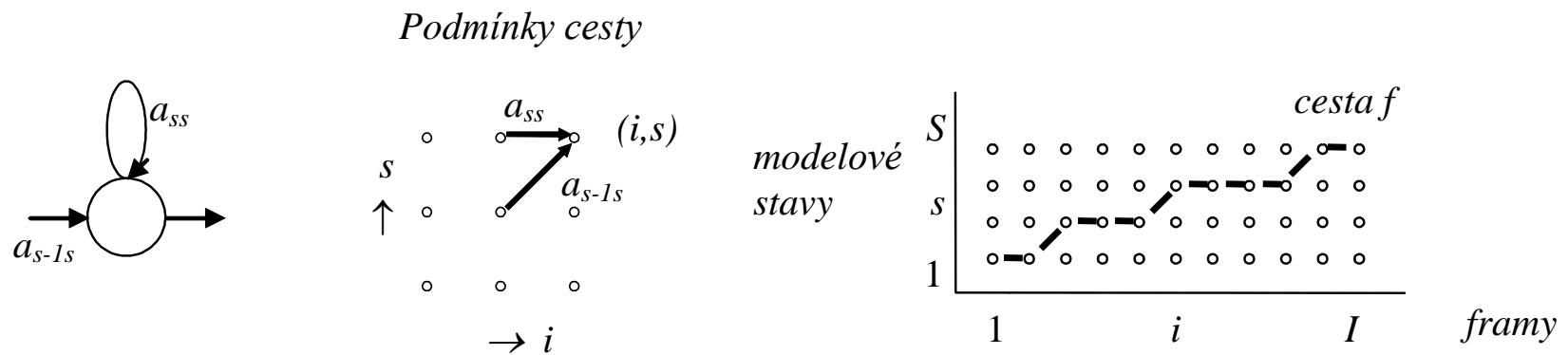
Ve skutečnosti nevíme, který frame patří k jakému stavu.
(Z tohoto důvodu se metodě HMM říká **skryté** markovovské modely)

Jak na to? Vysvětlíme si příště.

Jak rozpoznávat pomocí HMM?

Podobně jako u metody DTW

Opět hledáme **nejlepší cestu** (přiřazovací funkci f s nejvyšší pravděpodobností), tentokrát ovšem **v rovině slovních framů a modelových stavů** z bodu $(1, 1)$ do (I, S)



Pravděpodobnostní „skóre“ pro výše nakreslenou cestu f určíme jako **součin pravděpodobností dílčích úseků**

$$P(f) = b_1(x_1) \cdot a_{11} b_1(x_2) \cdot a_{12} b_2(x_3) \cdot a_{22} b_2(x_4) \dots$$

Určit pravděpodobnostní skóre rozpoznávaného slova pro daný model znamená najít nejlepší skóre ze všech možných cest.

$$HMM : \quad P(\mathbf{X}, \mathbf{M}) = \underset{f}{\operatorname{Max}} \prod_{i=1}^I a_{f(i-1)f(i)} b_{f(i)}(\mathbf{x}(i))$$

Pro porovnání
DTW : $D(\mathbf{X}, \mathbf{R}) = \underset{w}{\operatorname{Min}} \sum_{i=1}^I d(x_i, r_{w(i)})$

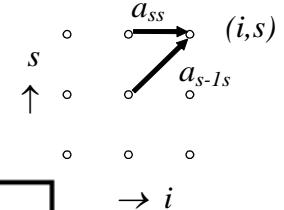
Viterbiho algoritmus

Jde o základní algoritmus pro rozpoznávání i trénování HMM

Definujme - kumulované skóre v bodě (i, s) :

$$V(i, s) = b_s(x_i) \cdot \text{Max}[a_{ss}V(i-1, s), a_{s-1s}V(i-1, s-1)]$$

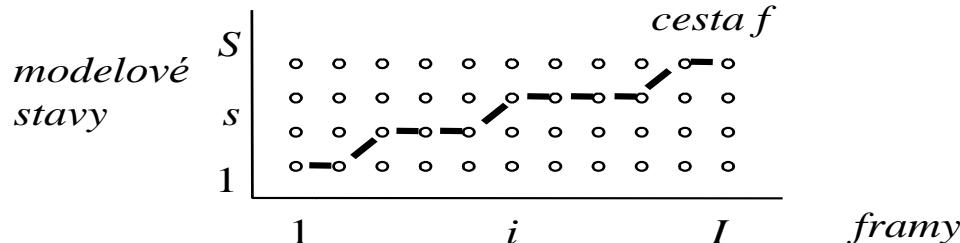
Viterbi algorithm	
Step 1:	<i>Initialization</i> $V(1,1) = b_1(x_1)$ $V(1,s) = -\infty$ pro $s=2,\dots,S$ $B(1,1) = 1$
Step 2:	<i>Recursion</i> For $i = 2, \dots, I$ For $s = 1, \dots, S$ $O(k) = a_{ks}V(i-1, k)$ pro $k = s-1, s$ (<i>temporary variable</i>) $V(i,s) = b_s(x_i) \cdot \text{Max}_k [O(k)]$ $B(i,j) = \text{ArgMax}_k [O(k)]$
Step 3:	<i>Termination</i> $P(\mathbf{X}, \mathbf{M}) = V(I, S)$
Step 4:	<i>Backtracking</i> $f(I) = S$ for $i = I-1, \dots, 1$ $f(i) = B(i+1, f(i+1))$



Všimněte si, že Vit. algoritmus je velmi podobný DTW. Pouze místo vzdál. počítá pravděpodobnost, místo sčítání se násobí, místo operátoru Min se pracuje s Max, ke každému bodu existují pouze 2 předchůdci a místo Eukl. vzd. se vyhodnocuje výstup. funkce (gaussovka)
Část „Backtracking“ není nutná pro rozpoznávání, ale je klíčová pro trénování

Efektivní implementace

Výpočty se zefektivní (zrychlí), když budeme místo pravděpodobností používat jejich (přirozený) logaritmus.



Připomeňme, že pravděp. „skóre“ pro výše uvedenou cestu f určíme jako

$$P(\mathbf{X}, \mathbf{M}) = b_1(x_1) \cdot a_{11} b_1(x_2) \cdot a_{12} b_2(x_3) \cdot a_{22} b_2(x_4) \dots$$

S využitím přirozeného logaritmu

$$\ln(P(\mathbf{X}, \mathbf{M})) = \ln(b_1(x_1)) + \ln(a_{11}) + \ln(b_1(x_2)) + \ln(a_{12}) + \ln(b_2(x_3)) \dots$$

- a) ve Viterbiho algoritmu se veškeré násobení nahradí sčítáním
- b) hodnoty $\ln(a_{ij})$ si spočítáme dopředu (po natrénování) a uložíme
- c) také výpočet $\ln(b_s(x))$ se dá zefektivnit

$$b_s(x) = \frac{1}{\sqrt{2\pi} \cdot \sigma_s} \exp \left[-\frac{(x - \mu_s)^2}{2\sigma_s^2} \right]$$

$$\ln(b_s(x)) = \ln\left(\frac{1}{\sqrt{2\pi} \cdot \sigma_s}\right) - \frac{(x - \mu_s)^2}{2\sigma_s^2}$$

první člen je konstanta (pro daný stav), druhý člen už snadno spočítáme

Jak s vícepříznakovými vektory

Vysvětlíme si příště.

Úloha pro cvičení

Rozpoznávač číslic založený na HMM

1. Cílem úlohy je naimplementovat Viterbiho algoritmus a otestovat ho na dodaných datech a modelech
2. Dodaná data představují nahrávky jedné osoby (p0079) – 50 slov, 3x10 použito na trénování, 2x10 je pro vaše testy
3. Data jsou již zparametrisována, a to pomocí 1 příznaku (ENE)
4. Modely jsem pro vás natrénoval

Tipy pro implementaci (1)

- Stáhněte si z elearningu soubor ZIP, v němž najdete 3 soubory:
 - soubor word_features-ENE.mat s připravenými daty,
 - soubor Load_data_for_Viterbi_test.m, v němž jsou podrobně popsány významy všech dat a proměnných,
 - soubor Prepare_data_for_Viterbi_test.m, v němž je kód, kterým jsem data připravil – může se vám hodit, pokud by něco nebylo jasné z popisu
- Pro implementaci využijte schéma Viterbiho algoritmu uvedené v přednášce
- Algoritmus implementujte tak, že už bude počítat logaritmické hodnoty, čímž se řada operací zjednoduší (např. místo součinu bude sčítání, atd.)
- Vytvořte si zvlášť funkci pro výpočet výstupní pravděpodobnosti, tj, výpočet **logaritmu gaussovky**, do které vstoupí:
 - příslušná střední hodnota, rozptyl a konstanta pro daný model a daný stav,
 - hodnota příznaku ENE v daném framu daného slova,
 - jedná se jednoduchý výpočet na 1 řádek
- Algoritmus je podobný DTW, takže by implementace neměla činit problém

Tipy pro implementaci (2)

- Algoritmus otestujte na testovacím setu, pro nějž jsem vám připravil kontrolní výsledky. Celková úspěšnost na 20 slovech by měla vyjít 75 %.
- Váš testovací program se může inspirovat mým výše uvedeným programem
- Program očekávám opět nejdéle do pondělí