Rozpoznávání pomocí DTW a HMM

Jan Černocký, FIT VUT Brno

20. dubna 2016

V tomto cvičení si ukážeme dvě základní techniky pro rozpoznávání izolovaných slov:

- dynamického borcení času (DTW – Dynamic Time Warping), které srovnává jednotlivé matice parametrů pomocí nalezení optimální srovnávací cesty
- skrytých Markovových modelů, kdy se na trénovacích promluvách trénují statistické modely a tyto jsou pak použity pro skórování testované promluvy.

1 Základní pojmy

Referenční promluva nebo **referenční sekvence** jsou data, která jsou známá a použijeme je při trénování. **Testovací promluva** nebo **testovací sekvence** je neznámá a j na vstupu natrénovaného rozpoznávače. V tomto cvičení samozřejmě známe obsah testovací promluvy, abychom mohli vyhodnotit, zda rozpoznávač pracuje správně.

2 Signály a parametrizace

Jako parametrizace jsou použity LPC-cepstrální koeficienty (viz přednáška o LPC). Ve funkci `c_matrice.m` je poněkud podivně implementován výpočet LPC-koeficientů (stačilo by na každý rámec zavolat funkci `lpc`, zřejmě jsem si před lety, kdy jsem `c_matrice` psal, potřeboval dokázat, že tomu rozumím...). Podívejte se ale do funkce `a_to_cepst.m` na implementaci převodu LPC koeficientů na LPCC — v přednášce je to úplně na konci, rovnice (38).
Jsou připraveny 4 trénovací promluvy a 7 testovacích promluv. Spuštěním

```
sigceps
```

se vše načte a zparametrizuje. Výsledkem je:

signál	cepstrální matice	popis
Trénování		
s1	c1	jedna
s2	c2	dvě
s3	c3	tři
s4	c4	čtyři
Testování		
s1t	c1t	jedna (jiná promluva)
s2t	c2t	dvě (jiná promluva)
s3t	c3t	tři (jiná promluva)
s4t	c4t	čtyři (jiná promluva)
s1l	c1l	jééédna
s2l	c2l	dvjééé
sb	cb	bedna

3 DTW

Pomocí DTW srovnáváme dvě posloupnosti vektorů: referenční $\mathbf{R} = [\mathbf{r}(1), \ldots \mathbf{r}(R)]$ o délce R a testovací $\mathbf{O} = [\mathbf{o}(1), \ldots \mathbf{o}(T)]$ o délce T. Chceme znát jejich vzdálenost $D(\mathbf{O}, \mathbf{R})$. V nejjednodušším případě odpovídá i-tému slovu w_i ve slovníku jen jedna reference $\mathbf{R}_i$, rozpoznané slovo je pak dáno jako:

$$i^\star = \arg\min_i D(\mathbf{O}, \mathbf{R}_i). \tag{1}$$

Při DTW definujeme matici $\mathbf{D}$ o rozměrech $T \times R$, kterou zaplníme lokálními vzdálenostmi vektorů: $d(i,j) = d(\mathbf{o}(i), \mathbf{r}(j))$. Vzdálenost $d(\cdot,\cdot)$ je například cepstrální míra. Druhou důležitou maticí je $\mathbf{G}$ – matice částečných

kumulovaných vzdáleností. Oproti **D** má **G** navíc nultý řádek a nultý sloupec, které inicializujeme hodnotami ∞, kromě $g(0,0) = 0$. Pro lokální omezení cesty typu I. a pro váhy typu a. (viz přednáška o DTW) můžeme další prvky **G** vypočítat:

$$g(i,j) = \min \begin{cases} g(i-1,j) + d(i,j), \\ g(i,j-1) + d(i,j), \\ g(i-1,j-1) + 2d(i,j) \end{cases} \tag{2}$$

pro $1 \leq i \leq T$ a $1 \leq j \leq R$. Poslední prvek této matice $g(T,R)$ udává celkovou optimální vzdálenost:

$$D(\mathbf{O},\mathbf{R}) = \frac{g(T,R)}{T+R} \tag{3}$$

V matici **G** mohu pak zpětně vyhledat optimální cestu, podle které se sekvence srovnávaly.
Všechny tyto operace jsou implementovány ve funkci `dtw.m`, kterou si **pečlivě prohlédněte**. Funkce má na výstupu DTW vzdálenost ale důležitý je její grafický výstup:

- na horním panelu je vidět matice lokálních vzdáleností (vektory každý s každým).
- na prostředním panelu je vidět matice částečných kumulovaných vzdáleností. Všimněte si, že je v ní vidět tmavé "koryto", kterým probíhá optimální srovnávací cesta.
- na spodním panelu je vidět tato cesta. V titulku tohoto panelu je zobrazena DTW vzdálenost.

3.1 Srovnání matice se sebou samou

Nejprve srovnáme jednu trénovací promluvu s tou samou promluvou:

```
D=dtw (c4, c4); soundsc ([s4 s4])
```

Úkoly

1. Jaká je DTW vzdálenost ?
2. Proč je srovnávací cesta úplně rovná ?
3. Proč vidíme na matici lokálních vzdáleností "čtyřlístek" černých oblastí v pravém horním rohu ?

3.2 Srovnání matic pro dvě různé promluvy stejného slova

```
D=dtw (c4, c4t); soundsc ([s4 s4t])
```

Úkoly

1. Komentujte, co vidíte.

3.3 Srovnání matic pro dvě různé promluvy stejného slova

Zkusíme 'jedna' a 'jééédna':

```
D=dtw (c1, c1l); soundsc ([s1 s1l])
```

Úkoly

1. Vidíte na zobrazených maticích a na srovnávací cestě srovnání krátkého 'e' s dlouhým ?

3.4 Skutečné rozpoznávání

Zatím jsme si ukazovali pouze srovnání jedné sekvence s jednou, nic se nerozpoznávalo. Skutečné rozpoznávání musí testovací promluvu srovnat se všemi referenčními a vybrat nejmenší vzdálenost. Následující kód otevře 4 grafická okna, poskládejte si je jako svislé pruhy vedle sebe, nezakryjte si úplně textové okno Matlabu. Budeme rozpoznávat testovací promluvu `s2t`, o které víme, že obsahuje slovo 'dvě', ale rozpoznávač to neví.

```
figure(1); pause(1); soundsc([s2t s1]); D1=dtw (c1, c2t);
figure(2); pause(1); soundsc([s2t s2]); D2=dtw (c2, c2t);
figure(3); pause(1); soundsc([s2t s3]); D3=dtw (c3, c2t);
figure(4); pause(1); soundsc([s2t s4]); D4=dtw (c4, c2t);
```

Zobrazte si všechny vzdálenosti a vyberte tu nejlepší – ta bude indikovat rozpoznané slovo.

```
D1
D2
D3
D4
[m,mini]=min([D1 D2 D3 D4]);
disp(sprintf('Rozpoznano ======= %d ========\n', mini));
```

Úkoly

1. Odpovídá výsledek očekávání ?
2. Poznáte na optimálních srovnávacích cestách, které referenční sekvence obsahují hlásku 'e' ?

3.5 The big thing

Rozpoznáme vše a budeme se dívat na výsledky. Pro novou sekvenci zmáčkněte Enter. Grafická okna pro jednotlivé reference si nechte seřazená vedle sebe.

```
reco_dtw
```

Úkoly

1. Komentujte výsledky.
2. Myslíte si, že v reálných úlohách (šum, více řečníků, velký slovník, spojená slova) dostanete vždy 100% úspěšnost ? Odhadněte chybovost state-of-the-art systémů pro rozpoznávání spontánní angličtiny (telefonní hovory, meetingy).

4 Skryté Markovovy modely (HMM)

Tato část se zabývá trénováním a rozpoznáváním pomocí HMM. Naše modely můžeme charakterizovat takto:

- N stavů, z nichž pouze $N-2$ je "vysílacích", první a N-tý slouží jen ke vstupu do modelu a k výstupu z modelu.
- levo-pravá struktura: ze současného stavu můžeme jít jen do stejného stavu nebo do následujícího stavu. Matice přechodových pravděpodobností $\mathbf{A}$ má tedy nenulové prvky jen na hlavní a na jedné postranní diagonále.
- modelování vysílacích rozložení pravděpodobnosti pomocí P-rozměrných Gaussových rozložení:

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

  kde kovarianční matice $\mathbf{\Sigma}$ je diagonální. K jejímu popisu nám tedy stačí vektor směrodatných odchylek σ o velikosti P. Informace o všech vektorových středních hodnotách pro všechny stavy modelu shrneme do matice **MI** (velikost $P \times N$). Informace o všech vektorech směrodatných odchylek pro všechny stavy modelu shrneme do matice **SIGMA** (velikost $P \times N$). Uvědomme si, že první a poslední sloupec v těchto maticích nejsou na nic, neboť první a poslední stav modelu nejsou "vysílací".

Jako vstupní sekvence $\mathbf{O}$ (sekvence pozorování) budeme používat sekvence LPC-cepstrálních vektorů bez nultého koeficientu c_0 – stejné jako pro DTW.

4.1 Trénování modelů

pro trénování máte k disposici tyto funkce:

- `initemis.m` — inicializuje "vysílací" rozložení pravděpodobnosti. Vstupem je matice pozorování a počet stavů, výstupem jsou matice **MI** a **SIGMA**. Všechny vysílací stavy jsou nastaveny **úplně stejně** — na globální střední hodnoty a na globální směrodatné odchylky.
- `inittran.m` — inicializuje matici přechodových pravděpodobností. Vstupem je počet stavů, výstupem je matice **A**. Pravděpodobnost přechodu do dalšího stavu je nastavena na 0.5. Pravděpodobnost setrvání ve stavu je nastavena taktéž na 0.5.
- `reestim.m` — Baum-Welchova reestimace. Vstupem je sekvence pozorování, stará matice přechodových pravděpodobností **A**, stará matice středních hodnot **MI** a stará matice směrodatných odchylek **SIGMA**.

  Funkce provádí (rovnice jsou ve slajdech `09_hmm.pdf`):

  - odhad částečných dopředných likelihoodů $\alpha_j(t)$, tedy součet likelihoodů všech cest, které začínají v čase 1 a ve stavu 1 a v čase t jsou ve stavu j.
  - odhad částečných zpětných likelihoodů $\alpha_j(t)$, tedy součet likelihoodů všech cest, které končí v čase T a ve stavu N a v čase t jsou ve stavu j.
  - z alf a bet se spočítají "posteriorní pravděpodobnosti bytí ve stavu j v čase t" nebo "soft occupation counts"
    $$L_j(t) = \frac{\alpha_j(t)\beta_j(t)}{P(\mathbf{O}|M)},$$
    kde $P(\mathbf{O}|M)$ je celková Baum-Welchova likelihood (součet likelihoodů **všech** cest). Ta se dá zjistit jako poslední alfa: $\alpha_N(T)$ (poslední částečná dopředná likelihood je celá likelihood) nebo první beta: $\beta_1(1)$ (poslední částečná zpětná likelihood je celá likelihood).

    Hodnoty $L_j(t)$ se musí sumovat do jedničky pro každý vektor a pro všechny stavy.
  - přetrénování parametrů HMM pomocí vážených průměrů, kde hodnoty $L_j(t)$ jsou váhy jednotlivých vektorů.

  Výstupů je mnoho: nová matice přechodových pravděpodobností **A**, nová matice středních hodnot **MI** a nová matice směrodatných odchylek **SIGMA**, dále pak celková pravděpodobnost P_{tot}, matice dopředných pravděpodobností **ALFA**, matice zpětných pravděpodobností **BETA**, a matice pravděpodobnosti obsazení stavů (state occupation likelihood) **L**. Tři poslední výstupy není nutné uchovávat, využijeme je pouze, chceme-li vědět, co se uvnitř modelu dělo během reestimace. BW-reestimaci nejčastěji provádíme v několika iteracích.

Máme-li ve slovníku několik slov, musíme pro každé natrénovat jeden model, můžeme jej označit například M_i. Mějme na paměti, že každý model je popsán sadou tří matic: $\mathbf{A}_i$, $\mathbf{MI}_i$, $\mathbf{SIGMA}_i$.

4.2 Rozpoznávání

Na vstup rozpoznávače přijde *neznámá* posloupnost $\mathbf{O}_t$. My máme určit, ke kterému slovu patří. Pro všechny modely M_i, které máme k disposici, vyhodnotíme *Viterbiho pravděpodobnost* $\mathcal{P}^\star(\mathbf{O}_t|M_i)$ a vybereme tu největší. Model, který dal největší pravděpodobnost, odpovídá rozpoznanému slovu.

Pro rozpoznávání máte k disposici funkci `viterbi_log.m`. Vstupem je sekvence pozorování, matice přechodových pravděpodobností **A**, matice středních hodnot **MI** a matice směrodatných odchylek **SIGMA**. Výstupem je logaritmus Viterbiho pravděpodobnosti $\mathcal{P}^\star$ a vektor X udávající optimální sekvenci stavů pro danou sekvenci pozorování a daný model. Jelikož sekvence stavů musí vždy začínat prvním stavem a končit stavem N, obsahuje vektor X vždy o dva prvky více, než je vektorů v sekvenci pozorování.

4.3 Inicializace

Budeme pracovat s 9-ti stavovými modely. Určení optimálního počtu stavů je černá magie, k tomuto číslu jsme došli následovně:

- nejdelší slovo jsou "jedna" a "čtyři", která mají 5 hlásek
- před s po slově je obvykle ticho.

- potřebujeme ještě prví a poslední nevysílací stav.
- ... celkem 9.

Tento počet stavů použijeme pro **všechna** slova.
Budeme si hrát s trénovacím slovem "jedna". Nejprve inicializujeme model a podíváme se na vypočtené parametry:

```
N=9; A=inittran(N); [MI,SIGMA]=initemis(c1,N);
A
MI
SIGMA
```

Úkoly

1. Zkontrolujte, zda matice přechodových pravděpodobností definuje levo-pravý HMM, kde nejsou povoleny přeskoky stavů.
2. Všechny střední hodnoty a směrodatné odchylky jsou nastaveny na stejné hodnoty. Jak si myslíte, že budou vypadat "occupation counts" $L_j(t)$?

4.4 Baum-Welchovo přetrénování – 1. iterace

Ještě jednou: ze starých parametrů se spočítají matice $\alpha_j(t)$ a $\beta_j(t)$ (všechny vektory, všechny stavy), z nich matice $L_j(t)$ (všechny vektory, všechny stavy) a z nich se odhadnou nové parametry modelu. BW přetrénování také produkuje BW-likelihood, která určuje kvalitu, jak model "sedí" k datům.
Provedeme jedno přetrénování a prohlédneme si pár výsledků:

```
[NEWA, NEWMI, NEWSIGMA, Ptot, ALFA, BETA, L] = reestim (c1, A, MI, SIGMA);
ALFA
BETA
```

... numericky toto není moc ke koukání. Zajímavější bude zobrazit si hodnoty $L_j(t)$ a podívat se, zda se opravdu sumují do 1 pro všechny vektory.

```
T = size (c1,2); plot (1:T, L)
sum(L(2:N-1,:))
```

Úkoly

1. Co konstatujete o rozhození vektorů na trénování jednotlivých stavů ? Má to nějakou souvislost s fonémy ?

4.5 Další iterace Baum-Welchovo přetrénování

Na základě nově vypočtených parametrů vypočítáme opět $L_j(t)$ a z nich opět nové parametry:

```
[NEWA,NEWMI,NEWSIGMA,Ptot,ALFA,BETA,L] = reestim (c1, NEWA, NEWMI, NEWSIGMA); plot (1:T, L); Ptot
```

... aj aj. `Ptot` je nula, to je divné. Analyzujeme, co se stalo, podíváme se na $L_j(t)$:

```
L
```

a vidíme, že je plná not-a-numbers. Stalo se to, že během reestimace se hodnoty směrodatných odchylek natolik "zpřesnily", že zabránily další reestimaci i rozpoznávání (variance byla zřejmě někde 0). Proto budeme každé iteraci aktualizovat jen matice **A** a **MI**, matici **SIGMA** ponecháme původní (z inicializace).
Celý postup spustíme:

```
show_bw_iters
```

skript `show_bw_iters.m` si prohlédněte, za jeho běhu je nutné mačkat Enter.

Úkoly

1. Sledujte posuny funkcí $L_j(t)$, všímejte si, jak si funkce "najdou" smysluplné segmenty.
2. Sledujte, jak se pohybuje celková BW likelihood `Ptot` – navrhněte, jak přetrénovávání zastavit bez tvrdého nastavení počtu iterací.
3. Všímejte si numerických hodnot `Ptot` - dynamika v BW trénování je docela problém, i ve funkci `reestim.m` bylo nutné zavést normalizace, prohlédněte si ji.

4.6 The big thing

Natrénujeme modely pro všechna 4 slova (prohlédněte si `train_hmms.m`).

`train_hmms`

Výsledné parametry pro modely budou v maticích `Ax, MIx, SIGMAx`, kde `x` je identita slova.
Rozpoznávání je realizováno funkcí `viterbi_log.m`, podívejte se nejprve dovnitř, poté spusťte:

`reco_hmms`

Úkoly

1. Trénování modelů zobrazuje výsledné hodnoty funkcí $L_j(t)$. Podívejte se, zda si BW algoritmus opět sám našel smysluplné segmenty.
2. U jednoho ze stavů modelu '3' si povšimněte, že jedna z funkcí $L_j(t)$ je pouze jeden rámec úzká. Je to dobře ?
3. Zkuste se alespoň v jednom případě Viterbiho dekódování podívat na sekvenci stavů `X` (funkci `viterbi_log` musíte dát dva výstupní parametry. Co z ní dokážete zjistit ?