Viterbiho algoritmus pro rozpoznávání izolovaných slov

Jan Černocký a Lukáš Burget, FIT VUT Brno

April 20, 2005

Úkolem tohoto cvičení a následujícího projektu je napsat Viterbiho algoritmus pomocí "token passing" (pivo passing) a vytvořit jednoduchý rozpoznávač izolovaných slov ano/ne.
K disposici máte:

- modely natrénované pomocí HTK.
- HTK toolkit, kterým budete provádět parametrizaci řeči (tool `HCopy`).
- pomocné funkce pro načtení souboru s příznaky a pro vypočet hodnoty hustoty gaussovské rozložení - viz soubor `lukas.[ch]`
- main pro výpočet Viterbiho pravděpodobnosti.

1 Příprava

1.1 Modely

Modely jednotlivých slov máme již natrénované z minula – uložíme je do textových souborů s touto strukturou:

```
5
39
mean ..... mean
mean ..... mean
mean ..... mean
mean ..... mean
mean ..... mean
var ... var
var ... var
var ... var
var ... var
var ... var
matice prechod pravdepodobnosti ....
```

všimněte si, že se prakticky jedná o HTK soubory s vymazanými klíčovými slovy. Tento převod za Vás může zařídit Perlový skript:

```
htk2lukas_5states.pl ...cesta.../HTK/ANO_NE/hmm1/ANO > ano
htk2lukas_5states.pl ...cesta.../HTK/ANO_NE/hmm1/NE > ne
```

1.2 Práce se zvukovými soubory a soubory s parametry

- Vstupem jsou standardní zvukové soubory `*.raw` (8 kHz, 16 bitů).
- provedete parametrizaci pomocí `HCopy` s konfiguračním souborem `tool.conf` (12 MFCC koeficientů + energie, delta, deltadelta):
  `HCopy -C tool.conf soubor.raw soubor.mfc`
  `soubor.mfc` bude vstupem pro Viterbiho.

2 Vlastní Viterbi

- napsat ho bude Vaším úkolem.
- bude se volat následujícím způsobem:
  `viterbi model soubor.mfc`
- na standardním výstupu vypíše Viterbiho log-pravděpodobnost.

2.1 Popis balíku `viterbi.tgz`

Jsou pro Vás nachystány soubory `viterbi.c` (main), `viterbi_pivo.[ch]`, `Makefile` a pomocné soubory `common.[ch]`, které už jste používali. V souborech `lukas.[ch]` jsou užitečné funkce od Lukáše Burgeta:

```
int ReadHTKFeature(FILE * fp_in, float *in, size_t fea_len, int swap)
načtení jednoho či více příznakových vektorů parametrů
- fp_in   - pointer na soubor s parametry, který musí být otevřený pro čtení.
- in      - vektor, kam budeme parametry ukládat, musí být naalokovaný
           na fea_len floatů (to bude 4*39*pocet_rámců bajtů).
- fea_len - počet načítaných keficientů (39*pocet_rámců)
- swap    - Opět při volání nastavit na 1.
výstupní hodnota: 0 pokud ok, -1 pokud chyba.
```


```
int ReadHTKFeature(FILE * fp_in, float *in, size_t fea_len, int swap)
načtení jednoho či více příznakových vektorů parametrů
- fp_in   - pointer na soubor s parametry, který musí být otevřený pro čtení.
- in      - vektor, kam budeme parametry ukládat, musí být naalokovaný
           na fea_len floatů (to bude 4*39*pocet_rámců bajtů).
- fea_len - počet načítaných keficientů (39*pocet_rámců)
- swap    - Opět při volání nastavit na 1.
výstupní hodnota: 0 pokud ok, -1 pokud chyba.
```


```
float LogGaussPDF(float  *observation, float *means, float *variances, int size)
Výpočet LOG hodnoty hustoty pravděpodobnosti (neboli log vysílací
pravděpodobnosti vektoru stavem).
- observation - vektor parametrů (39 floatů)
- means - vektor středích hodnot (39 floatů)
- variances - vektor rozpytlů (39 floatů) (pamatujeme si, že pokud
   pracujeme s diagonálními kovar. maticemi, ukládáme jen rozptyly,
   které jsou na diagonále).
- size - velikost všech vektorů (39).
výstupní hodnota: logaritmus "vysílací pravděpodobnosti" b_j[o(t)]
```

Zjistíte, že soubor `viterbi_pivo.c` je poněkud vykuchaný – Vaším úkolem je dopsat sem Viterbiho.

2.2 Praktické poznámky k Viterbimu:

Je vhodné si definovat dva vektory, každý bude mít tolik buněk, kolik je v HMM stavů. V jednom budou hodnoty žetonů (piv) TEĎ, ve druhém hodnoty žetonů po posunu z času t do času $t+1$: POTOM.
Inicializace a ukončení algoritmu viz přednáška a soubor `viterbi_pivo.c`. Běžný posun piv vypadá takto:

- vektor POTOM nastavíte na velmi záporné hodnoty (pozor, ne na nuly, logaritmy mohou být běžně záporné !)
- každý žeton podle potřeby naklonujete (více připojených stavů), a "dolejete" příslušené logaritmy vysílací a přechodové pravděpodobnosti.
- umístíte na správné místo v POTOM, ale díváte se, zda už na tom místě něco není, pokud ano a je to větší, necháte tuto větší hodnotu. To je ekvivalentní "vyhubení méně plných piv".

Když jste s posunem hotoví, překopírujete POTOM do TEĎ.

2.3 Testování

V balíku `viterbi.tgz` jsou pro Vás nachystané modely `ano, ne` (můžete si je vygenerovat sami z těch, co jste v minulém cvičení natrénovali, pomocí skriptu `htk2lukas_5states.pl`) a dva testovací soubory: `cut_A10162A0.mfc` obsahuje "ano", `cut_A10162A1.mfc` obsahuje "ne". Spuštění Viterbiho by Vám mělo dát pro `cut_A10162A0.mfc` tuto likelihood:

```
viterbi  ano   cut_A10162A0.mfc
-2905.672852
```

V podadresáři `htktest` můžete tento výsledek srovnat s likelihood, kterou dá standardní dekodér `HVite` (je tam pár pomocných souborů, které `HVite` potřebuje, jako rozpoznávací síť, seznam modelů (v tomto případě model**u**

a velmi obsažný výslovnostní slovník):
`HVite -T 3 -d htktest -w htktest/ano_net htktest/ano_dico htktest/ano_list_models cut_A10162A0.mfc`
Výstup `ANO == [45 frames] -64.5705 [Ac=-2905.7 LM=0.0] (Act=3.7)` obsahuje za klíčovým slovem `Ac=` akustickou likelihood, což je právě ta, která nás zajímá.

3 Vlastní rozpoznávání

musíte srovnat Viterbiho pravděpodobnosti ze dvou modelů - podobně jako u DTW to můžete udělat v jednom programu (bude načítat dva modely), nebo v externím programu nebo skriptu.