Základní tón, kódování a dekódování řeči

Jan Černocký, FIT VUT Brno

1 Detekce základního tónu

Na úvod si ukážeme princip metod detekce základního tónu pomocí ACF (autokorelační funkce) a NCCF (normalizovaná cross-korelační funkce). V obou případech:

- zpožďujeme signál o možné hodnoty lagu (perioda základního tónu ve vzorcích).
- násobíme a sčítáme hodnoty pod sebou.
- ve výsledku pak hledáme maximum, pokud je toto maximum významné, tj. vetší než (práh $\times$ nultý autokorelační koeficient), prohlásíme rámec za znělý, poloha maxima je lag. Pokud významné není, je rámec neznělý.

Rozdíl je v tom, že ACF používá pouze hodnoty z jednoho rámce a "nevidí" jinam, NCCF používá při zpožďování historii signálu i z předchozího rámce (rámců).

Nejprve bychom si měli něco načíst, ustřednit a rozdělit na rámce. Budeme používat rámce bez překrytí. Je připraven oblíbený testovací signál `test.wav`, ale můžete použít i cokoliv vlastního. Pro začátek si také vybereme jeden rámec na hraní.

```
% signal
s = wavread ('test.wav');
sm = s - mean(s);
plot (sm); sound (sm);
sr = frame (sm, 160, 0); % no overlap !
% playing with one frame - 7th one is the nice one ...
x = sr (:,7);
plot (x);
```

1.1 ACF

Z vybraného rámce standardně spočítáme autokorelační koeficienty:

```
% pitch detection from pure ACF
R = xcorr (x);
R = R(160:end);
n = 0:159; plot (n, R);
```

musíme také definovat minimální a maximální lag a hodnotu prahu. Maximální lag je menší než 160, uvedených na přednášce, protože používáme pouze 160-tivzorkové rámce a hodnota 160 by nefungovala. Práh byl určen experimentálně. Všimneme si, že je nutné dávat pozor na Matlabovské indexování, které začíná od 1...

```
% some constants
Lmin = 20; Lmax = 146;
thr = 0.3; % maximum needs to be at least thr * R[0]
% detect lag
[Rmax,ii] = max(R((Lmin+1):(Lmax+1)));  % needs to add 1 because of Matlab indexing
if Rmax >= thr * R(1)  % R[0] in Matlab indexing
  L=ii+Lmin-1;         % and here needs to remove it again...
else
  L=0;
end
hold on; plot (L,Rmax,'or'); hold off;
```

Nyní se můžeme podívat na hodnotu lagu, základní periody v sekundách a základního frekvence (pitch) v Hz:

```
% show lag in samples
L
% in seconds
T0 = L / 8000
% and fundamental frequency in Hz
F0 = 1/T0
```

1.2 NCCF

U NCCF nebudeme pracovat s maticí rámců `sr`, ale nadefinujeme si ukazatele do signálu, které odpovídají vybranému 7. rámci. Ten vybereme, zkontrolujeme, že je to skutečně ten, se kterým jsme pracovali v předcházející sekci a spočítáme jeho energii – budeme ji potřebovat při normalizaci a stačí ji spočítat pouze 1x:

```
% NCCF - we choose the same portion of signal as 7th frame.
from = (7-1) * 160 + 1; to = from + 160 -1;
selected = sm(from:to);  % nonshifted frame
x - selected             % check that it is really the same one ...
E1 = sum(selected .^ 2); % energy of non-shifted frame.
```

V následujícím cyklu budeme pak definovat zpožděný rámec `shifted` a spočítáme pro něj NCCF. Všimněte si, že počítáme $NCCF[0]$ (nemuseli bychom, měl by být vždy jedna), pak "jedeme" pouze v oblasti zajímavých lagů. Cyklus nejprve nechte proběhnout se zobrazováním, mačkejte `Enter`, sledujte vztah `selected` a `shifted` na obrázku a vypočtený NCCF koeficient. Pak můžete tuto linku zakomentovat.

```
% in the following cycle, look at a few values, then comment out the plot and pause
Rnccf = zeros(1,Lmax + 1);
for n = [0 Lmin:Lmax]
  froms = from-n; tos = to-n; % indexes of the shifted frame
  shifted = sm(froms:tos);
  plot(1:160,selected,1:160,shifted); pause;
  E2 = sum(shifted .^ 2); % energy of the shifted one
  numerator = selected' * shifted;
  nccf = numerator / sqrt(E1 * E2)
  Rnccf(n+1) = nccf;  % Matlab indexing.
end
n = 0:Lmax; plot (n, Rnccf);
```

Nakonec opět detekujeme lag:

```
% detect lag
[Rmax,ii] = max(Rnccf((Lmin+1):(Lmax+1)));  % needs to add 1 because of Matlab indexing
if Rmax >= thr * Rnccf(1)  % R[0] in Matlab indexing
  L=ii+Lmin-1;             % and here needs to remove it again...
else
  L=0;
end
% show lag:
L
```

1.3 Detekce základního tónu pro celý signál, zlepšení kvality

Určení základního tónu musíme provést pro **všechny** rámce. Jsou pro Vás nachystány funkce `lag_acf` a `lag_nccf`. Podívejte se na jejich helpy. První je triviální a prakticky copy-pastem toho, co jste viděli výše. `lag_nccf` řeší výpočet koeficientů i na začátku signálu.

```
Lacf = lag_acf (sm,160,0,20,146,0.3);
Lnccf = lag_nccf (sm,160,0,20,146,0.7);
subplot(411); plot(sm); axis tight
subplot(412); plot(Lacf); axis tight
subplot(413); plot(Lnccf); axis tight
```

Vidíme, že odhad pomocí NCCF vypadá lépe než ACF, ale i tak jsme se nevyhnuli chybám (double lags). Pokusíme se situaci napravit pomocí mediánové filtrace:

```
Lnccf_med = medfilt1 (Lnccf, 5);
subplot(414); plot(Lnccf_med); axis tight
```

Nakonec vyzkoušíme detekci pro delší signál – `train.wav`, který jsme používali minule:

```
% testing on some longer signal ... train.wav
s = wavread ('train.wav');
sm = s - mean(s);
Lacf = lag_acf (sm,160,0,20,146,0.3);
Lnccf = lag_nccf (sm,160,0,20,146,0.7);
subplot(411); plot(sm); axis tight
subplot(412); plot(Lacf); axis tight
subplot(413); plot(Lnccf); axis tight
Lnccf_med = medfilt1 (Lnccf, 5);
subplot(414); plot(Lnccf_med); axis tight
```

Tento výsledek Vám jasně ukazuje, že detekce základního tónu není triviální úloha a poprvé se u ní setkáme s faktem, že pro lidskou řeč prostě někdy algoritmy selhávají :-(

Úkoly

1. Zkontrolujte, zda je $NCCF[0]$ skutečně vždy 1.
2. Prostudujte, jak je ve funkci `lag_nccf` řešen výpočet na začátku signálu.
3. Jak byste NCCF řešili při on-line zpracování, kdy nemáte k disposici celý signál ?
4. Prahy 0.3 pro ACF a 0.7 pro NCCF jsou velice hrubě nastřelené. Dosáhnete lepších výsledků, když je trochu poladíte ?
5. Vyzkoušejte jiné délky mediánového filtru. Co se dá říci o výsledku ?
6. Pro experty: napište v Matlabu funkci, která "zahraje" průběh základního tónu v celé promluvě. Stačí pomocí jedné cosinusovky, která bude mít normovanou frekvenci rovnou $\frac{1}{lag}$. Pokud je $lag = 0$, hrajte ticho. POZOR, musíte zachovat kontinuitu fáze u cosinusovek !
7. Pro experty: přidejte k signálu šum a detekujte základní tón. Jak vypadají výsledky ?

2 Kódování

Naším cílem bude vyrobit ZRE-super kodér řeči, který bude pracovat na bit-rate 1050 bit/s. Nebudeme se zabývat tvorbu bitového streamu, výstupem kodéru bude textový soubor, kde budou celá čísla – o těch budeme vědět, že se dají zakódovat patřičným počtem bitů.

2.1 Výpočet parametrů a kódování bez kvantování

Na základě funkce `param.m` z minula je pro Vás připraven funkce `param2.m`, která provede LPC analýzu i odhad základního tónu pro všechny rámce. Prostudujte její help. Pro syntézu je nachystána funkce `synthesize.m`. Opět se podívejte na její help. Nejprve budeme analyzovat a syntetizovat **bez** kvantování parametrů:

```
s = wavread ('train.wav');
sm = s - mean(s);
% first parameterization function:
[A,G,L,Nram] = param2 (sm,160,0,10,20,146,0.7);
% and synthesis
ss = synthesize (A,G,L,10,160);
plot(ss); soundsc(ss);
```

Úkoly

1. Prostudujte, jak je ve funkci `synthesize.m` zajištěno, že buzení má před násobením gainem vždy jednotkovou energii (výkon) na jeden vzorek.
2. Prostudujte, jak se ve funkci generuje znělé buzení – co je v proměnné `nextvoiced` ?
3. Fungovala by funkce v případě, že by maximální možný lag přesahoval délku rámce ?
4. Doplňte o zobrazení toho, jak vypadá buzení a výsledný rámec — přidejte `plot(...); pause` a prohlédněte si je.
5. Buzení je ve znělých rámcích realizováno nejblbější možnou metodou – jednotkovými impulsy. Dalo by se udělat něco inteligentnějšího ?
6. Jak myslíte, že se s buzením pracuje v GSM kodecích ?

2.2 "Ostré" kódování

Hodnoty lagu jsou 0,20...126, což je výhodné – můžeme je kvantovat 7-mi bity. Hodnoty koeficientů a gainu jsou prozatím reálná čísla ve formátu double, což by znamenalo, že na jeden rámec potřebujeme:
`7 + 64 * 10 + 64 * 1` bitů. To je příliš mnoho.

Ke **kvantování koeficientů filtru** použijeme vektorové kvantování. Detaily byly v minulé laboratoři, dnes je pro Vás nachystána kódová kniha o 210ti kódových vektorech: `cb210.txt`[1] Tímto krokem snížíme počet bitů na koeficienty na jeden rámec na `8`.

Ke **kvantování gainu** použijeme skalární kvantování na 64 úrovních. Potřebnou kódovou knihu máte v `gcb64.txt`. I když se jedná o skaláry, můžeme použít naprosto stejné funkce `vq_*` jako pro vektory[2]. Potřebný počet bitů bude `6`.

Upozorňuji, že se jedná o školní příklad – obě dvě kódové knihy byly natrénovány na mluvčím, který nahrál soubor `train.wav`. Budou tedy uspokojivě fungovat jen pro tohoto mluvčího, pro jiné to bude bída...

```
load cb210.txt
load gcb64.txt

s = wavread ('train.wav');
sm = s - mean(s);
% first parameterization function:
[A,G,L,Nram] = param2 (sm,160,0,10,20,146,0.7);
[asym,gd] = vq_code(A, cb210);
gsym = vq_code(G, gcb64);
% decoding
Adecoded = cb210(:,asym);
Gdecoded = gcb64(:,gsym);

% and synthesis
ss = synthesize (Adecoded,Gdecoded,L,10,160);
plot (ss); soundsc(ss);
```

Úkoly

1. Ověřte, že je bit-rate skutečně 1050 bit/s. Řešení: `(8 + 7 + 6) * 50` .
2. Srovnejte tento bitový tok s kodeky GSM.
3. Proč má kódová kniha na LPC koeficienty velikost 210 a ne 256 ? Podívejte se do `hrani.m`.
4. Jak byla vytvořena kódová kniha pro gainy ?

[1] Na detaily její tvorby se můžete podívat do `hrani.m`, jsou použity funkce `vq_*` jako minule.
[2] I na tvorbu skalární kódové knihy se můžete podívat do `hrani.m`

5. Je výsledná řeč srozumitelná ?

6. Je výsledná řeč přirozená ?

7. Jakou byste tomuto kodéru dali známku MOS (mean opinion score – viz přednáška o kódování) ?

2.3 Kódování jiných signálů

Jako konečný výstup této laboratoře uzavřeme kompletní kódování a dekódování do dvou funkcí. Prostudujte jejich helpy !

- `coder` vezme WAV soubor a udělá z něj textový soubor, kde na každém řádky budou údaje pro 1 rámec: index do kódové knihy LPC koeficientů (8 bitů), index do kódové knihy gainů (6 bitů) a lag (7 bitů). Jako pomocný výstup pro Matlab tato funkce produkuje přečtený ustředněný signál.

- `decoder` z parametrů v textovém souboru dekóduje signál a uloží ho do WAV. Jako pomocný výstup pro Matlab tato funkce produkuje dekódovaný signál.

Nejprve budeme kódovat signál `testspdat.wav` od stejného mluvčího. Nejprve si poslechněte syntetizovaný signál, zhodnot'te, zda mu rozumíte, pak teprve originál:

```
s = coder('testspdat.wav', '/tmp/koko');
ss = decoder ('/tmp/koko', '/tmp/testspdatd.wav');
soundsc(ss); pause; soundsc (s);
```

Druhý test je s jiným mužským mluvčím:

```
s = coder('ses0019.wav', '/tmp/koko');
ss = decoder ('/tmp/koko', '/tmp/testspdatd.wav');
soundsc(ss); pause; soundsc (s);
```

A konečně test s nahrávkou od ženy:

```
s = coder('ses0099.wav', '/tmp/koko');
ss = decoder ('/tmp/koko', '/tmp/testspdatd.wav');
soundsc(ss); pause; soundsc (s);
```

Úkoly

1. Je výsledná řeč srozumitelná ?

2. Je výsledná řeč přirozená ?

3. Jakou byste tomuto kodéru dali známku MOS (mean opinion score – viz přednáška o kódování) ?

4. Prohlédněte výstupní soubor a zkontrolujte, zda tam opravdu není nic nad 1050 bit/s.

5. Rozpitvejte funkce `coder` a `decoder` – doplňte zobrazování původních a dekódovaných parametrů, a pokuste se zjistit, v čem je hlavní problém s kvalitou.

6. Ověřte stabilitu všech filtrů, jejichž koeficienty jsou uloženy v kódové knize `cb210.txt`.

7. Zkuste kódovat a dekódovat nějaký vlastní signál (nahrajte nebo stáhněte z netu, pokud není 8 kHz, 1 kanál a 16 bitové vzorky, převed'te pomocí `sox`).

8. Navrhněte, jak kódování zlepšit !

9. Pro experty: natrénujte vlastní kódovou knihu - více vektorů, více mluvčích, zastoupení mužů/žen, atd. Dejte pozor na to, aby všechny kódové vektory nepokryly **ticho**. Při výběru trénovacích dat je potřeba ticho odstranit – ručně nebo jednoduchým detektorem řečové aktivity (voice activity detector – VAD) založeným třeba na krátkodobé energii – jeho přesnost tady nehraje moc velkou roli.

10. Pro experty: vytvořte funkci, která srovná dekódovaný signál s kódovaným pomocí průměrné logaritmické spektrální vzdáleností (logarithmic spectral distance – rovnice 2 na straně 11 první přednášky o kódování). Spektrální hustotu výkonu spočítáte pomocí DFT. Průměrujte přes všechny rámce a zjistěte také maximální hodnotu přes všechny rámce.

2.4 Varování

Sekce o dekódování pečlivě čtěte, v rámci projektu 1. budete takový dekodér implementovat v C ☺.