ZRE 01 - ZAHŘÍVACÍ CVIČENÍ

Jan Černocký, FIT VUT Brno

Cílem tohoto cvičení je presentovat základní operace se zvukem, především v Matlabu. Toto i další cvičení budou probíhat pod operačním systémem LINUX. Matlab se Vám rozběhne i pod Windous, ale některé mimo-Matlabové utilitky (`ws`, `sox`) nejsou ve standardní FIT distribuci Windows.

Dobrá rada

vřele doporučuji otevřít si pro každé počítačové cvičení okně **textový soubor** (Emacs, vi, kate, ...), vše zapisovat do něj, a příkazy do Matlabu, případně do příkazové řádky shellu (pro další utilitky) kopírovat pomocí copy and paste. Po ukončení cvičení Vám pak zbyde přesný záznam toho, co jste dělali, a co (snad) chodilo.

1 Jak zvuk nahrát a rychle se na něj podívat - WaveSurfer

Pro hraní se zvukem budeme v tomto kursu vyžadovat ten nejjednodušší možný formát: 1 kanál, vzorkovací frekvenci 8 kHz, kvantování 16 bit lineárně. Zvuk si můžeme buď nahrát nebo odněkud stáhnout a zkonvertovat. Pro oba účely použijeme WaveSurfer: `ws`[1].

1.1 Nahrávání z mikrofonu

Ve vedlejším terminálu je vhodné si otevřít mixér: `alsamixer`, který se ovládá šipkami a klávesou 'M' (mute). Vstup z mikrofonu bývá někdy označen jako 'Mic', někdy jako 'Capture' – "vytáhněte" raději oba dva tyto vstupy. Bude pravděpodobně nutné zapnout i zesílení mikrofonu (MicBoost) — nesmí být 'M'.

Ve `ws` označte v 'preferences', že hodláte nahrávat se vzorkovací frekvencí 8000 Hz a s kvantováním na 16 bitů. Pak otevřete nový zvukový soubor a něco nahrejte. Zkontrolujte přehrátím. Uložte ve formátu WAV a RAW (bez hlavičky).

1.2 Konverze z něčeho

Na webu jsou k disposici tuny řeči (např. v PodCastech), k převodu je možné opět použít WaveSurfer. Příklad:
`ws ws ESLPod451.mp3`

- vyberte prvních 11 sekund, COPY.
- otevřete nový soubor, PASTE.
- Upravte vzorkovací frekvenci, počet kanálů a rozlišení v menu `Transform -> Convert`.
- Uložte ve formátu WAV nebo RAW.

Pokud nemáte mikrofon, můžete použít soubory `test.l16` (RAW) a `test.wav`.

2 Matlab - základní poznámky

- budeme pracovat s Matlabem (později i s C) v Linuxu.
- Matlab spustíte z terminálu: `matlab`. Pokud chcete konzolovou verzi, která nevolá Java-mašinku (doporučuji):
  `matlab -nojvm` nebo
  `matlab -nodesktop`
- v Matlabu můžete zadávat příkazy pouze, je-li kursor na příkazové řádce »
- Obsahuje-li řádek znak `%`, bere se jeho část za tímto znakem jako *poznámka* a není Matlabem prováděna.
- v Matlabu je možné se k předchozím příkazům vracet pomocí ↑. Návrat může být i "selektivní", např. k již použitým příkazům `plot` se můžete vrátit pomocí:
  `pl` ↑ ↑ ↑ ...

[1] dostupný i pro Windous z KTH v Kodani: `http://www.speech.kth.se/wavesurfer/`

- Matlab obsahuje velmi kvalitní on-line nápovědu: `help funkce`

Pro toho, kdo Matlab v životě neviděl, viz 1. laboratoř ze signálů a systémů: `http://www.fit.vutbr.cz/~cernocky/sig` sekce 1–3.

3 Načtení řečového signálu do Matlabu

. . . je velmi jednoduché. Vzorky budou v Matlabu uloženy jako řádkový vektor. Soubory RAW (1 vzorek $\sim$ 1 short o 16 bitech) čteme podobně jako v C (všimněte si "vektorového chování" funkce `fread`).

```
ff=fopen('test.l16','r');
sig=fread(ff, [1 inf],'short')/32768;
fclose(ff)
plot(sig);
```

Všimněte si, že vzorky normalizujeme tak, aby ležely v intervalu ± 1. U souboru RAW musíme vždy předem vědět, jaký je formát vzorků a jaká je vzorkovací frekvence.

Formát WAV tyto informace obsahuje v hlavičce a čte se ještě jednodušeji:

```
sw=wavread('test.wav'); sw=sw';
[sw,Fs,Nbits] = wavread('test.wav'); sw=sw';
```

Druhé volání nám poskytne i informace z hlavičky. Všimněte si, že výsledek je nutné transponnovat, protože funkce wavread má na výstupu sloupcový vektor. Tato funkce sama normalizuje vzorky do intervalu ± 1.

Úkoly

1. Podívejte se na načtené signály: `plot(sig); plot(sw);`, vypadají stejně, že ?
2. Zkontrolujte, zda hlavička WAV obsahovala správné hodnoty vzorkovací frkvence a počtu bitů.
3. Srovnejte, zda jsou signály opravdu stejné: `ws - sig` nebo `plot(ws - wsig);`
4. Přehrejte si signál:
   `sound(ws)`
   `sound(50*ws)`
   `soundsc(50*ws)`
   Proč druhé přehrátí škrčí ? Proč třetí už zase ne ? Kdy budeme používat funkci `sound` a kdy `soundsc` ?

4 Střední hodnota a ustřednění

Důležitou operací je výpočet a zničení stejnosměrné složky v řeči (dc-offset). Jedná se o střední hodnotu signálu. Pokud máme k disposici celý signál, je výpočet a ustřednění jednoduché:

```
mean(sw)
% ustredneni
sc = sw - mean(sw);
% kontrola
mean(sc)
```

V případě, že signál zpracováváme on-line, nemůžeme čekat, až skončí. V tomto případě se počítá tzv. "klouzavý odhad" střední hodnoty, pro n-tý vzorek:

$$m[n] = k\, m[n-1] + (1-k)s[n],$$

kde k je konstanta blízká 1.

```
k = 0.99;
N = length(sw);
sconline = zeros(1,N);
mntolook = zeros(1,N);
mn_1=0;
for ii=1:N,          % ... radeji nepouzivat 'i' jako ridici promnennou
                     % komplexni jednotka :-(
  mn = k * mn_1 + (1-k) * sw(ii);
  sconline(ii) = sw(ii) - mn;
  mntolook(ii) = mn;
  mn_1 = mn;
end
```

Cykly jsou v Matlabu velmi pomalé a nepraktické, ale jinak to v tomto případě nejde...

Úkoly

1. Srovnejte původní a ustředněný signál: `plot (1:N,sw,1:N,sconline)`
2. Podívejte se na průběh odhadnuté střední hodnoty: `plot (1:N,mntolook);`
3. ... a srovnejte ji s předchozím odhadem: `mean(sconline)`
4. Proč hodnoty on-line odhadu kmitají okolo správné hodnoty ? Jak byste tyto kmity utlumili ? Jakou by to mělo nevýhodu ?

5 Výběr a uložení signálu

Výběr signálu uskutečníme tak, že zadáme hraniční vzorky, např: `x = sw(1000:2000);`
Do formátu WAV ukládáme: `wavwrite(x,8000,16,'vyrez.wav');`
Do formátu RAW:

```
ff=fopen('vyrez.l16','w');
fwrite(ff, round(x * 32768), 'short');
fclose(ff)
```

Všimněte si nutnosti de-normalizace, ukládáme celá čísla — `short`'s.

Úkoly

1. Vyberte znělý úsek, poslechněte si jej pomocí `sound(x)`. Pak opakujte s neznělým a s přechodovým úsekem (plozíva). Poznáte z krátkého úseku, o kterou hlásku se jedná ?
2. Uložte vybraný úsek ve WAV a RAW a pokuste se načíst do WaveSurferu. Vše v pořádku ?

6 Dělení signálu na rámce

Dozvíme se, že díky tzv. kvazistacionaritě řeči nemůžeme téměř nic dělat se signálem jako celkem, ale musíme dělit na kratší úseky — tzv. **rámce** (frames). Typická délka je 20 ms a překrytí 10 ms. Výpočet délky rámců ve vzorcích:

```
l = 20e-3; r = 10e-3;
ls = round (l * Fs)
rs = round (r * Fs)
```

A dělení na rámce, které budou ve sloupcích matice `swram`:

```
% frame shift
fs = ls - rs;
%Pocet ramcu - bez odvozeni (ale jde na to prijit) - bude na prednesu
Nram = 1 + floor((length(sw) - ls) / fs)
% alokujeme pro ramce - budou ve sloupcich
swram = zeros(ls, Nram);
% a naplnime:
odtud=1; potud=ls;
for ii=1:Nram,
  ramec = sw(odtud:potud)';
  swram(:,ii) = ramec;
  odtud = odtud + fs;
  potud = potud + fs;
end
```

Úkoly

1. Podívejte se na znělý a neznělý rámec (např pro test.l16 13-tý a 100-tý):
   `plot (swram (:,13));`
   `plot (swram (:,100))`
2. Poslechněte si tyto rámce — stačí Vám 20 ms na rozpoznání hlásky ?
3. Uzavřete dělení na rámce do funkce — budeme ji potřebovat. Umístíte-li tuto funkci do svého `$HOME/matlab`, "uvidí" na ni Matlab odkudkoliv.

7 Spektrogram

je oblíbenou pomůckou pro analýzu řeči. Zatím bez teorie, pouze srovnání se signálem:

```
subplot(211);
plot(sw); axis tight;
subplot(212);
specgram(sw,256,Fs);
```

Úkoly

1. Proč jsou použita nastavení os `axis tight` ?
2. Jakým zvukům odpovídá vysoká/nízká energie na vyšších/nižších kmitočtech ?

8 Generování signálu, filtrace, frekvenční analýza

V této části budeme pracovat s uměle vygenerovaným signálem. Na vzorkovací frekvenci $F_s = 8000$ Hz vygenerujeme směs dvou harmonických signálů na frekvencích $f_1 = 440$ Hz a $f_2 = 1$ kHz. Využijeme přepočtu skutečných frekvencí na normované:

$$f' = \frac{f}{F_s}$$

a toho, že cosinusovka se generuje:

$$x[n] = C\cos(2\pi f' n)$$

Signál bude trvat 2 sekundy.

```
% generovani signalu - smes 2 cosinusovek, trvani 2 sec.
Fs = 8000; f1 = 440; f2=1000;
n = 0:15999;
s = 0.49*cos(2 * pi * f1/Fs * n) + 0.49*cos(2 * pi * f2/Fs * n);
plot (s); soundsc(s)
```

Budeme frekvenčně analyzovat prvních 1024 vzorků tohoto signálu. Všimněte si, že se díváme jen na interval frekvencí do $F_s/2$ (horní polovina je symetrická a nezajímá nás).

```
% frekvencni analyza - vybereme 1024 vzorku
x = s(1:1024);
f = (0:511) / 1024 * Fs;
X = fft(x); X = X(1:512);
subplot (211); plot(f,abs(X)); subplot (212); plot(f,angle(X));
```

8.1 Filtrace

Z tohoto signálu chceme vybrat pouze složku na 1 kHz. V Matlabu máme pro návrh IIR filtrů k disposici výkonnou funkci `ellip` (s pomocnou `ellipord`). Podívejte se do jejich helpů. Navrhovaný filtr bude pásmová propust'. Pozor: zatímco všude se normalizuje vzorkovací frekvencí, u matlabovksých funkcí pro návrhy filtrů normalizujeme Nyquistovou frekvencí ($F_s/2$) !

```
% navrh filtru tak, aby vybral 1 kHz:
Fs2 = Fs / 2;  % timto budeme normalizovat pro navrh filtru
Wp = [900/Fs2 1100/Fs2];
Ws = [800/Fs2 1200/Fs2];
Rp = 3; Rs = 20;
[N, Wn] = ellipord (Wp, Ws, Rp, Rs)
[B,A] = ellip(N,Rp,Rs,Wn)
```

Prohlédneme frekvenční charakteristiku a rozložení nul a pólů navrženého filtru:

```
% koukani na frekv char
freqz(B,A,512,8000);
% a na poly a nuly
zplane (B,A)
```

. . . a budeme filtrovat signál:

```
% filtrovani signalu
sf = filter (B,A,s);
plot (sf); soundsc(sf)
```

Úkoly:

1. Výstupní signál si přehrejte a frekvenčně analyzujte (opět 1024 vzorků). Zbavili jsme se 440 Hz ?
2. Ve spektru výstupu možná objevíte zbytek čáry na 440 Hz. Ověřte, že tato čára je utlumena přesně podle frekvenění charakteriskty filtru
   - odečtěte z $H(f)$ přenos na 440 Hz v dB.
   - převed'te z dB zpět na normální číslo (jsme v amplitudách, takže $10^{dB/20}$).
   - vynásobte tímto číslem výšku původní čáry. Tuto velikost byste měli vidět na 440 Hz ve spektru výstupu.

9 Frekvenční analýza řeči

Nahrejte si řečový signál na $F_s = 8000$ Hz. Ustředněte (do proměnné `sm`) a rozdělte jej na rámce o délce 160 vzorků bez překrytí (matice `sr`). Pokud nefunguje Vaše funkce, použijte přiloženou `frame.m`.

```
sw=wavread('test.wav'); sw=sw';
sc = sw - mean (sw);
sr = frame (sc, 160, 0);
```

V signálu vyberte jeden znělý rámec (nejlepší je zoomovat na waveformu signálu, odečíst číslo vzorku, podělit 160 a přičíst 1 – tak dostanete příslušné číslo rámce). Vyberte jej (např. pro rámec 10):

```
x = sr(:,10);
```

Podle přednášek provedeme frekvenční analýzu:

1. se zobrazením plného pásma od 0 do F_s.
2. se zobrazením od 0 do $F_s/2$.
3. s doplněním nulami (zero-padding) pro zvýšení rozlišení ve frekvenci.
4. spektrální hustotu výkonu.
5. její logaritmickou verzi.

```
% frekvencni analyza + spek. hustota vykonu - podle prednasky.
f = (0:159) / 160 * Fs; X = fft(x);
subplot (211); plot(f,abs(X)); subplot (212); plot(f,angle(X));

% jen polovina
f = (0:79) / 160 * Fs; X = fft(x); X = X(1:80);
subplot (211); plot(f,abs(X)); subplot (212); plot(f,angle(X));

% zero padding
f = (0:511) / 1024 * Fs;
X = fft([x' zeros(1,1024-160)]); X = X(1:512);
subplot (211); plot(f,abs(X)); subplot (212); plot(f,angle(X));

% spek hustota vykonu + zero padding
f = (0:511) / 1024 * Fs
X = fft([x' zeros(1,1024-160)]); X = X(1:512); Gdft= 1/160 *abs(X) .^ 2;
subplot (111); plot(f,Gdft);

% totez, ale logaritmicky v dB:
Gdftlog = 10 * log10 (Gdft);
subplot (111); plot(f,Gdftlog);
```

Na přednášce se dozvíte, že spektrum řeči je určeno základním tónem (jemná struktura) a postavením artikulačního ústrojí (hrubá struktura). Vidíte je na spočítaných spektrech?

10 Hand-made spektrogram

V sekci 7 jsme pouze jako 'black-box' spouštěli funkci `specgram`. Nejedná se o nic jiného než o odhad spektrální hustoty výkonu pro jednotlivé rámce – viz `help specgram`. Budeme schopni si takový spektrogram spočítat ručně:

```
Nr = size(sr,2); % pocet ramcu
% alokujeme pro specgram (jednotliva spektra budou ve sloupcich)
sg = zeros (512,Nr);
for n = 1:Nr,
  x = sr(:,n); %plot(x); pause;
  X = fft([x' zeros(1,1024-160)]); X = X(1:512); Gdft= 1/160 *abs(X) .^ 2;
  sg(:,n) = Gdft';
end
imagesc (10 * log10(sg)); axis xy;
```

Úkoly

1. Poslední řádka je zobrazení – zjistěte, proč musíme použít funkci `axis`.
2. Do druhého obrázku vypočtěte spektrogram pomocí matlabovské funkce:
   `figure(2); specgram(sm)`
   Zjistěte, zda se liší. Pokud ano, zkuste si "pohrát" s dalšími parametry funkce `specgram` tak, aby byl stejný (hodnoty na osách neřešte...). Parametry funkce `specgram` jsou následující:
   `B = specgram(A,NFFT,Fs,WINDOW,NOVERLAP)`
   Řešení: `figure(2); specgram(sc,1024,8000,boxcar(160),0)`

3. Zkuste modifikovat parametry spektrogramu tak, aby
   - byly dobře vidět násobky základního tónu.
   - aby bylo dobré časové rozlišení.
   - jde udělat obojí zároveň ?
4. Představte si, že potřebujete vložit spektrogram do černobíle vytištěného článku. Vygenerujte pěkný černobílý spektrogram. Zkuste použít něco jako:

```
c = colormap ('gray');
colormap (flipud(c))
brighten(magicka_konstanta)
```

... nebo zkuste použít funkci `contrast`.