Vektorové kvantování

Jan Černocký, FIT VUT Brno

March 30, 2005

Vektorové kvantování (Vector Quantization – VQ) se ve zpracování řeči používá např. při redukci bitového toku v kódování či při práci s diskrétními HMM. Spočívá v tom, že P-rozměrné vektory parametrů $\mathbf{x}$ převedeme na **symboly** pomocí **kódové knihy** o L kódových vektorech:

$$\mathbf{Y} = \{\mathbf{y}_i; 1 \leq i \leq L\}. \tag{1}$$

Při VQ-**kvantování** přiřadíme vektoru $\mathbf{x}$ kódový vektor $\mathbf{y}_i$ s minimální vzdáleností:

$$q(\mathbf{x}) = \mathbf{y}_i \quad \text{pokud} \quad d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j), \quad j \neq i, \ 1 \leq j \leq L. \tag{2}$$

Vzdálenost $d(\cdot, \cdot)$ může být libovolná z dříve probíraných; zde použijeme klasickou kepstrální míru, se kterou jsme se setkali již u DTW. **Globální vzdálenost** je dána součtem všech minimálních vzdáleností, normujeme počtem kvantovaných vektorů. Vektory, které jsou kvantovány jedním kódovým vektorem, tvoří tzv. **cluster**.
Kódovou knihu a-priori neznáme a musíme ji natrénovat na souboru trénovacích vektorů. Používáme následující postup:

1. Inicialisace L kódových vektorů.
2. První kvantování trénovacích vektorů podle rovnice 2.
3. "Přeučení" kódových vektorů – výpočet **centroidů** jednotlivých clusterů. V případě kepstrální míry jsou centroidy definovány jako vektorová střední hodnota všech trénovacích vektorů v clusteru. Centroidy nahradí původní kódové vektory.
4. Kvantování trénovacích vektorů novou kódovou knihou.
5. Návrat ke kroku 3. pokud
   - jsme nedosáhli maximálního povoleného počtu iterací.
   - nebo se globální vzdálenost ještě podstatně mění.

Inicialisace je důležitým krokem v učení kódové knihy. Používají se tyto postupy:

1. Náhodná inicialisace: ze souboru trénovacích vektorů se náhodně vybere L vektorů.
2. Inicialisace z menší kódové knihy "štěpením" kódových vektorů. Každý kódový vektor knihy o velikosti L se rozdělí na dva nové tak, že se původní kódový vektor "posune" ve dvou opačných směrech. Vznikne tak nová kódová kniha o velikosti $2L$. Tímto způsobem se dají vytvářet pouze knihy o $L = 2^g$ vektorech, kde g je celé číslo. První generace ($L = 1$) se dá jednoduše spočítat jako vektorová střední hodnota **všech** trénovacích vektorů.

1 Příklady

1.1 Příprava

Budeme trénovat VQ kódovou knihu pro kódování LPC-koeficientů (v praxi se běžně používají spíš, LAR, LSF nebo PARCOR, ale toto je školní příklad). Pro trénování máme k disposici dlouhý řečový soubor `speech.raw`, pro testování budeme používat oblíbený `test.l16` ("létající prase").

Proveďte LPC-parametrizaci (10 koeficientů LPC, bez a_0) obou souborů, parametry rámců jako vždy - délka 240, frame-shift 80. Můžete pustit souborek `param.m`

1.2 Kódová kniha VQ - trénování s náhodnou inicializací

Pro práci s VQ máte na k disposici tyto funkce:

- `vq_code.m` – kóduje vektory pomocí zadané kódové knihy.
- `vq_clust.m` – z trénovacích vektorů a příslušnosti ke clusterům spočítá nové centroidy.
- `vq_split.m` – provádí štěpení kódové knihy.

Okopírujte si je a prostudujte jejich helpy (tentokrát existují !!).

Natrénujte kódovou knihu s $L = 8$ a s náhodnou inicialisací. Zobrazujte globální vzdálenost během trénovacích iterací. Snižuje se ? Kvantujte vektory z testovacího souboru. Jaké jste dosáhli globální vzdálenosti na testovacím souboru ? Viz souborek `train_random_init.m`. Výsledná kódová kniha je v proměnné `cb` (každý sloupec je jeden kódový vektor).

1.3 Zobrazení a kódování testu.

Zobrazte clustery a centroidy ve 2D (např. v rovině a_1, a_2) a 3D (např. v prostoru a_1, a_2, a_3). Pro každý cluster/centroid použijte jinou barvu. Můžete použít souborek `show.m`, který zobrazuje ve 3D (pro 2D vynecháte jednu souřadnici). Centroid je označen velkým kolečkem, trénovací vektory jsou malé křížky. Pohrejte si s koeficienty, na které se "díváme" – na lince `i1=1; i2=2; i3=3;` dáte do proměnných prostě jiné indexy než 1,2,3. ZKuste různé zoomvání a rotování obrázku.

Zakódujte pomocí natrénované kódové knihy vektory z testovacího souboru. Jaké jste dosáhli globální vzdálenosti ?

1.4 Linde-Buzo-Gray

Natrénujte kódovou knihu s $L = 8$ postupným štěpením: $1 \rightarrow 2 \rightarrow 4 \rightarrow 8$. Pro štěpení kódové knihy můžete použít funkci `vq_split.m`.

Zobrazte výsledek (v souborku `show.m` musíte použít správnou kódovou knuihu) a opět zakódujte pomocí natrénované kódové knihy vektory z testovacího souboru. Dosáhli jste lepší globální vzdálenosti než v případě náhodné inicializace ?

1.5 Jak to bude hrát ?

Natrénujte kódovou knihu o velikosti $L = 64$ (pozor, bude to chvilku trvat, kdybyste se nemohli dočkat, použijte už natrénovanou `cb64.mat` - do matlabu ji dostanete pomocí `load cb64`, proměnná s kódovou knihou bude mít název `cb64`).

Spusťe Váš kodér a spočítejte LPC-koeficienty, pitch a energii. Nacucněte soubor s parametry do Matlabu, zakódujte LPC vektory pomocí Vaší kódové knihy a vylejte zpět do bin. souboru. Dekódujte a zahrajte si výsledek! Pokud ještě nemáte hotový dekodér v C, můžete použít i dekodér z Matlabu `dekoder.m`. (viz minulá labina) s ní vektory ze souboru `test.l16`. Může to vypadat asi takto:

```
% ... pusteni kodovani, vysledek v souboru out.bin ...

P = 10;
ff = fopen ('/tmp/out.bin','r');
inp = fread (ff,[P+2 inf],'float');
fclose (ff);
% get its size
Nfr = size (inp,2);
% and separate it into LPC, energy, LAGS
A = inp(1:P,:);
E = inp(P+1,:);
LAGS = inp(P+2,:);

%%% VQ coding %%%
[sym, gd]=vq_code(A, cb64);
Adecoded = cb64(:,sym);

%%% put it together and save - let pitch and energy as they are
aux = inp;
aux (1:P,:) = Adecoded;
% write it out
ff = fopen ('/tmp/outdec.bin','w');
fwrite (ff,aux,'float');
fclose (ff);

% ... na outdec.bin se zavola dekoder ...
```

Načtěte dekódovaný soubor do Matlabu. Zkuste zjistit, ve kterých místech je signál přebuzen. Není to náhodou tím, že jsme dostali nestabilní filtr $\frac{1}{A(z)}$? Jak byste to zjistili z kvantovaných koeficientů v matici `Adecoded` ?