DSP 2006 fall 期中考解答

1. 相較於過去的TTS synthesis使用獨立的聲音單位互相連結以達到語音合成的目的，corpus-based TTS synthesis 首先建立由一個人錄製的連續語料，使得該語言中，因為上下文不同造成的各種聲學特徵及抑揚頓挫(prosody)都能被記錄，在做語音合成的時候便可選出適當的段落(可以是non-uniform unit)來做連結。【5】

   Corpus-based TTS synthesis 的優點在於可以由 non-uniform unit 或較大較完整的聲音來合成，使合成出來的聲音語調較為自然，也比較不會有聲音斷斷續續的問題。【5】

2. (a) - Define a forward variable:

   $$\alpha_t(i) = P(o_1 o_2 ... o_t, q_t = i \mid \lambda)$$

   - Initialization:

   $$\alpha_t(i) = \pi_i b_i(o_1) \qquad 1 \le i \le N$$

   - Induction:

   $$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] \bullet b_j(o_{t+1}) \qquad \begin{matrix} 1 \le t \le T\text{-}1 \\ 1 \le j \le N \end{matrix}$$

   - Termination:

   $$p(\bar{O} \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$ 【5】

   How it works:
   因為$\alpha_t(i)$為給定一個 HMM model λ，看到$o_1 o_2 ... o_t$，而在時間t時 state 為i包括所有過去可能路徑的機率，所以當$t = T$時，表示已經看到了整個$\bar{O}$，此時只要把所有不同i的$\alpha_T(i)$加起來就可以得到全部可能路徑的機率，也就是$p(\bar{O} \mid \lambda)$。【5】

   (b) Viterbi Algorithm:

   - Define a new variable

$$\delta_t(i) = \max_{q_1 q_2 \ldots q_{t-1}} P[q_1 q_2 \ldots q_{t-1}, q_t = i, o_1 o_2 \ldots o_t \mid \lambda]$$

- Induction

$$\delta_{t+1}(j) = \max_i [\delta_t(i) a_{ij}] \bullet b_j(o_{t+1})$$

- Backtracking

$$\psi_t(j) = \arg\max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}]$$

Complete Procedure:

- Initialization

$$\delta_1(i) = \pi_i b_i(o_1), \qquad 1 \le \mathrm{i} \le \mathrm{N}$$

- Recursion

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] \bullet b_j(o_t) \qquad \begin{matrix} 2 \le \mathrm{t} \le \mathrm{T} \\ 1 \le \mathrm{j} \le \mathrm{N} \end{matrix}$$

$$\psi_t(j) = \arg\max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] \qquad \begin{matrix} 2 \le \mathrm{t} \le \mathrm{T} \\ 1 \le \mathrm{j} \le \mathrm{N} \end{matrix}$$

- Termination

$$P^* = \max_{1 \le i \le N} [\delta_T(i)]$$

$$q_T^* = \arg\max_{1 \le i \le N} [\delta_T(i)]$$

- Path backtracking

$$q_T^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \ldots 2, 1$$【5】

How it works:

$\delta_t(i)$為給定一個 HMM model λ，當時間走到t時，在 state i上所有可能 path 中最高的機率；而$\psi_t(j)$則為記錄了當時間為t，state 為 j，在時間$t-1$中使$\delta_t(j)$機率最高的 state。當算到最後我們可以計算$\delta_T(i)$找出λ中可以得到最大機率的最後一個 state q_T^*，並藉由$\psi_t(j)$做 backtracking，找出得到最大機率的整條 path 的 state sequence。【5】

(c) $\arg\max_k P\left(\overline{O}|\lambda_k\right)$是利用 forward algorithm，算出所有 path 機率的總合，找出擁有最大機率的 model λ_k ； $\arg\max_k P\left(\overline{q}^*,\overline{O}|\lambda_k\right)$則是利用 viterbi algorithm，找出擁有最大機率 path 的 model。【2】

雖然 forward algorithm 算出來的機率比較精確，但是由於兩者找出來的 model 通常是同一個，即擁有機率最大 path 的 model 通常其所有 path 的機率總合也會是最大，所以可以用計算量較小的 viterbi algorithm 來取代 forward algorithm。【3】

3. Procedure:

(1) Initialization: L=1，train a 1-vector VQ codebook

$$\bar{v}=\frac{1}{N}\sum_j \bar{x}_j$$

(2) Splitting: split the L codewords into 2L codewords

方法一:

$$\begin{cases}\bar{v}_k^{(1)}=\bar{v}_k(1+\varepsilon)\\ \bar{v}_k^{(2)}=\bar{v}_k(1-\varepsilon)\end{cases}$$

方法二:

$$\begin{cases}\bar{v}_k^{(1)}=\bar{v}_k\\ \bar{v}_k^{(2)}=\textit{the vector most far apart}\end{cases}$$

(3) K-means algorithm to obtain L-vector codebook

(4) Termination, otherwise go to (2)【5】

Why and how it is better than the K-means algorithm:
k-means algorithm 會因為不同的初始條件收斂到不同的 local optimal solution，所以 k-means 的初始條件很重要，而 LBG 就是利用分階段訓練 VQ codebook 的方法，來改進 k-means algorithm 初始條件方面上的限制，例如 $L=1$ 時就會得到整套 data 中的質心，把它一分為二作為初始條件，因而進一步獲得更好的 codebook。【5】

4. 要將某個 node n 分成兩個 node a、b

- Weighted entropy:

定義 $\bar{H}_n = (-\sum_i p(c_i|n)\log[p(c_i|n)])p(n)$，其中 $p(n)$ 是 node n 在全部 data sample 所佔的比例；$p(c_i|n)$ 是 class i 在 node n 中 data sample 所佔的比例。$\bar{H}_n$ 代表了 node n 中資料分佈的 weighted entropy。【3】

- Entropy reduction for the split for a question q

定義 $\Delta\bar{H}_n(q) = \bar{H}_n - (\bar{H}_a + \bar{H}_b)$，即透過 question q 將 n 分成 a、b 後，entropy 下降的程度。【2】

- Choosing the best question for the split at each node

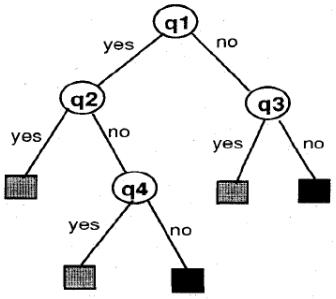
$q^* = \arg\max_q[\Delta\bar{H}_n(q)]$，選出最合適的 q^* 使得 entropy 下降的程度最大。【2】

- Entropy of the Tree T

最後整棵 decision tree T 的 entropy $\bar{H}(T) = \sum_{terminal} \bar{H}_n$，而整個過程會使得 $\bar{H}(T)$ 越來越小。【3】

5. 在 CART 中，只要對某個 node 做 split 後，原本 node 中的成員便不能在 sibling 之間移動，這可能會使原本性質十分相近的 elements 卻因為某個 question 而落到不同的 leaf，這種問題稱作 data fragmentation problem。如果我們用 composite question 來做 split，便能夠有效的將性質相近的 group 分到同一邊，減輕 data fragmentation problem 的影響。【5】

Composite question 的概念在於，當我們想對某個 node x 做 split 前，我們可以先假想 x 為一 root，利用和原本相似的演算法長出一棵 small tree (為了限制 leaf 的數目，一般來說 node 數應介於 4~8 個)。接著我們將所有的 leaf 分成兩群，使得這兩群的 total entropy 為最小。隨機選擇一組叫做 black group，另一組則為 grey group (如下圖)。



然後我們將所有 node 中 question 拿來做 conjunctions、disjunctions 或是 negations 形成 composite questions，目的在於將所有走到 black group 的 path 區別出來。以上圖為例，我們的 composite question 為 $q_1\overline{q}_2\overline{q}_4 + \overline{q}_1\overline{q}_3$。最後再用 composite question 對 node x 做 split【5】

6. $H(S) = -\sum_i p(x_i)\log[p(x_i)]$，相當於某個語言 S 中，平均每個 word x_i 要用多少個 bit 來表示。【3】

如果從 perplexity 的角度來看，我們可以令 $PP(S) = 2^{H(S)}$，就像是我們有一個「虛擬詞典」，在這個詞典中每一個字出現的機率均相等，而這個詞典的大小就是 $PP(S)$。【3】

因此，我們想要去猜下一個字是什麼，就相當於要從 $PP(S)$ 個字中挑一個出來，而 $PP(S)$ 就可以用來作為一個語言 S 的 branching factor estimate。【4】

7. 對於觀察到某一個事件的次數給定一個 threshold r_0，當 observed event 出現的次數超過 r_0 時，我們認為它是可靠的，所以次數不變。【3】

若 observed event 出現的次數小於 r_0 時，則利用 Good-Turing 的概念分配一些次數給 unseen event。【3】

其中 discount 的比例正比於 $\frac{r^*}{r}$（r^* 為 Good-Turing 中的 r^*），並且使 $\sum_{r=1}^{r_0} n_r(1-d_r)r = n_1$，其中 n_r 為出現 r 次的事件的個數，d_r 為出現 r 次的事件的 discount ratio。而 unseen event 是根據 next-lower-order model (back-off)的概念去分配 discount 下來的次數。【4】

8. KL distance: $D[p(x)\|q(x)] = \sum_i p(x_i)\log[\frac{p(x_i)}{q(x_i)}] \geq 0$，是用來計算兩個機率分佈的 relative(cross-) entropy【5】

KL distance 相當於在看當用 $q(x_i)$ 取代 $p(x_i)$ 的分佈描述時，information 的量或是 uncertainty 相差多少，可以用來評估兩個機率分佈的距離，是一種 asymmetric distance ($D[p(x)\|q(x)] \neq D[q(x)\|p(x)]$)。【5】

9. (a) voiced signals: 由振動聲帶所產生的聲音，稱為濁音。在 time domain

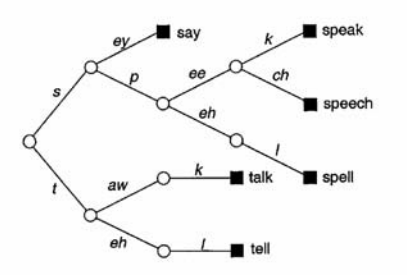
上的波形是會有固定的形狀重複出現。
unvoiced signals: 不經由振動聲帶所產生的聲音，稱為清音。在 time domain 上的波形特徵是沒有固定的形狀重複出現。【5】

(b) pitch 是 voiced signals 重複出現的週期或其頻率，也就是音高。國語的聲調(tone)主要就是音高對時間變化的 pattern。【5】

10. 乘上 hamming window 後會讓 mainlobe 變寬，使得解析度變差，在找 formant frequency 時造成混淆，但因 MFCC 是用 filter bank 取整段 frequency band 的訊號，也就是本來解析度要求就不高，所以影響不大。【5】

但乘上 hamming window 同時也使得 sidelobe 變低，大幅降低 leakage effect（不同頻帶能量的錯置），這方面的效果很好【3】，並增強音框左右端的連續性。【2】故權衡輕重之下，乘上 hamming window 的效果仍舊優於 rectangular window。

11. 在 LVCSR 中，我們可以建立一個 Tree lexicon（如下圖）



【2】

其中每一條線都是一個 HMM (phone)，每一個 leaf node 都是一個字(word)，對於不同但是存在某一部分相同音素(phone)的字群，在搜尋的過程中可以共用一部份的區塊。同樣的樹狀結構都可以在 leaf node 再接下去發展。【3】

演算法如下:

(1) phone HMM 之中:
已為 active 的 node 在自己的 model 中跑 viterbi。

(2) phone 與 phone 之間的 transition:
當發現某個 active node 已跑到最後一個 state 時，參考 tree lexicon 增加新的 active node。

(3) word 與 word 之間的 transition:
當 active node 跑到最後一個 state 且為一個 tree lexicon 的 leaf

時，要產生新的 tree lexicon root node 為 active node，紀錄前一個字為何，並引入 language model 的機率。

根據上述演算法，當給定一個 $\bar{O}$，我們便可以利用 viterbi algorithm 來 search 出機率最大的是哪一個 $\bar{W}$。【5】

12. 以 shortest-path problem 為例，如果我們能使估測距離 $h^*(n)$ 永遠是實際距離 $h(n)$ 的低估的話（$h^*(n) \le h(n)$ for all n），則保證能在第一次就找到最佳解(admissible)。【5】

A^*演算法:

令 $g(n)$ 為從起點走到 n 的距離
$h(n)$ 為從 n 走到終點 G 的估測距離
$f(n) = g(n) + h(n)$

- 建立一 priority queue PQ(based on $f(n)$)，初始為 empty
- V(=set of previously visited ($state, f, backpointer$)-triples)初始為 empty
- 將起點 S 分別放入 PQ 和 V，$f(s) = g(s) + h(s) = h(s)$
- 檢查 PQ 是否為 empty
  - Yes: 很抱歉，沒有解
  - No: 把 $f(n)$ 最小的 node 從 PQ 拿出來，叫做 n
  - 如果 n 是終點 G 的話，回報成功並結束
  - 展開 n: 對於所有 n' 屬於 $successors(n)$
    - 令 $f' = g(n') + h(n') = g(n) + cost(n, n') + h(n')$
    - 如果之前沒看過 n'，或 n' 之前被展開過且 $f(n') > f'$，或 n' 現在在 PQ 中且 $f(n') > f'$，則將 n' 放入 PQ(priority 為 f')；並更新 V，新增 triple($state = n', f', BackPtr = n$)
    - 如果不滿足上式，則忽略 n'

利用反證法證明 A^*演算法為 admissible:

假設 A^*演算法找到的是 suboptimal path，則
$f(G) > f^*$（$f^* = h^*(start)$ = 最短路徑）

則我們一定可以找到一個 node n 滿足:
(1) n 沒有在 A^*演算法中被展開過

(2) 從起點走到n的 path 必定是 optimal path 的起始

可得:

$f(n) \geq f(G)$(不然的話search不會結束)

$f(n) = g(n) + h(n)$

$= g^*(n) + h(n)$(因為在optimal path上)

$\leq g^*(n) + h^*(n)$ (根據admissibility assumption)

$= f^*$

$f^* \geq f(n) \geq f(G) \rightarrow\leftarrow$【10】

13. (a) MLLR 將 Gaussian 分成不同的 class，每次調整都是將整個 class 一起調，使得沒有見過的 tri-phone 也能夠一起被調整。所以即使只有少量的 adaptation data，也可以達到一定的效果。但是相較於 MAP，MLLR 調整的方式就顯得比較粗糙，所以當擁有大量的 adaptation data 時，MLLR 便無法達到跟 MAP 一樣的效果了。【8】

(b) 因為分群的方式對於 MLLR 的 performance 有著很大的影響，因此我們可以利用 tree structure 的觀念，將特性相近而擁有足夠資訊的 node 組合成一個 class。當有新的 adaptation data 進來，我們便可以動態去調整分群的方式而達到比較好的 performance。【7】

14. c_{ij}: word w_i 出現在 document d_j 的次數

$t_i = \sum_{j=1}^{N} c_{ij}$: word w_i 在所有 document 中出現過的總次數【2】

N: document 的總數

$\frac{c_{ij}}{t_i}$: word w_i 出現在 document d_j 的機率

$-\sum_{j=1}^{N} (\frac{c_{ij}}{t_i}) \log(\frac{c_{ij}}{t_i})$: word w_i 分佈在所有 document 中的 entropy，值越小代表 word w_i 對於某些 document 具有獨特性;越大則代表 word w_i 均勻分佈在所有的 document 中(e.g.「的」)

$\varepsilon_i = -\frac{1}{\log N} \sum_{j=1}^{N} (\frac{c_{ij}}{t_i}) \log(\frac{c_{ij}}{t_i})$: 經過 normalize 的 entropy，使其介於 0~1【4】

n_j：document d_j 中的字數

$w_{ij} = (1-\varepsilon_i)\frac{c_{ij}}{n_j}$：word w_i 在 document d_j 中的頻率，並用 document d_j 中的總字數以及 word w_i 分佈在所有 document 中的 entropy 做 normalize。也可以說是 word w_i 對於 document d_j 的重要程度。【4】