Discrete Hidden Markov Model Implementation

Environment: OS: Ubuntu 12.04.2 64bit (Linux 3.2.0-38-generic)
Compiler: g++ (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3

執行方式

- 編譯程式: $ make
- Train the models: $ ./train 155 model_init.txt seq_model_**N**.txt model_**N**.txt (for N = 01~05)
- 測試 models: $ ./test modellist.txt testing_data**N**.txt result**N**.txt (for N = 1,2)

額外指令:

- 輸出所有 iterations 的 models:
  $ ./train 155 model_init.txt seq_model_N.txt model_N.txt **-a** (輸出檔名： model_N.txt.1~155)
- 使用不同的演算法來進行測試:
  $ ./test modellist.txt testing_dataN.txt resultN.txt **METHOD** (where METHOD = naive, viterbi, forward)
- 利用解答檔案來計算 accuracy (METHOD 也必須被指定):
  $ ./test modellist.txt testing_dataN.txt resultN.txt METHOD **testing_answer.txt**
- 計算 1~N iterations 的 accuracy:
  $ ./test modellist.txt testing_dataN.txt resultN.txt METHOD testing_answer.txt **-i N**
- 以 forward algorithm 作為預設演算法來測試 models:
  $ ./test_**forward** modellist.txt testing_dataN.txt resultN.txt (for N = 1,2)

(當使用 forward algorithm 時，model 必須使用 214th iteration 才能達到最高 accuracy)

結果

在這次作業中，三種方法被使用來估計 $P(O|\lambda)$:

1. 使用 Viterbi 來估計 $P(P^*|\lambda) \cong P(O|\lambda)$
2. 使用 Forward algorithm 來估計 $P(O|\lambda)$
3. 使用投影片 p.19, chapter 4 提到的簡單方法來估計 $P(P^*|\lambda) \cong P(O|\lambda)$

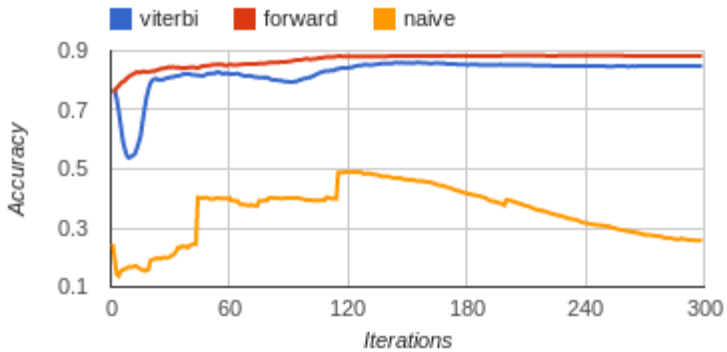

Figure 1. comparison between different methods

從圖中可看出，Forward 是最精確的，而簡單方法幾乎沒有用處。事實上，這個簡單方法甚至是最花時間的演算法。（其實在它算出 $P(P^*|\lambda)$ 之前，早就已經先算過 $P(O|\lambda)$ 了，所以使用這個方法其實沒有意義。）

利用 testing_data1.txt 產生測試結果後，我們將其和 testing_answer.txt 比較來取得 accuracy。同樣的程序分別針對三個方法、**1~400** iterations，分別執行。但由於 naive 以外的數值很快穩定下來，所以在圖表中我們只秀出一部分的結果。

Viterbi

使用 Viterbi algorithm 的時候，我們可以看到 accuracy 曲線跟 FAQ 頁面上的非常類似，accuracy 一開始會降低，但增加 iterations 以後就會增高。我認為之所以如此是因為，當 model 從初始值慢慢轉向新的狀態時，機率一開始被分配的還不夠平均，有些不穩定的突起，而由於 Viterbi 使用最佳的路線來估計，很容易就會受極端值所影響。

以 testing_data1.txt 測試時觀察到最高的 accuracy 是 **0.8608**, 他在 **155th** iteration 被觀察到。而且 accuracy 在大約 230 iterations 以後就差不多穩定在 0.85 上下.

Forward

使用 Forward algorithm 並以 testing_data1.txt 測試時，accuracy 差不多都是一直增加的，然後在 155 iterations 以後會穩定在 0.8835 左右。最高的 accuracy 是在 **214th** iteration 出現的 **0.8852**。

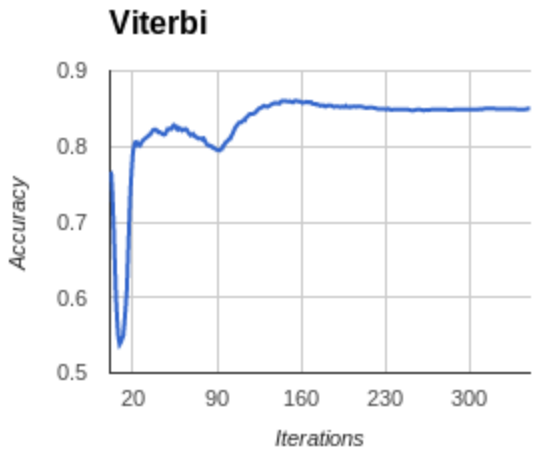

Figure 2. Viterbi

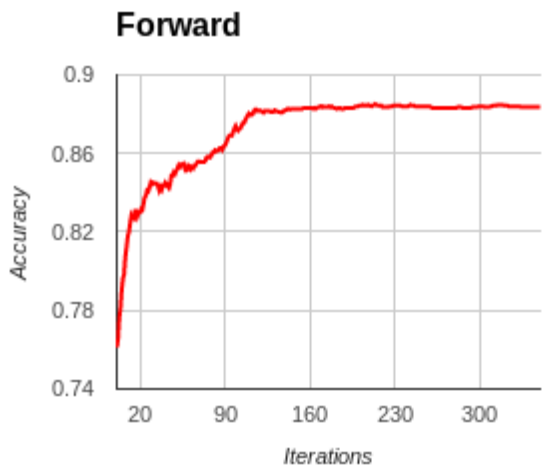

Figure 3. Forward

時間

或許最有趣的結果是時間的比較，從圖中可以看出 Forward 竟然是最快的。以演算法複雜度來分析，Forward 和 Viterbi 其實同樣都是 $O(T \times N^2)$ ，然而由於實做上的細節，使得 Viterbi 的乘法次數較多，故時間也拉長。

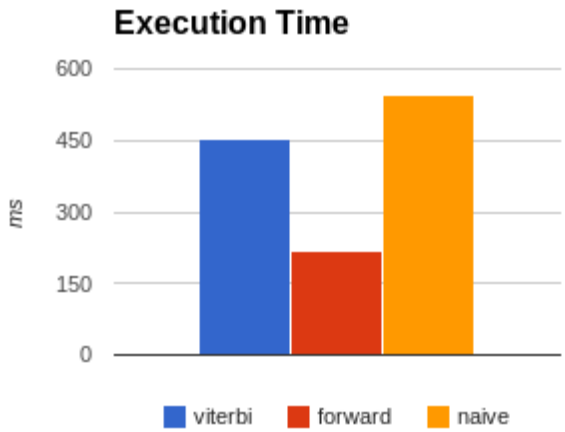

Figure 4. Execution Time

老師在上課時有提到，真正做長句的語音分析時，因為某些特殊的原因，會使得 Viterbi 比較好算，這個作業因為是單個字的簡單模型，所以還看不出來這種效應。