

第三章 改進方法

[2]、[3]中對於整個 MFCC feature 中已提出了一些較具體的做法，其中[2]是針對數字 0~9 的單音節部分及 70 人左右約七百句，長度約兩秒的語料來進行實驗，而[3]所採用的語料則是由長庚大學所提供的唐詩語料，資料如表格 3-1，來對 MFCC 的所有調整參數進行調整。

語料資訊

Speaker	38 males and 12 females
Sampling rate	16 KHz
Bits per sample	8 bits
Total	1000 files

表格 3-1 唐詩語料資訊

正因為經過 HTK 訓練出來的 HMM Macro File 會隨著不同語料而產生不同的機率分佈，所以在這邊我們將會對 TCC300 及 TIMIT 所有的語料進行分析，然後來由實驗的結果來產生適用於兩個語料個別的對應參數，而 TCC300 及 TIMIT 相關資訊見表格 3-2。

	TCC300 Corpus	TIMIT Corpus
Speaker	150 males and 150 females	483 males and 192 females
Sampling rate	16 kHz	16 kHz
Bits per sample	16 bits	16 bits
Total files	8913	6300
Total Time	26.34 hours	5.38 hours

表格 3-2 TCC300 及 TIMIT 的語料資訊

而整個 MFCC 的流程請參照圖 2-2，接下來的章節將會一一提出 MFCC 各個流程所選取的最佳參數。

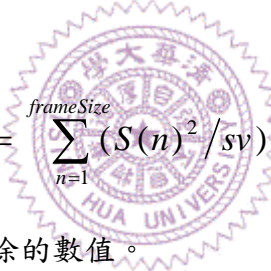
3.1 計算能量 (Energy)

首先，我們先對每一個音框計算其能量的值，公式如下：

$$E = \sum_{n=1}^{frameSize} S(n)^2 \dots (3.1-1)$$

其中 $S(n)$ 代表音框內第 n 點之值， E 表能量。

但在這邊我們馬上會遇到一個棘手的問題，即是“資料溢位”的問題，因為我們在這邊所有的資料型別均為整數的資料型別，而所有的 $S(n)$ 的資料大小都是由 16 bits 轉型至 32 bits，所以如果只有直接對 $S(n)^2$ 的值加總造成資料溢位的機率相當高，因此我們在這邊勢必要作一些改變來避免溢位，我們將 eq. (3.1-1) 修改成 eq. (3.1-2)。


$$E = \sum_{n=1}^{frameSize} (S(n)^2 / sv) \dots (3.1-2)$$

其中 sv 是為了要避免溢位所除的數值。

由於 E 將會在往後的步驟中計算其 \ln 的值，因此為了方便運算，所以在這邊我們取 sv 為 e 的指數次方的量值，根據 TCC300 與 TIMIT 的語料測試我們得到資訊如下：

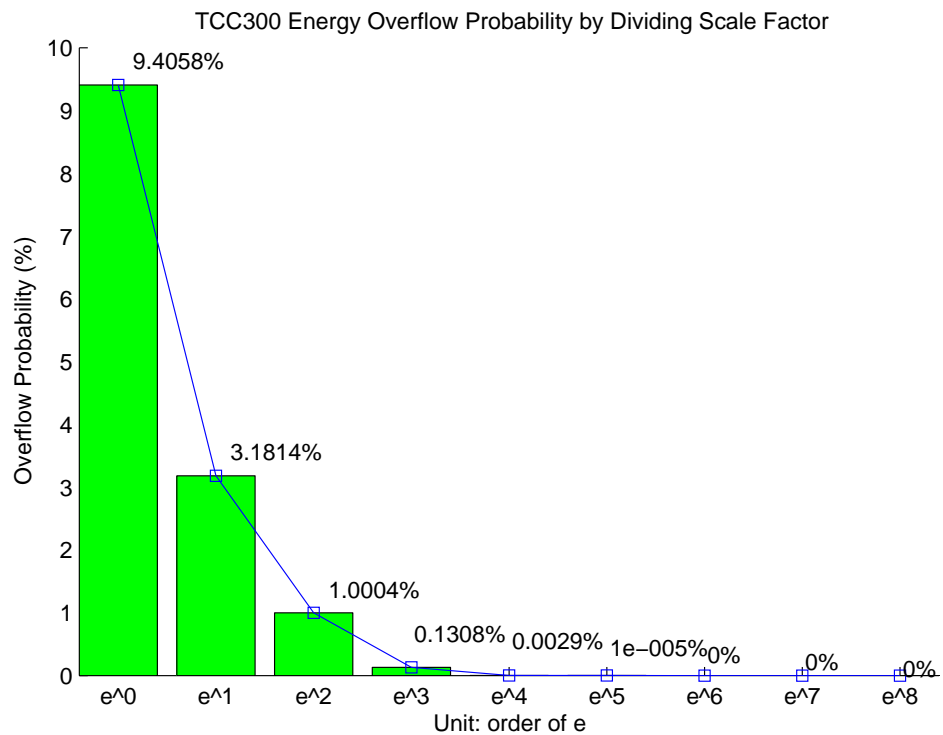


圖 3-1 TCC300 語料於計算能量與調整參數值資料溢位的機率

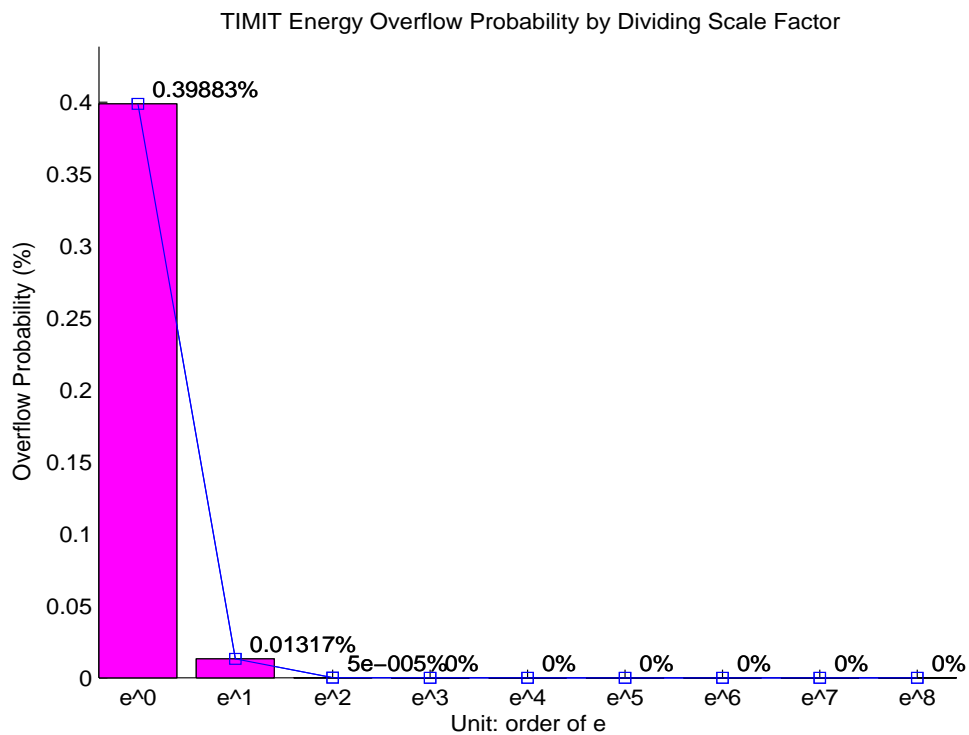


圖 3-2 TIMIT 語料於計算能量與調整參數值資料溢位的機率

圖 3-1、圖 3-2 可以得知 TCC300 與 TIMIT 的能量在除以 e^3 及 e^6 以後就不會再有資料溢位的狀況產生，所以我們依據公式 3.1-2，可將 sv 取 e^3 及 e^6 ，不過由於 TIMIT 語料的錄製聲量較小，並考量如果受測語料的錄製聲量較大，所以我們在這邊 sv 值取為 e^6 ，這樣會比較符合一般的狀況。

3.2 預強調 (Pre-Emphasis)

預強調的公式為 $S_2(n) = S(n) - a \times S(n-1)$ ，其中 $a = 0.975$ ，所以我們要放大 a 的數值，在這邊我們將 a 的值放大 2^{14} 倍，並得到 $a = 0.975 \times 2^{14} = 15974.4 \cong 15974$ ，由於 $S(n)$ 為 16 bits 擴增到 32 bits 之資料，所以並不用擔心資料溢位的問題，而整個式子重新改寫成為：

$$S_2(n) = (S(n) \times 2^{14} - 15974 \times S(n-1)) / 2^{14} \dots (3.2-1), \text{ 其中 } 2 \leq n \leq \text{frameSize}$$

$$S_2(n) = 410 \times S(n) / 2^{14} \dots (3.2-2), \text{ 其中 } n=1, 0.025 \times 2^{14} = 409.6 \cong 410$$

eq. (3.2-2) 是根據[9]而來。

3.3 漢明窗 (Hamming Window)

乘上漢明窗後的訊號為： $S'(n) = S(n) \times W(n)$ ，其中 $W(n)$ 算法可參照公式 (2.2-1)，但由於 $W(n)$ 的值為浮點數，所以我們必須建立一個表，而裡面的數值為 $W(n) \times hmSv$ ，在這邊我們取 $hmSv$ 為 2^{14} 來建立整數型態的漢明窗，我們可由圖 3-3，得知我們取得不同 $hmSv$ 值所建立出的漢明窗值的分佈，並由圖 3-4 可判斷出漢明窗鑑別度與離散度的差異。最後我們將原來的 $S'(n) = S(n) \times W(n)$ 改寫成為：

$$S'(n) = S(n) \times \text{HamTable}(n) / 2^{14} \dots (3.3-1)$$

因為漢明窗整數表已經被放大 2^{14} 倍，所以在與訊號 $S(n)$ 相成後還要再除以

2^{14} 來還原其值。

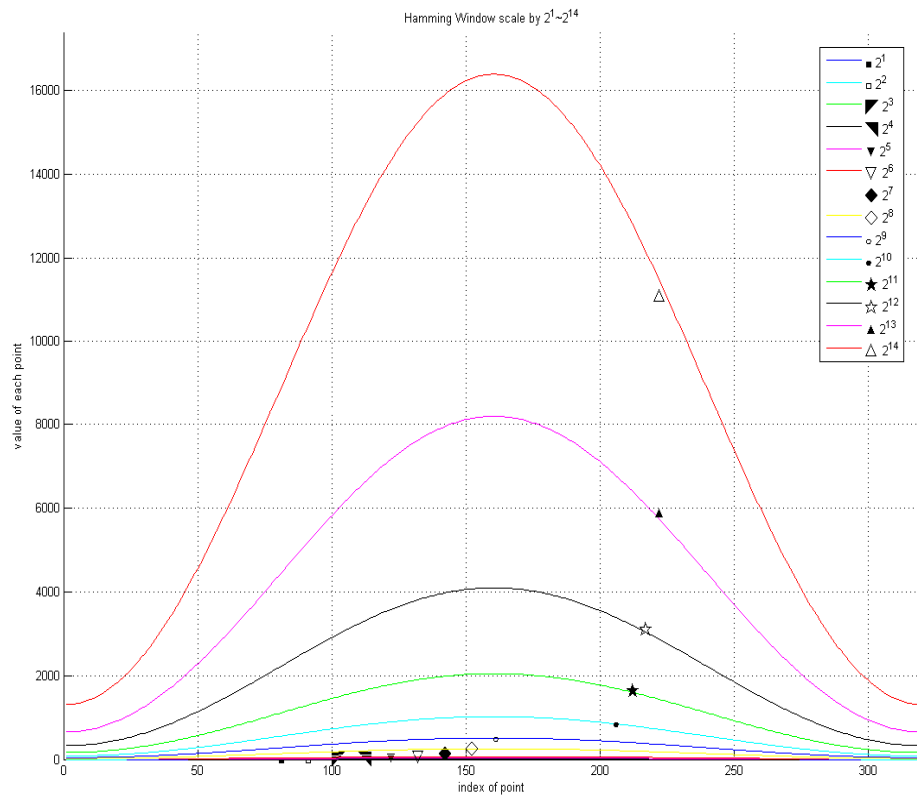


圖 3-3 不同倍率放大之整數漢明窗值的分佈圖

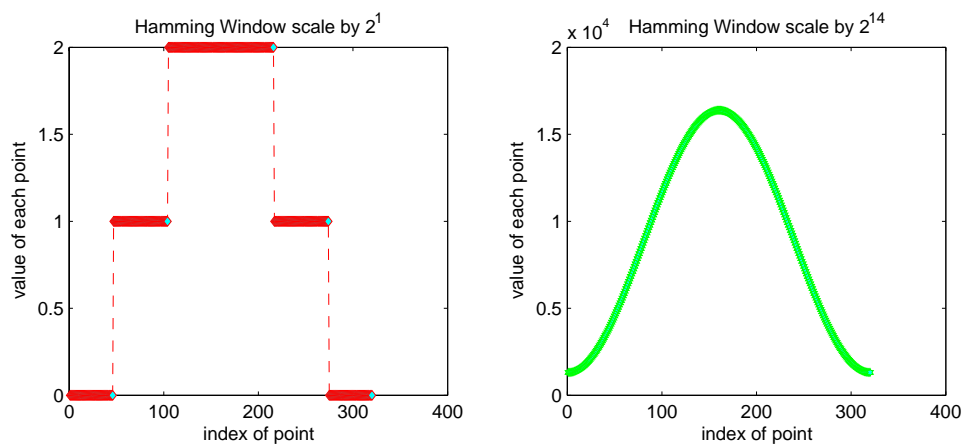


圖 3-4 放大倍率 2^1 與 2^{14} 的圖形

3.4 快速傅立葉轉換 (FFT)

在 FFT 的步驟中，我們可以得知 FFT 的公式如下：

$$X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn}, \quad 0 \leq k \leq N-1, \quad \text{where } W_N = e^{-jk2\pi/N}$$

其中， $e^{-jk} = \cos(k) - j\sin(k)$ ，在這邊我們必須建立兩個 Table 給 FFT 使用，分別為 Cos Table 與 Sin Table，由於要經由 Cos 及 Sin 運算的值，是可以預先運算而得到的，我們就可以根據這些值，去建立相對應的表來運算之，而不需要去建立一個完整的 Cos 及 Sin 對照表。圖 3-5 為 DCT 演算法，

```

pi_factor = PI / numChans;
FOR cepNumCounter = 1 TO numCepCoef DO
    cepstral[cepNumCounter] = 0.0f;
    ftemp = cepNumCounter * pi_factor;
    FOR chanNum = 1 TO numChans DO
        cepstral[cepNumCounter] += fbank[chanNum] * cos(ftemp * (chanNum - 0.5));
    END DO
    cepstral[cepNumCounter] *= mfnorm;
END DO

```

圖 3-5 DCT Algorithm in MFCC Procedure

我們可得需要經過 Cos 運算的值為：

$$\left\{ \begin{array}{l} value = ftemp \times (chanNum - 0.5), \quad 1 \leq chanNum \leq 26; \\ ftemp = cepNumCounter \times pi_factor, \quad 1 \leq cepNumCounter \leq 12; \\ pi_factor = PI \times numChans, \\ numChans = 26, \quad PI = 3.14159265358979 \end{array} \right\} \dots (3.4-1)$$

所有的值共有 312 個(12×26)，所以我們將這 312 個值，代入 Cos(Value) 運算，然後將所有的值乘以 2^{14} (16384)，再建立一個一維陣列表來存放這些值。

Sin 部分的運算，總共在 FFT 及 FFT 的次函數與 Weight Cepstrum 函數出現，

[9]其中 Weight Cepstrum 公式如下：

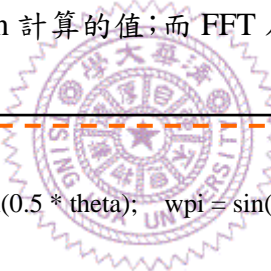
$$C'_n = \left(1 + \frac{2}{L} \sin \frac{\pi n}{L}\right) \times C_n, \quad L = 22, \quad 0 < n < 12 \dots (3.4 - 2)$$

Pseudo Code 如圖 3-6：

```
ftemp = PI / cepLifter;
lby2 = cepLifter / 2.0f;
FOR cepNumCounter = 1 TO numCepCoef DO
    cepWin[cepNumCounter] = 1.0f + lby2 * sin(cepNumCounter * ftemp);
END DO
```

圖 3-6 Weight Cepstrum Algorithm in MFCC Procedure

這邊有 12 個需要運用 Sin 計算的值；而 FFT 及 FFT 次函數的部分如圖 3-7：



```
limit = 2;
REPEAT
    theta = TPI / limit;  x = sin(0.5 * theta);  wpi = sin(theta);
    Other Procedure...
    FOR ... DO
        Other Procedure...
        FOR ... DO
            Other Procedure...
        END DO
        Other Procedure...
    END DO
    limit Shift Left 1;
UNTIL limit < n

n = size Shift Right 1;
theta = PI / n;
CALL SUBROUTINE OF FFT
x = sin(0.5 * theta);
yi2 = sin(theta);
Other Procedure...
FOR...DO
    Other Procedure...
END DO
Other Procedure...
```

sub-routine of fft

function of fft

圖 3-7 Sub-routine of FFT and FFT Function Parts of Algorithm in MFCC Procedure

在這邊 FFT 的副程式會運算 16 個不同的值，如表格 3-3，而 FFT 函數中僅僅只有 2 個不同的值，如表格 3-4 所示，加上先前的 12 個值，我們總共會有 30 個值來建立我們的 Sin Table，將這些值經由 Sin 運算出來，再放大 2^{14} 倍存入我們所建立的 Sin Table，而 Sin Table 的結構如圖 3-8。

	Round 1	Round 2	Round 3	Round 4
Index 1	$\pi/2$	$\pi/4$	$\pi/8$	$\pi/16$
Index 2	π	$\pi/2$	$\pi/4$	$\pi/8$
	Round 5	Round 6	Round 7	Round 8
Index 1	$\pi/32$	$\pi/64$	$\pi/128$	$\pi/256$
Index 2	$\pi/16$	$\pi/32$	$\pi/64$	$\pi/128$

表格 3-3 Input Value for Calculating Sin in FFT Sub-routine

	Index 1	Index 2
Round 1	$\pi/512$	$\pi/256$

表格 3-4 Input Value for Calculating Sin in FFT Function

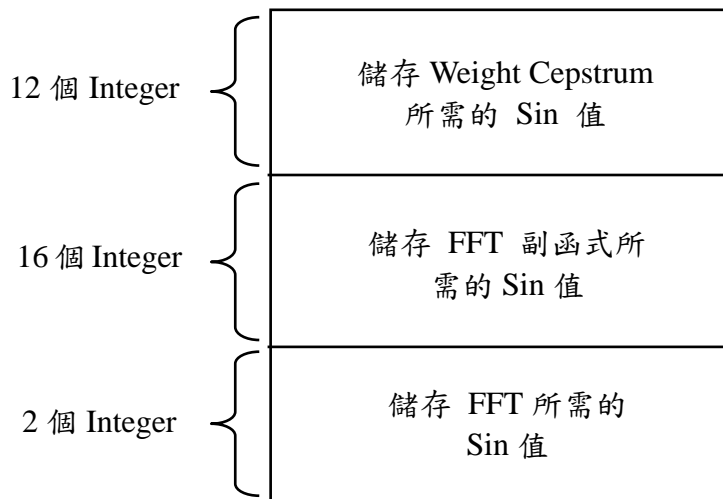


圖 3-8 Sin Table 結構圖

3.5 三角帶通濾波器

在尚未計算 \ln 之前的訊號，經過三角帶通濾波器的方程式如下：

$$S[m] = \sum_{k=0}^{N-1} |X_a[k]|^2 H_m[k], \quad 0 < m \leq M \dots (3.5-1), \text{ 其中}$$

$$H_m[k] = \begin{cases} 0 & , \quad k \leq f[m-1] \\ \frac{(k - f[m-1])}{(f[m] - f[m-1])} & , \quad f[m-1] < k \leq f[m] \\ \frac{(f[m+1] - k)}{(f[m+1] - f[m])} & , \quad f[m] \leq k < f[m+1] \\ 0 & , \quad k > f[m+1] \end{cases}$$

... (3.5-2)

與此我們必須要建立一個 Filter Table 來儲存 $H_m[k]$ 的值，由方程式(3.5-1)我們可以把 $|X_a[k]|^2$ 拆解為 $\sqrt{X_r[k]^2 + X_i[k]^2}$ ，其中 $X_r[k] = X_r[k] + j \cdot X_i[k]$ ，為了避免 $X_r[k]^2$ 及 $X_i[k]^2$ 會造成資料溢位的情形，所以我們將 $X_r[k]$ 及 $X_i[k]$ 各別除以 16：

$$X_r[k] \gg 4$$

$$X_i[k] \gg 4$$

然後我們可以推得 $|X_a[k]|^2 = \sqrt{X_r[k]^2 + X_i[k]^2} \ll 4$ ，而 Filter Table 我們則是建立了一個大小為 256 (針對一個頻道)，並且將值放大 2^{11} 的 Filter Table，圖 3-9、圖 3-10 顯示在 TIMIT 及 TCC300 Corpus 中 $|X_a[k]|^2$ 根據不同放大倍率的 Filter Table 發生資料溢位的機率(只要頻譜乘以三角濾波器的一點發生溢位)。

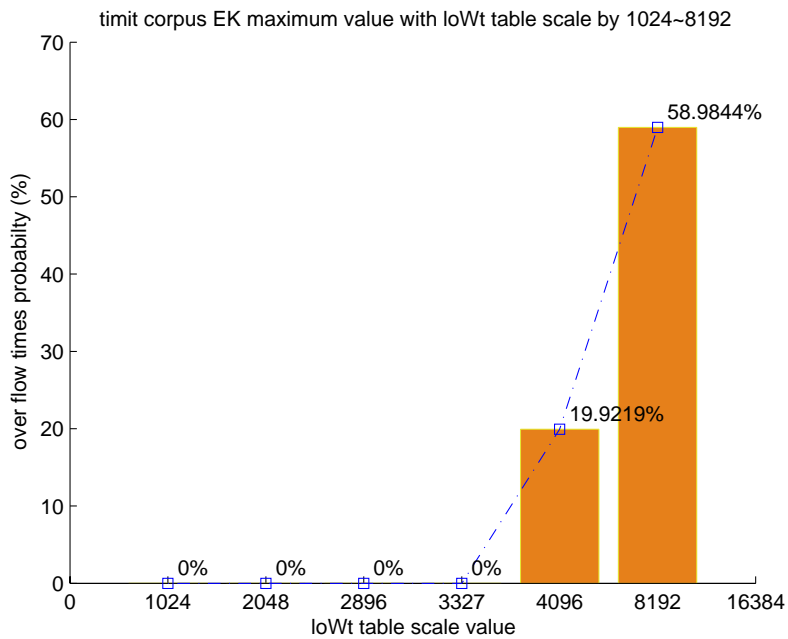


圖 3-9 TIMIT 語料中 $|X_a[k]|^2$ 與不同倍率 Filter Table 產生溢位之機率圖

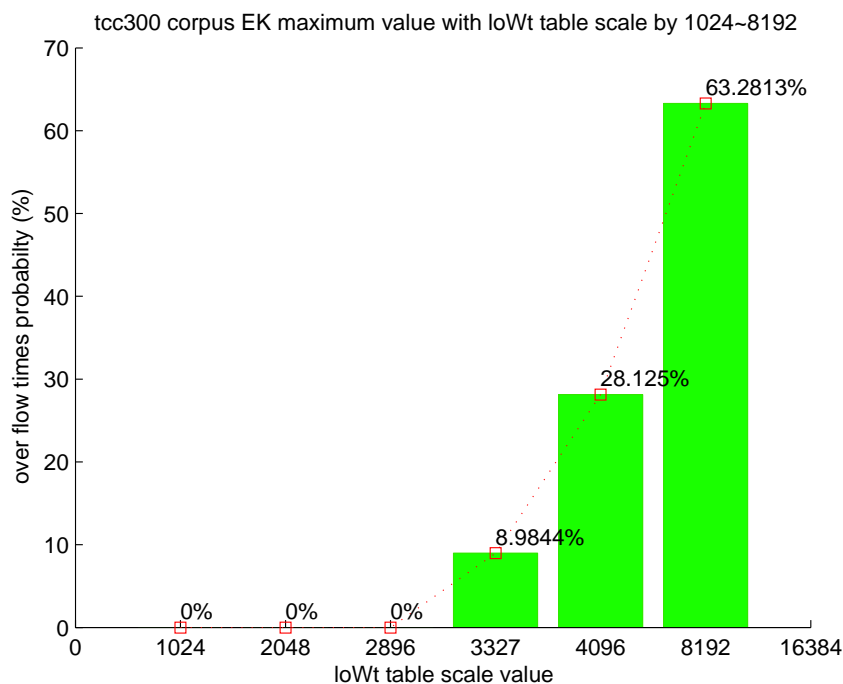


圖 3-10 TCC300 語料中 $|X_a[k]|^2$ 與不同倍率 Filter Table 產生溢位之機率圖

3.6 離散餘弦轉換

經過三角帶通濾通器後，必須還要再經過離散餘弦轉換的運算，公式如下：

$$c_i = c \sum_{j=1}^N m_j \cos\left(\frac{\pi \cdot i}{N} \times (j - 0.5)\right) \dots (3.6-1)$$

其中 $i=12$ (MFCC Coefficient 的數目)， $N=26$ (三角帶通濾波器個數)，而 $c = \sqrt{\frac{2}{N}} = \sqrt{\frac{2}{26}} = 0.27735$ ，請參照[9]，在這邊我們將 c 放大 1846 倍， c 約等於 512，避免資料溢位，然後直接運用我們在 3.4 節中所建立的 Cos 表來查詢相對應的 Cos 值，因此公式 3.6-1 可以更改成：

$$c_i = \left(c \sum_{j=1}^N m_j \cdot \cosTable[i] \right) / 1846, \quad c = 512 \dots (3.6-2)$$

因為 Cos 的值是對陣列的註標直接存取，所以可以保證效能為 $\Theta(1)$ 。

3.7 計算能量對數

由 3.1 節中我們將計算能量的公式改寫為 $E = \sum_{n=1}^{frameSize} (S(n)^2 / sv)$ ，其中

$sv = e^6 = 403$ ，因此在對能量計算 \ln 的時候必須再乘上 sv ，所以我們可以得到

$c_{13} = \ln(E \times sv) = \ln(E \times e^6) = \ln(E) + 6$ ；而計算 \ln 處，我們必須要建立整數的表格

來代替直接使用內建函數，但是 \ln 並不是一個線性的對應，如圖 3-11 所示，y 軸的遞增趨勢，會因為 x 軸值越大而變緩的相當劇烈，[2][3]所提出的方法是依照不同的區段去建立表格，以達到節省空間與降低誤差產生的目的，經實作的結果，必須產生 283211 大小的表來儲存，大約佔了約 1.08MB 的記憶體空間，如果我們去檢視表內的值，會發現有相當程度的重複，因此我們在這邊還有可以改善的空間，遂於此提出另一種建表的方式，冀能降低表的大小與降低浮點數值轉換成整數時所產生的誤差。

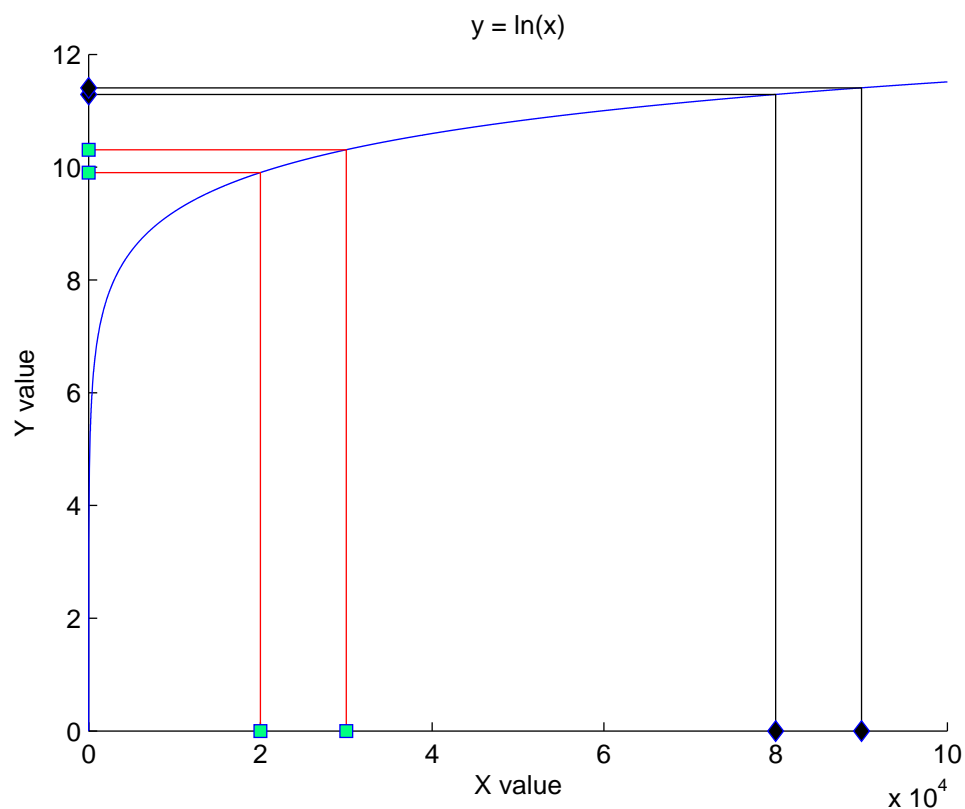


圖 3-11 $\ln(x)$ X 軸與 Y 軸的相對應關係

有別於之前的以 X 軸等分的值來對應到 Y 軸的建表方法，我們在這裡改由從 Y 軸反過來對應 X 軸，也就是等分 $\ln(x)$ 的值，然後以適當的比例放大，再將 Y 軸的值反對應到 X 軸（即對 Y 軸的值計算 \exp ），如圖 3-12 所示，就會產生 X 軸的某段區間，如 X 軸範圍 403~1096 ($e^6 \sim e^7$) 會落在 Y 軸 6~7 之間，但是我們的運算器是沒有浮點數運算，所以我們必須要對 Y 軸的值放大並取整數，例如：TCC300 在這邊必須還要乘上 896[3] 來放大之。而 X 軸的對應也不再是之前的一一對應，所以在這邊我們必須要再利用 Binary Search 來找出落在某範圍的 x 值所對應到的 y 值，因此我們在計算 $\ln(x)$ 的時間，相對會較之前提升一點，從 $\Theta(1)$ 提升到 $\Theta(\log_2 n)$ ，雖然所需時間提升了一些，不過很明顯的效應是表的大小降低很多，以此例我們建出的表大小約是 15526 元素，約佔記憶體空間 0.06MB，與之前的 1.08MB，約省下了 1.02MB 的空間，降低了約 90%，如表格 3-5 所示。

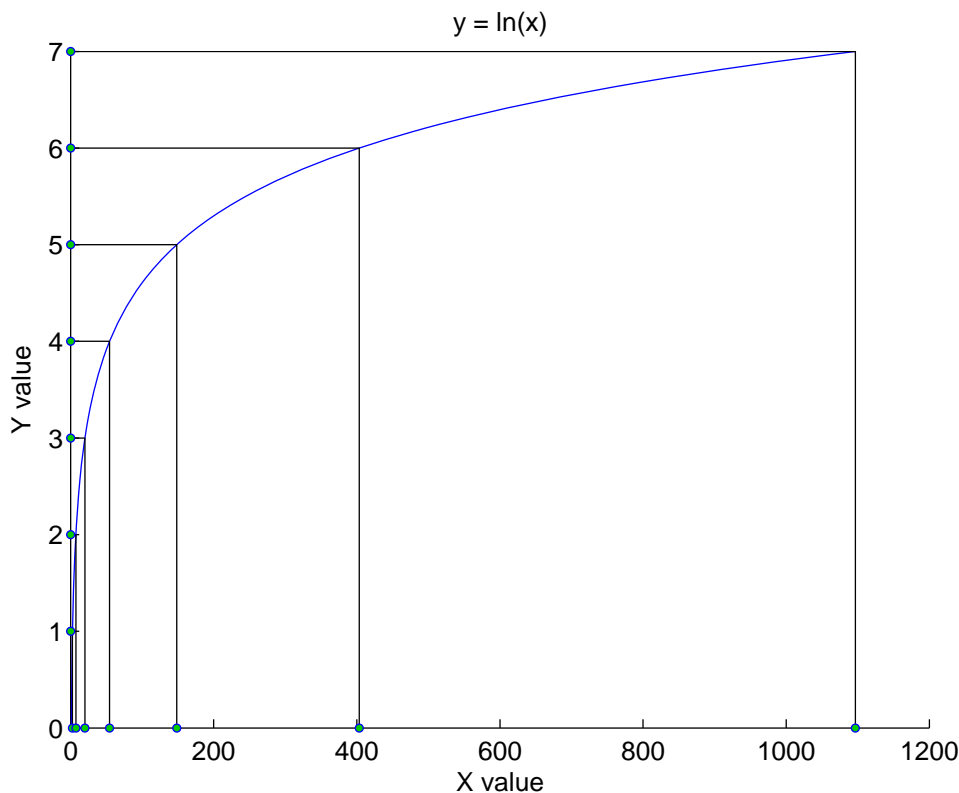


圖 3-12 整數 $\ln(x)$ Table X 軸與 Y 軸的相對應關係

	查詢時間	表的元素個數(總合)	表的個數	表的大小
[2]、[3]的建表方式	$\Theta(1)$	283211	3	1.08MB
我們的建表方式	$\Theta(\log_2 n)$	15526	1	0.06MB

表格 3-5 [2][3]與我們的建表方式比較

3.8 開平方根表格建立

平方根的 X 軸與 Y 軸對應的趨勢圖基本上與計算 \ln 的圖形有相當程度的類似，只是開平方根的坡度相較之下比較陡許多，因此我們也可以運用相同的方式來建立整數型態的開平方根表 (Square Root Table)，而查表的搜尋方式也是運用 Binary Search 來行運算之。圖 3-13 顯示整數型態的開平方根 X 軸與 Y 軸相對應的關係。

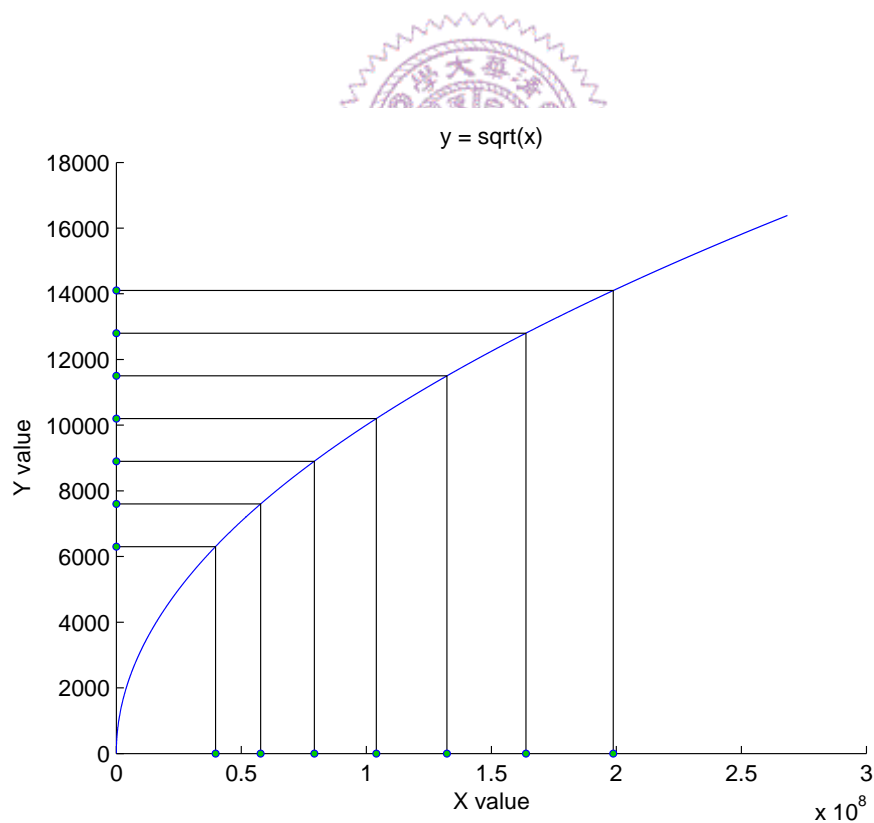


圖 3-13 整數 $Sqrt(x)$ Table X 軸與 Y 軸的相對應關係