学　　号：14507127

天津商业大学宝德学院毕业设计（论文）

# 基于频谱分析的 MFCC 研究与测试

**MFCC Research and Testing**

**Based on Spectrum Analysis**


系:　　计算机与信息技术系

专业:　　计算机科学与技术

班级:　　1401

学生姓名:　　陈　静

指导教师:　　丁玄功　　副教授


**2018 年 5 月**

# 目　　录

# 内容摘要

近年来，无论是无人驾驶汽车，还是 Alpha Go 的出现，人工智能发展的趋势已是势不可挡。所谓的人工智能——顾名思义就是让机器人拥有智能。而语音识别技术，就是机器智能体现的一方面。MFCC 算法，是一种根据人耳识别声音的方法而开发出来的声音识别算法。近年来，语音识别技术不断提高。未来在医学、工业、教育甚至是进入家家户户都是有必要的，语音识别技术也被列入为计算机发展十大技术之一，可谓是很重要。而其中很重要的一方面就是语音信号的特征提取。说到特征提取，对于信号来说那是相当重要的步骤，无论信号是要用来做什么？怎么做？前提就是特征提取，提取了信号的特征才能够继续进行下一步。 所以语音信号的特征提取是十分重要且必要的。而 MFCCs（Mel Frequency Cepstral Coefficents）是识别时广泛使用的特征。本实验采用的软件是 MATLAB，准确来说是一款十分强大的商业数学软件，对于语音识别技术的特征提取以及做傅里叶变换、反傅里叶变换都起着技术上的简化作用。

**关键词**：语音识别技术；傅里叶变换；MATLAB；MFCC

# Abstract

In recent years, whether it is driverless cars or the emergence of alpha go, the trend of artificial intelligence development has been unstoppable. The so-called artificial intelligence, as the name implies, is to make robots have intelligence. And voice recognition technology is one aspect of machine intelligence. MFCC algorithm is a kind of sound recognition algorithm based on the method of human ear recognition. In recent years, speech recognition technology has been improving. In the future in medicine, industry, education and even into every house is necessary, speech recognition technology has also been listed as one of the top ten computer development technology, is very important. One of the most important aspects is the feature extraction of speech signals. When it comes to feature extraction, that is a very important step for the signal, no matter what the signal is intended to do? How? The premise is that the feature extraction, extracted features of the signal can continue to the next step. The feature extraction of speech signal is very important and necessary. MFCC (Mel frequency ceps traffic) is widely used in identification. The software used in this experiment is MATLAB, which is a very powerful commercial mathematics software. it plays a simplified role in feature extraction, Fourier Transformation and inverse Fourier transform of speech recognition technology.



**Key Words:** SPEECH RECOGNITION Fourier Transformation MATLAB MFCC

# 1 导言

当今世界，人工智能是人们不可缺少的议题之一，他也是计算机科学的一个分支。人工智能的研究领域包括：语音识别、自然语言处理、专家系统、图像识别和机器人等。其中在语音识别技术中，可以说是在全世界范围都有了突破性的发展。研究也发现，最好的识别方法其实就是仿造人耳的构造，越接近识别程度越高。于是，在多方努力下，MFCC 算法问世了。MFCC 算法，是研究人员为了仿造人耳接收辨别声而开发的声音识别算法，其精妙程度立于当今世界声音识别算法之巅。近年来，语音识别技术不断提高。未来在医学、工业、教育甚至是进入家家户户都是有必要的，语音识别技术也被列入为计算机发展十大技术之一，可谓是很重要，而其中很重要的一方面就是语音信号的特征提取。本研究与测试主要就是针对于如何实现特征提取逐步解析。基本过程是：声音的读取、语音信号预处理、MFCC 特征提取。该研究与测试还涉及了一些 SVM（支持向量机）二分类器的知识。但由于实验时间有限，加上 SVM 是机器学习的一种，需要大量数据，以此来作为一个库，实现机器学习。

## 1.1 人工智能

人工智能（Aritificial Intelligence，AI），如图 1 所示，是一门融合性的前沿综合性学科，通俗的解释也就是：他是一门大杂烩学科，什么都要会，什么方面都要狩猎。其融合了计算机科学、统计学、脑神经学和社会科学。人工智能的目标也很明确，是让计算机能够在无限的算法中做到能自我学习、认知、分析、决策等多种多样的人类所具备的功能，使其无限接近人类，像人类一样能够有自己的智力。但是曾经看过不少人工智能的影片，即使是科学前沿的代表人物——霍金，都不赞成过度开发人工智能，也就是所谓的超人工智能。那么，人工智能与我们而言到底是好是坏，是优大于劣还是劣大于优呢？其实，就目前而言，我们还不到需要考虑这个问题的时候，因为就目前而言，人工智能还不足以能够发展到超人工智能的时代，那或许是几十年几百年亦或者是几千年之后的人类需要考虑的问题。而就当今社会来说，发展人工智能是必不可少的，无论是在军事层面上、国家实力层面上还是人民生活水平层面上。

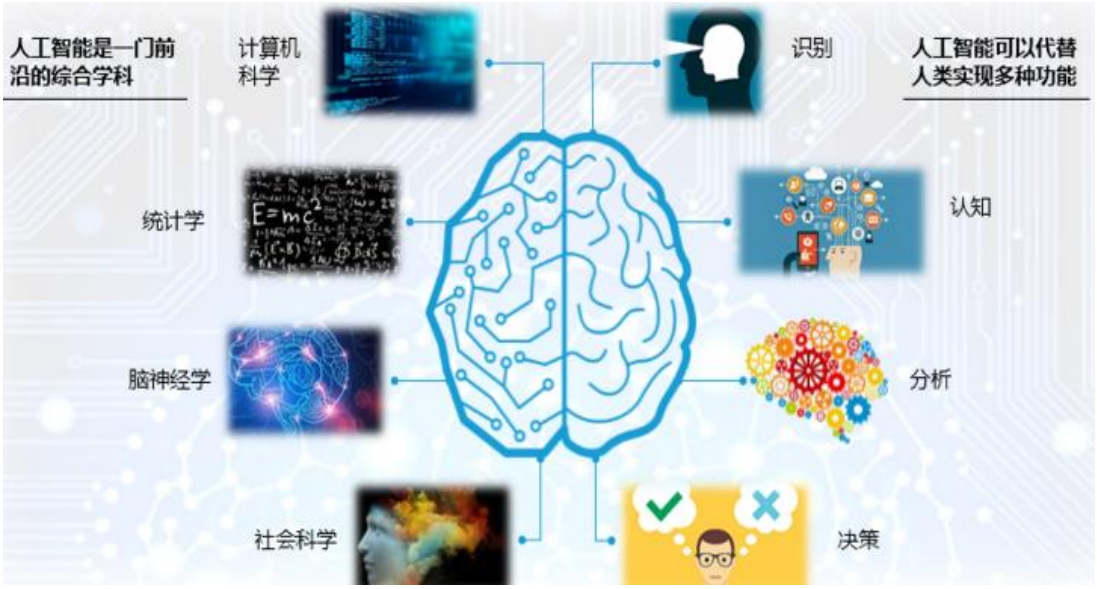


**图 1 人工智能引申图**

## 1.2 人类智能应用领域

人工智能的研究领域其实在网上一查便知。主要有 5 层，从下往上，也就是从基层看来，最下面一层是基础设施建设，类似于建房子总得有砖和水泥，人工智能的砖和水泥就是大数据和计算能力，显而易见数据量越大，人工智能的能力越强。往上一层是算法，这就类似于架构图，是建筑的灵魂。人工智能在算法中采用了譬如深度学习、机器学习等算法。再往上一层是一些主要的技术方向。第二层是各个技术方向中的具体技术。最上层就是人工智能的应用领域。

## 1.3 语音识别技术

语音识别技术又称为自动语音识别（Automatic Speech Recognition，ASR），是一种运用大量算法，将人类语音发出的词汇内容，转变成一种可读输入，让计算机进行读取，例如二进制编码、字符序列或按键。通俗来说就是实现录音、识别音频、辨别音频。

## 1.4 语音识别的发展及前景

古有上天入海之想，于是后来的我们将其实现了，发明了飞机和潜水艇。可以看出我们是多么的憧憬我们不可及之事。那么自动语音识别这么有魅力的事儿，又怎么会落后呢？事实上早在计算机出现之前，自动语音识别这种“异想天开”的想法就已经在人们的脑海中出现并不断盘旋，在早期可以被认为是达到一

些语音识别初阶段的就是声码器。

1920 年代，当时出现了一只名为“Radio Rex”的玩具狗，那可以被称之为最早的语音识别器。这种玩具狗具有基本的语音识别功能，当你发出命令时，它能从底座上弹出来。

在随后的几年里，AT&T 贝尔实验室的伙伴们在不懈的努力下实现了伟大的突破，开发出了最早的基于电子计算机的语音识别系统——Audrey 语音识别系统。这个系统在当时震惊了人工智能界，她能够识别 10 个英文数字，识别方法则是跟踪语音中的共振峰。此系统得到了 98%的正确率。

随后的 1950 年代末，语法概率的加入；1960 年代，人工神经网络的引入等等。

随着先驱者们的努力探索与实现，语音识别技术发展至今，其技术已经完全可以满足我们通常应用的需求。我们有众所周知的苹果智能机器人 Siri、上海第一家无人银行等都离不开语音识别技术。在未来，在各个行业各个领域都会有大幅度的应用。毫无疑问，语音识别系统在日后的应用会更加广泛。

# 2　语音识别的基本过程

语音识别系统架构如图 2 所示，首先就是获取语音信号，其实获取信号手法有很多，例如现实生活中的话筒、工业上的信号采集器等，而我们的语音信号的采集，则可以简单地通过个人计算机的语音收入插件来完成。获取语音后经过预处理模块，该模块主要作用是使采集到的语音信号更加平滑，其对于后续信号处理是不可或缺的重要环节。预处理模块包括预加重、加窗分帧以及端点检测部分；信号经过预处理模块后，接下来的一步就是对处理过的信号进行特征提取，于是信号进入到特征值提取模块。在该模块内，包括的方法有：MFCC、LPCC、LPC 系数等，而本论文采用了模仿人耳结构的 MFCC 系数；是一种根据人耳识别声音的方法而开发出来的声音识别算法。近年来，语音识别技术不断提高。未来在医学、工业、教育甚至是进入家家户户都是有必要的，语音识别技术也被列入为计算机发展十大技术之一，很是重要。特征提取后进行模型匹配模块，在这一块分为训练和识别算法。最后输出识别结果。

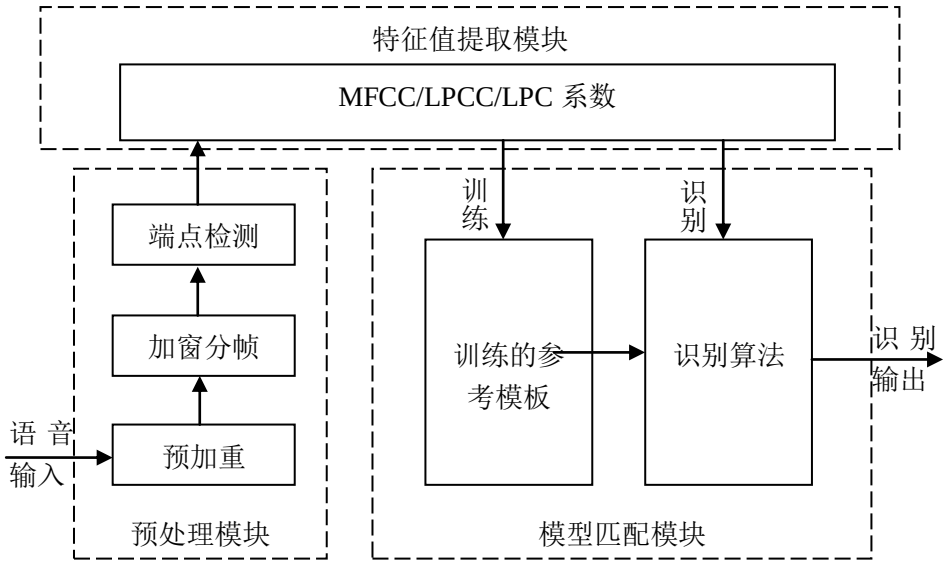


**图 2 语音识别系统架构图**

## 2.1　声音的读取

在声音的读取上，其实难度不是很大，尤其是在MATLAB这样一个集成度很高的环境。在读取上，通过简单的MATLAB程序可以做到直接通过计算机的话筒将语音信号录入，该信号由模拟信号转变成数字电信号后进行后续处理。其中，由于MATLAB是一个非常强大的编写器，东西特别多，所以很多功能包过于庞大而不足以放入环境之中，为了使梅尔频率倒谱系数中的melbankm函数能够实现，首先，我们需要在MATLAB中安装语音工具箱voicebox，该工具箱的添加方法最重要的是路径问题。首先将该工具箱放入MATLAB安装包的toolbox目录下后，需要在程序中加入该路径，然后再开始程序的运行。声音的读取函数中录音函数使用audiorecorder，该函数是针对声卡采集声音信号的一个函数。其设置录音条件有三个，分别是采样率、采样点的比特数以及声道。本论文中，我的采样率设为44.1kHz，这是wav音频的频率，采用该音频的原因是现今大多都是支持wav音频；同时该论文的采样点的比特数设为16；声道设为单声道。接下来就是何时开始录音何时结束录音。其中recordblocking函数是用来设置采集到的音频录制持续时间的函数。最后创建一个figure，即创建一个窗口用于放图，最后将录入的语音用plot函数画出来，过程描述如下：

MATLAB 程序：

```
clc;clear;close all;
% clear清除workspace里的内容；
% clc关闭所有command的命令；
% close all关闭所有figure；
path('D:\Program Files\matlab\toolbox\voicebox',path)
%  添加语音插件（此为本电脑插件路径）
R = audiorecorder(44100,16,1);
%  采样点的比特数为16；
% 1为单声道，2为双声道。
disp('Start speaking.')
%  开始录音
recordblocking(R,3);
% R条件下的录音行为坚持3s
```

```
disp('End of Recording.');
% 结束录音
play(R);
% 播放录音，即录入声音的回放
myRecording = getaudiodata(R);
% 获取录音数据，即将模拟信号转变为数字电信号
figure
%创建一个窗口用于放图
plot(myRecording);
%画出myRecording的时域图
```

图3是语音发出的原始信号获取图，图4是尖锐声音发出的原始信号获取图。

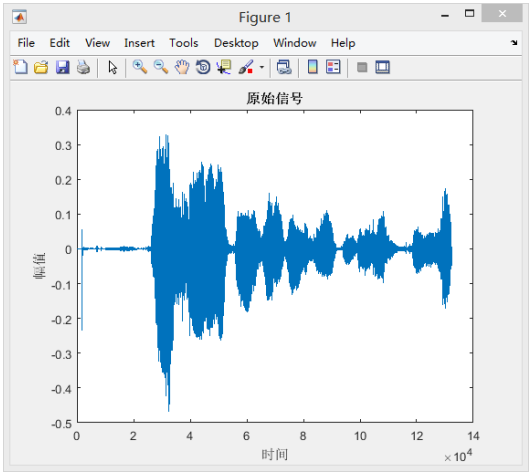


**图 3 语音原始信号获取**

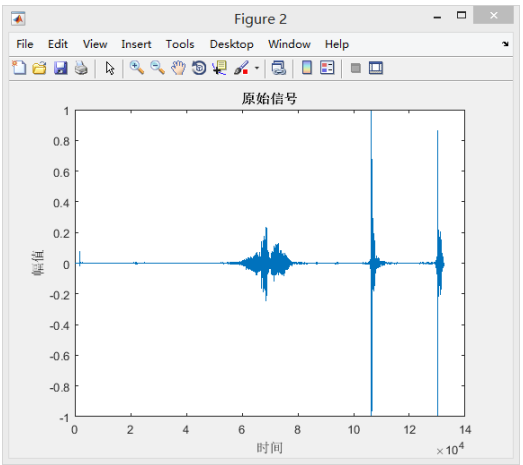


**图 4 尖锐敲击原始信号获取**

## 2.2 语音信号的预处理

在对采集到的语音信号进行特征提取之前，首先最基础的就是对语音信号进行预处理。我们都知道，当我们采集信号时，由于采集语音信号的设备不同，再加上人类本身的发声器官会带有的高频、高次谐波失真、混叠等因素，这些都会使信号不准确，出现失真现象。而预处理的作用就在于消除这些因素，使得信号更平滑更均匀，使参数更精确，从而减少信号后续处理会遇到的各种失真、不准确性，从而提高语音处理的质量。

预处理十分重要，不能出现差错，其操作的正确与否，直接可以影响后续一系列识别算法的精度，甚至会影响到整个语音识别系统的准确性。

预处理包括：预加重—分帧加窗—端点检测。

### 2．2．1 预加重

什么是预加重呢？预加重其实是通过高通滤波器——当然我们通常直接称之为预加重滤波器——将获取的信号中那些具有干扰作用的低频信号消除。人的声音是有高低频之分的，而噪音干扰是不可避免的，所以为了消除信号的这种干扰，预加重就是必不可少的。从示例图中我们不难看到，预加重的主要作用就是通过高通滤波器提升高频部分的频谱，增加辨识度，以便于语音参数的分析。

$$H(z) = 1-\lambda z^{-1} \tag{1}$$

以下是预加重前后信号的关系式：

$$Y(n) = S(n)-\lambda S(n-1) \tag{2}$$

其中，系数$\lambda$的取值一般在 0.93-1.0 之间，常见的取值是 0.9375，实验中，我的取值是 0.97。

MATLAB 程序：

```
Y = filter(B,A,X);
%使用滤波器
```

以下图 5 以及图 6 是一段语音信号在预加重前及预加重后的时域图、频域的

幅值图以及相位图：

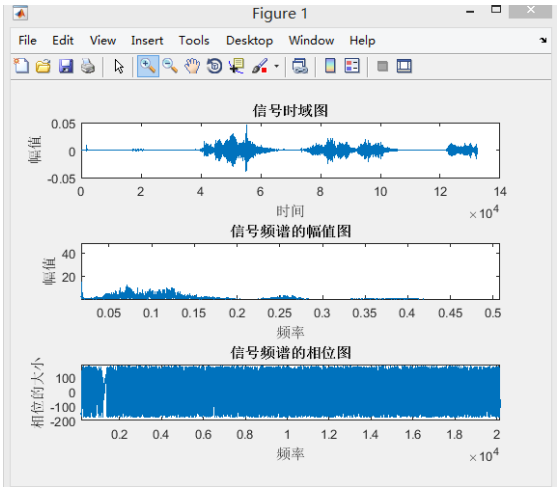


**图 5 预加重前的时域图、频域的幅值图以及相位图**

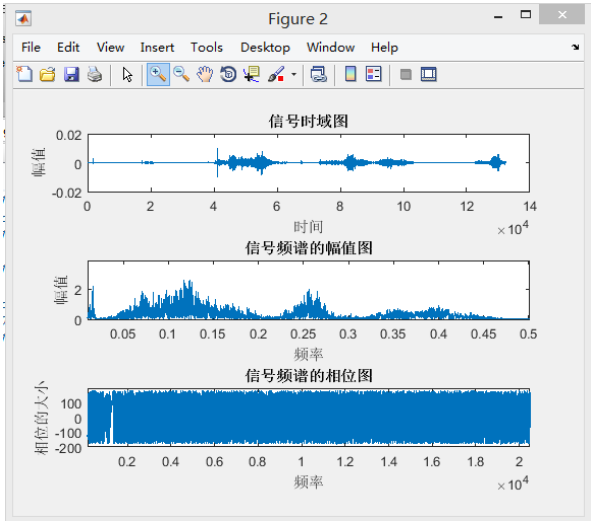


**图 6 预加重后的时域图、频域的幅值图以及相位图**

### 2．2．2 分帧加窗

虽然声音信号是一种时变信号，但是由于声音震动速率比其发音器官的物理运动快得多，所以在 10ms 到 30ms 时间段里，几乎可以认为语音信号的频率是固定不变的，所以可以对信号进行分帧加窗。也因为 fft（快速傅里叶变换）对应的是无限信号，分帧后无限信号就变成了有限信号，而分帧信号再进行 fft 后，会造成高频“泄露”，所以必须加窗改善“泄露”现象。

数字信号处理中，比较常见的窗函数有：矩形窗、汉宁窗以及汉明窗。

在数学公式中，其公式是：

矩形窗：

$$W(n)=\begin{cases}1,0\le n\le L-1\\0,\text{其他}\end{cases} \tag{3}$$

汉宁窗：

$$W(n)=\begin{cases}0.5\left[1-\cos\left(\dfrac{2\pi n}{L-1}\right)\right],0\le n\le L-1\\0\ ,\text{其他}\end{cases} \tag{4}$$

汉明窗：

$$W(n)=\begin{cases}0.54\text{-}0.46\cos\left(\dfrac{2\pi n}{L-1}\right),0\le n\le L-1\\0\ ,\text{其他}\end{cases} \tag{5}$$

MATLAB 使用函数：

```
wvtool(hamming(64),hann(64),boxcar(64));
%此为汉明窗、汉宁窗以及矩形窗可视化工具窗函数
```

如下图 7 所示的是矩形窗、hanning 窗以及 hamming 窗函数的可视化工具。

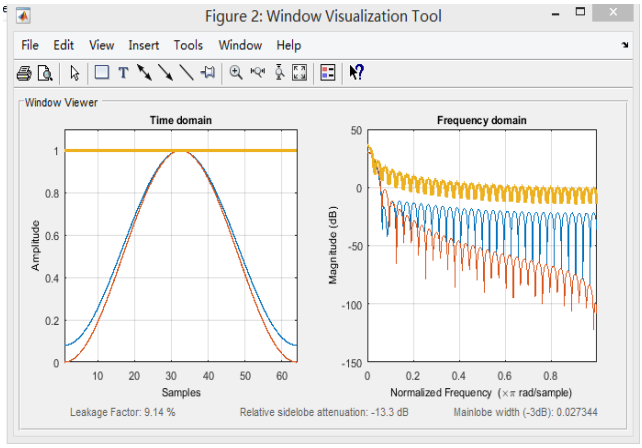


**图 7 显示窗函数的 GUI 工具**

矩形窗：从图 7 中我们可以清楚的看到，矩形窗的主瓣对比于其他两个窗而言较高，同时其旁瓣也比较高，这在信号处理中并不是一个好现象，相邻谐波影响比较严重，信号处理失真。

汉宁窗：从图 7 中可以看出（红色线条），汉宁窗主瓣高，旁瓣与主瓣易于区分。

汉明窗：从图 7 中蓝色线条可以看出，汉明窗相对于汉宁窗，其旁瓣更小，主旁瓣差距更显著，更易于区分，相邻谐波影响小。

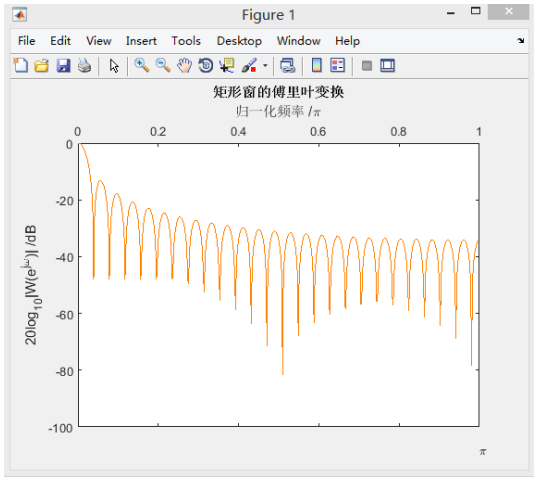


**图 8 矩形窗做 FFT**

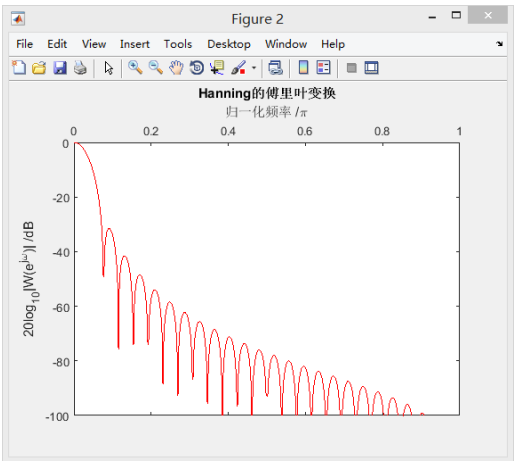


**图 9 Hanning 的 FFT**

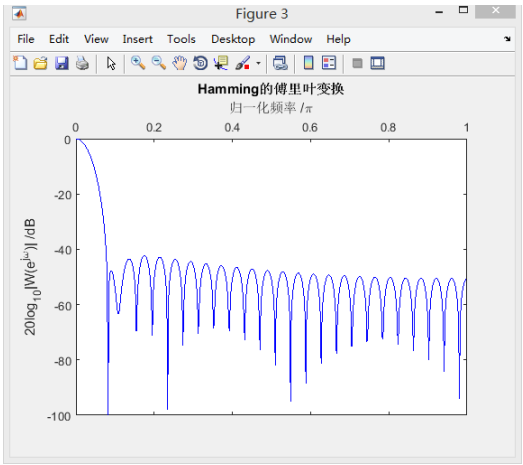


**图 10 Hamming 的 FFT**

图 8、图 9、图 10 是对同一信号利用三种窗口做 FFT 变换后的结果图，从图中就可以很明显的看到主瓣旁瓣的差异，可以看出汉明窗在提升高频并减少谐波影响上的作用是很显著的，过程描述如下：

MATLAB 使用函数：

```
%矩形窗做 FFT
wn =y'.* rectwin(256) ;
[h1,w] = freqz(wn,1);
figure;
plot(w/pi,20*log10(abs(h1/max(h1))),'Color',[1 0.5 0]);
```

```
%hanning 做 FFT
wn =y'.* hanning(256) ;
[h1,w] = freqz(wn,1);
figure;
plot(w/pi,20*log10(abs(h1/max(h1))),'Color',[1 0.5 0]);
%hamming 做 FFT
wn =y'.* hamming(256) ;
[h1,w] = freqz(wn,1);
figure;
plot(w/pi,20*log10(abs(h1/max(h1))),'Color',[1 0.5 0]);
```

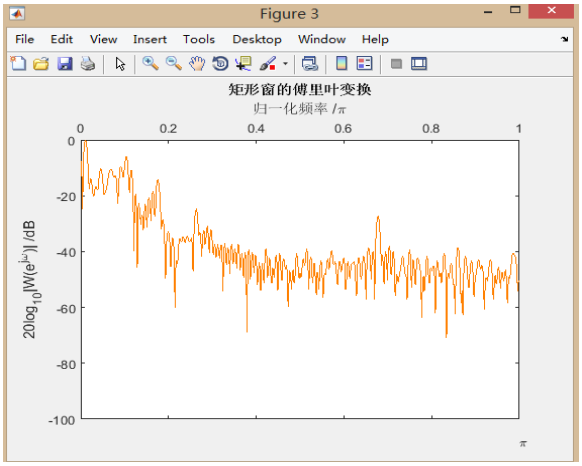


**图 11 矩形窗在程序中的 FFT**

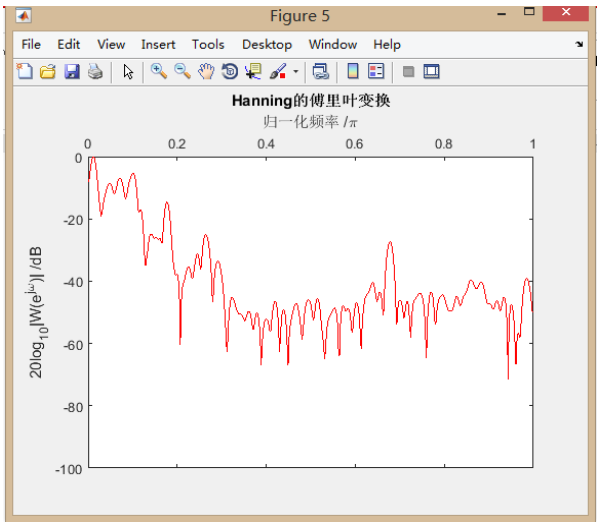


**图 12 hanning 窗在程序中的 FFT**

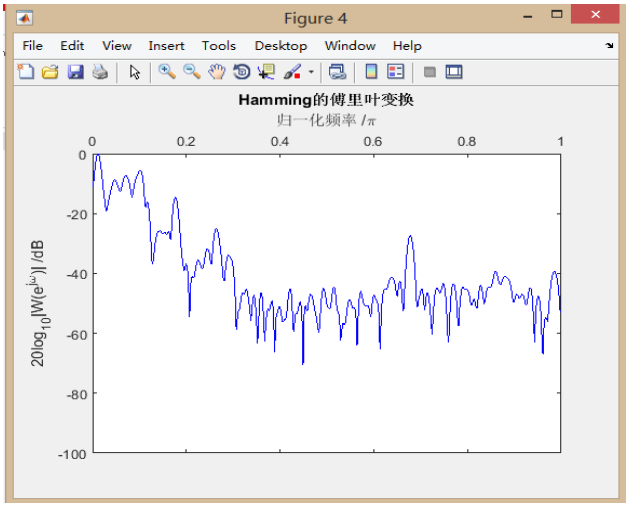


**图 13 hamming 窗在程序中的 FFT**

图 11 所示是矩形窗在完成程序后的 FFT 显示图；图 12 所示是 hanning 窗在完成程序后的 FFT 显示图；图 13 所示是 hamming 窗在完成程序后的 FFT 显示图。

### 2．2．3　端点检测

端点检测——语音识别中的重点，端点检测是为了去除掉语音信号中的静音部分，检测出有用的语音信号。这可以大大提高语音系统的识别率，还可以减少语音识别的计算量。

目前有很多种端点检测方法，例如滑动窗口语音端点检测、频带方差的端点检测、双门限比较法端点检测和分型技术的端点检测算法等等。其中双门限比较法最常用，是根据语音信号的短时过零率以及短时能量来找到语音信号的起止点。以下是端点检测算法：

（1） 语音信号的短时能量：

$$E=\sum_{m=-\infty}^{\infty} T\left[x(m)\right]w(n-m)=\sum_{m=-\infty}^{\infty}\left[x(m)w(n-m)\right]^2=\sum_{m=n}^{n+N-1}x(m)^2h(n-m)$$
$$=x^2(n)*h(n) \qquad (6)$$

式（6）中，N 代表的是窗函数长度，h(n)代表的是窗函数，即 w2(n)，其主要区分语音的轻音和浊音。而计算短时能量需要调整能量门限。

（2）语音信号的短时过零率：

$$Z_{\mathrm{n}}=\sum_{n=-\infty}^{\infty}\left|\operatorname{sgn}\left[x(n)\right]-\operatorname{sgn}\left[x(n-1)\right]\right|\cdot w(n-m) \tag{7}$$

• 其中，sgn[]是符号函数：

$$\operatorname{sgn}(n)=\begin{cases}1\ ,\ x(n)\geq 0\\ 0\ ,\ x(n)<0\end{cases} \tag{8}$$

$$\mathrm{w}(n)=\begin{cases}\dfrac{1}{2N}\ ,\ 0\leq n\leq N-1\\ 0\ ,\ \text{其他}\end{cases} \tag{9}$$

• 过零率，可以粗略地描述信号频谱分析，也是用于区分和判断清音和浊音。

（3）开始端点检测

图 14 是端点检测过程图，其显示了采集到的初始信号以及短时能量图，最后是短时过零率显示图。

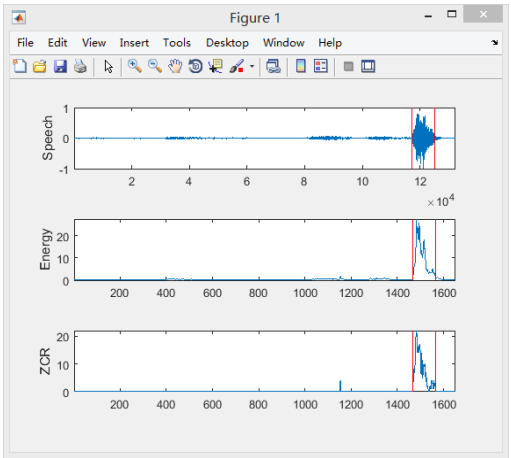


**图 14 端点检测图示**

## 2．3 基于 MFCC 算法语音信号特征值的提取

MFCC 的全名为 Mel-Frequency Cepstral Coefficients，译为：Mel 频率倒谱系

数；它包含两个步骤：

步骤一：转梅尔频率

步骤二：进行倒谱分析

### 2．3．1　梅尔频率

梅尔刻度是基于彼此等距的听众对音高(pitch)的感官判断的刻度。梅尔频率与普通频率（赫兹）的关系为：

$$m = 2595\log_{10}\left(1+\frac{f}{700}\right) \tag{10}$$

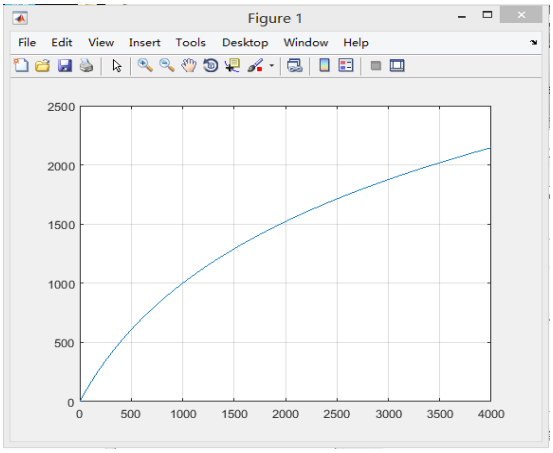


**图 15 m 与 f 的关系**

从图 15 中可以看出，f 越大，m 的上升斜率越小，也就是说若是等分梅尔刻度频率，赫兹频率的距离会越来越大，即如图 16 所示。

MATLAB 使用函数：

```
% 两频率关系图
f=1:4000;
mel(f)=2595*log10(1+f/700);
plot(f,mel(f));
grid on
```

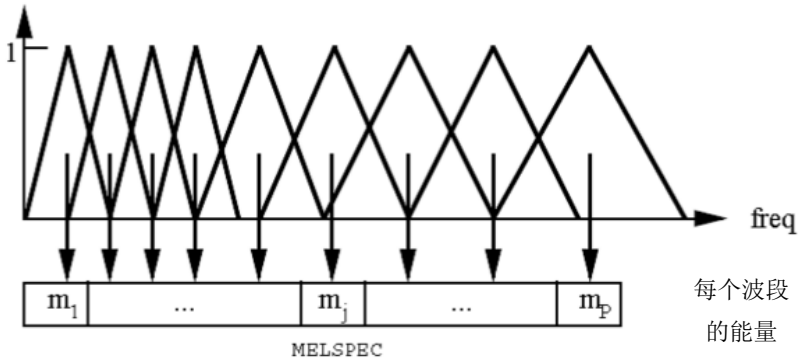


**图 16 梅尔刻度与频率分度对应图**

图 16 是梅尔刻度与频率分度对应图，即当梅尔刻度平均划分时，频率分度是呈现越来越大的趋势的。

### 2．3．2　倒谱分析

算法也是人工智能中非常重要的环节之一。好的算法，可以减少运算量，加快运行速度，是衡量任何一种计算机运行或编写的程序是否优秀的重要标准之一。倒谱具体的物理意义是：用于分解信号，将两信号的卷积转变为两信号的相加。很明显，用数学视角来看，就是利用取对数的方法将乘法简单化为加法。其具体操作步骤就是：先对时域信号做 DFT，变为频域信号，然后取 log，使卷积变成相加，最后再做 IDFT。

假设频率谱为 X(k)，时域信号为 x(n)，那么对时域信号做傅里叶得到频域信号：

$$X\left(k\right)=DFT\left(x\left(n\right)\right) \tag{11}$$

将频域 X(k)拆分为两部分的乘积 H(k)和 E(k)：

$$X\left(k\right)=H\left(k\right)E(k) \tag{12}$$

记这两部分对应的时域信号分别是 h(n)和 e(n)，则：

$$x(n)=h(n)*e(n) \tag{13}$$

然而此时，我们并不能够将 h(n)和 e(n)区分开，所以此时要对频域两边取 log：

$$\log(X(k))=\log(H(k))+\log(E(\mathrm{k})) \tag{14}$$

然后进行反傅里叶变换：

$$IDFT(\log(X(k)))=IDFT(\log(H(k)))+IDFT(\log(E(\mathrm{k}))) \tag{15}$$

记为：

$$x'(n)=h'(n)+e'(n) \tag{16}$$

当然此时的时域信号 x(n)已经是倒谱，与(13)中的 x(n)不一样。以上式子从（11）至（16）的计算过程中很清楚的将倒谱过程，也就是将卷积变为线性相加的过程描绘了出来。

## 2．4　MFCC 结果图

以下是几个语音原始信号及其做完 MFCC 之后出的结果图。梅尔频率倒谱系数是在多种方法尝试后，最终确定的通过模拟人类的听觉特征，而发明的一种思想。其意在通过“耳朵的结构”将语音的特征提取出来。

如图 17，是采集的尖锐刺激原始信号图以及该原始信号图经过 MFCC 的结果示例图。

如图 18，是采集到的人声音的原始信号图以及该原始信号图经过 MFCC 的结果示例图。

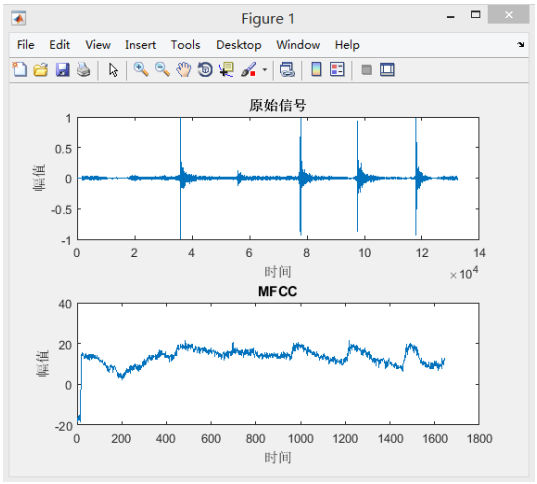


**图 17 尖锐原始信号及其 MFCC 结果图**

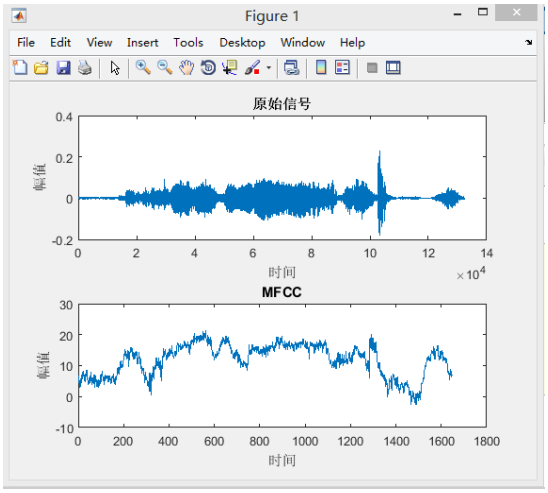


**图 18 语音原始信号及其 MFCC 结果图**

# 3 支持向量机

支持向量机 SVM 是一种常见的用于分类回归的机器学习算法。用于制作训练参考模板，基于此次重点在于 MFCC 特征提取，所以此处简单介绍一下神经网络的 SVM。

## 3．1 基本原理

SVM 是神经网络的一种，最典型的分类案例就是葡萄酒分类案例。其基本原理就是将待分类的点映射到“高维”（一定是比原来的维度要高至少 1 维）并且用线性函数

$$f(x)=w\cdot x+b \tag{17}$$

来构造分类器，以此来训练特征。

线性函数（17）中 b 为偏置，x 是训练元组，w 是权重向量，w·x 是 x 和 w 的点积。

当某段数据需要被分类时，就需要发挥式（17）的作用来区分该数据段类别。至于到底是大于 0 分为类一还是小于 0 分为类一，这都是可以自己决定的。

本示例中，我令 f(x)>0 时为类一。

## 3．2 最大边缘距离分类图

事实上，可分类的线有很多很多，问题的关键是在于应该选择哪条？而针对此问题，SVM 是通过寻找“最大边缘距离分类线/面”。那么，问题来了，什么是边缘距离，又为什么要去寻找她呢？这其实就是支持向量机的关键以及其侧重点所在。

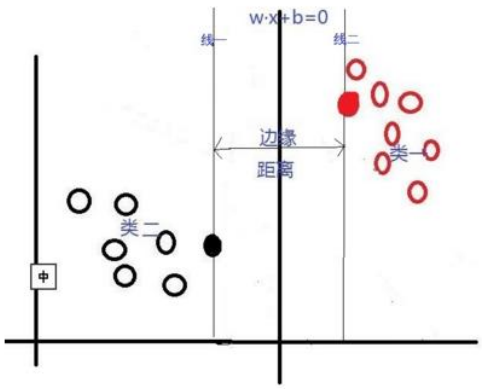


**图 19 一般边缘距离分类图**

如图 19 所示，在垂直的条件下，如果将线一最大限度的平移至最左边，前提条件为：必须将数据分成两类。图中线一正好穿过类二的实心黑点，并将数据分为两类。此时在线二右侧及线上数据记为类一，在线一左侧数据记为类二。同理，若是将此分类线最大限度的向左平移同时满足将数据分为两类，即如图 19 的线一，此时这条线正好穿过图中类二的实心点，在线一左侧及线上数据记为类二，在线一右侧数据记为类一。而图中这两条左右最大限度平移的线，就是支持线，当然在高维就是支持面，则用来确定支持线支持面的数据点就是支持向量。两个支持线或面之间的距离，称作边缘距离。

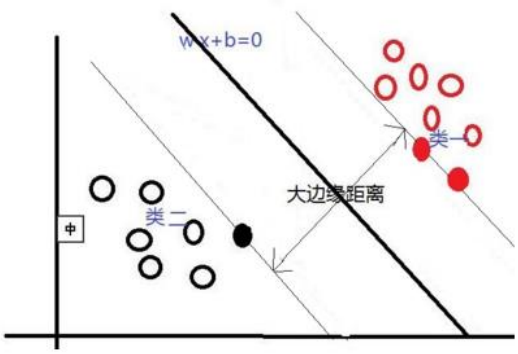


**图 20(a) 二维最大边缘距离分类图**

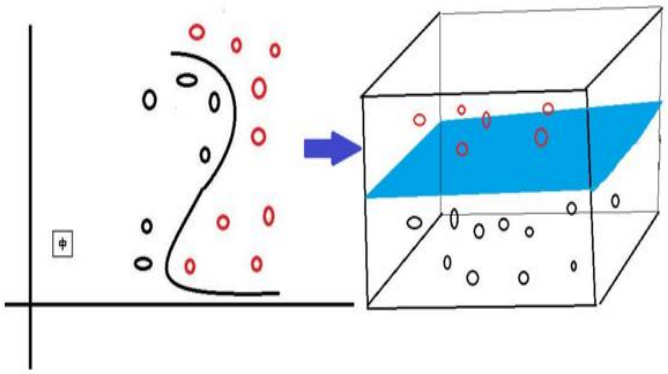


**图 20(b) 三维最大边缘距离分类图**

图 20(a)是二维的某最大边缘距离分类图；图 20(b)是三维的某最大边缘距离分类图。

由此我们能发现，不同的分类线/面显然是对应不同的支持线/面的，而支持线/面间的边缘距离也有着显著的不同，而分类线的边缘距离越大，分类精度就越有保障，因此，需要找到最好的分类线/面，也就是边缘距离最大的分类线/面。即：就线性可分而言，SVM 解决二分类的方法就是通过找到拥有最大边缘距离的分类线/面，同时该线/面平分边缘距离且平行于两个支持线/面。图 20(a)明显比图 19 拥有更大边缘距离的分类线/面。

### 3．3　SVM 分类图

图 21 是 SVM 神经网络机器学习算法获取的语音信号分类图。SVM 用于训练，但是需要大量的数据制作神经网络，所以本实验就简单做了两个对比的结果。

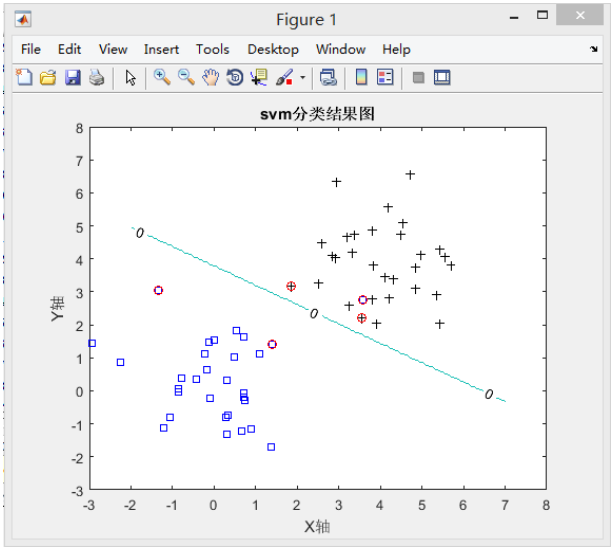


**图 21　SVM 分类结果图**

# 总结

以上是本人大四的毕业论文报告，完成的是基于频谱分析的 MFCC 研究与测试。近年来，无论是无人银行，还是 Alpha Go 的出现，人工智能发展的趋势已是势不可挡。所谓的人工智能——顾名思义就是让机器人拥有智能。而声音识别技术，就是机器智能体现的一方面。MFCC 算法，是一种根据人耳识别声音的方法而开发出来的声音识别算法。近年来，语音识别技术不断提高。未来在医学、工业、教育甚至是进入家庭都是有必要的，语音识别技术也被列入为计算机发展十大技术之一，可谓是很重要，而其中很重要的一方面就是语音信号的特征提取。本研究与测试主要就是针对于如何实现特征提取逐步解析。基本过程是：声音的读取、语音信号预处理、MFCC 特征提取。该研究与测试还涉及了一些 SVM（支持向量机）二分类器的知识。但由于实验时间有限，加上 SVM 是机器学习的一种，需要大量数据，以此来作为一个库，实现机器学习。所以在本研究与测试中，SVM 仅限于知识方面的讲解。通过本次毕业论文的学习，我深切的体会到完成一件事一定要专心致志，并且仔细阅读大量书籍。这样的集中学习可以说是知识储备质的飞越。

# 参考文献


[1]韩纪庆,张磊等.语音信号处理[M].北京:清华大学出版社,2004:9-11.

[2]张雷英等.数字语音处理及 Matlab 仿真[M]. 北京:电子工业出版社,2010:25-35.

[3]俞栋,邓力,等.解析深度学习---语音识别实践[M].北京:电子工业出版社,2016：135-150.

[4]郭武,王仁华,戴礼荣等.基于基音周期与清浊音信息的梅尔倒谱参数[J].数据采集与处理,2007,(2):71-76.

[5]王敏妲.语音识别技术的研究与发展[J].微机与应用,2009,(23):1-6.

[6]叶海民,戎蒙恬,邓晓东等.基于绝对值差的归一化波形匹配混合算法[J].信息技术,2011,(8):89-93.

[7]李宇,郭雷勇,谭洪舟等.基于噪声倒谱阈值频谱估计的语音活动检测[J].计算机工程2011,37(14):140-142.

[8]王宏志，徐玉超等. 基于 Mel 频率倒谱系数相似度的语音端点检测算法[J]. 吉林大学学报，2012,42(5):31-35.

[9]Krull.V,X.Luo,K.I.Kirk.Talker-identification training using simulations of binaurally combined electric and acoustic hearing Generalization to speech and emotion recognition[J].The Journal of the Acoustical Society of American,2012,131(4 pt.1):3069-3078.

[10]张怡然，白静等.基于多窗频谱估计和平滑幅度谱包络的 Mel 频率倒谱系数（MFCC）改进方法[J].科学技术与工程，2014,14(19):253-256.

[11]常飞,乔欣等.基于 MFCC 特征提取的故障检测与评价方法[J].计算机应用研究，2015,32(6):16-19.

[12]孙华娟 闫晓红 郝学元等.多值数据的自适应脉冲宽度调制预加重方法[J].物理学报，2015(1):372-378

[13]fuyfu. 英特尔芯片将加入语音识别技术,Siri 登陆桌面之日将近.http://finance.591hx.com.2016.

[14]王冠雄.声学建模中若干问题的研究[D].北京，北京邮电大学，2010:3-66.

[15]徐利军.基于 HMM 和神经网络的语音识别研究[D].湖北工业大学,2012.

[16]李云.基于 HMM 语音分组识别系统的研究[D].广东工业大学,2013.

# 致谢

本论文完成于天津商业大学宝德学院计算机与信息技术系。在这几个月的毕业设计中，遇到了很多问题也解决了很多问题。不得不说，成长了很多，也成长的很快。

在此期间我有许多想要感谢的人。首先要特别感谢论文指导教师丁玄功老师；感谢所有支持我并给予我资料、图片、文献及设想方法的人们；感谢给予我实验环境的天津商业大学宝德学院实验室。

# 附录

1. 图 7 显示窗函数的 GUI 工具

```
%添加路径 ： D:\Program Files\matlab\bin
clc;clear;close all;
wvtool(hamming(64),hann(64),boxcar(64))
```

2. 图 15 梅尔频率与赫兹频率的关系

```
f=1:4000;
mel(f)=2595*log10(1+f/700);
plot(f,mel(f));
grid on
```

3. 图 14 端点检测程序

```
R = audiorecorder(44100,16,1);
disp('Start speaking.')
recordblocking(R,3);
disp('End of Recording.');
回放录音数据
play(R);
获取录音数据
x = getaudiodata(R);

基于 Matlab 编写的语音端点检测程序
幅度归一化到[-1,1]
x = double(x);
x = x / max(abs(x));
常数设置
FrameLen = 240;
FrameInc = 80;
amp1 = 10;
amp2 = 2;
zcr1 = 10;
zcr2 = 5;
maxsilence = 8;  % 6*10ms  = 30ms
minlen  = 15;    % 15*10ms = 150ms
status  = 0;
count   = 0;
silence = 0;
计算过零率
tmp1  = enframe(x(1:end-1), FrameLen, FrameInc);
tmp2  = enframe(x(2:end)  , FrameLen, FrameInc);
```

```
    signs = (tmp1.*tmp2)<0;
    diffs = (tmp1 -tmp2)>0.02;
    zcr   = sum(signs.*diffs, 2);
    计算短时能量
    amp = sum(abs(enframe(filter([1 -0.9375], 1, x), FrameLen, FrameInc)),
2);
    调整能量门限
    amp1 = min(amp1, max(amp)/4);
    amp2 = min(amp2, max(amp)/8);
    开始端点检测
    x1 = 0;
    x2 = 0;
    for n=1:length(zcr)
       goto = 0;
       switch status
       case {0,1}
            0 = 静音,1 = 可能开始
          if amp(n) > amp1
      确信进入语音段
                x1 = max(n-count-1,1);
                status  = 2;
                silence = 0;
                count   = count + 1;
          elseif amp(n) > amp2 | ... % 可能处于语音段
                 zcr(n) > zcr2
                status = 1;
                count  = count + 1;
          else
                静音状态
                status  = 0;
                count   = 0;
          end
       case 2,
                2 = 语音段
          if amp(n) > amp2 | ...     % 保持在语音段
                zcr(n) > zcr2
                count = count + 1;
          else
                语音将结束
                silence = silence+1;
                if silence < maxsilence % 静音还不够长，尚未结束
                   count  = count + 1;
                elseif count < minlen   % 语音长度太短，认为是噪声
                   status  = 0;
```

```
            silence = 0;
            count   = 0;
          else
            语音结束
            status  = 3;
          end
        end
      case 3,
        break;
      end
    end
    count = count-silence/2;
    x2 = x1 + count -1;
    subplot(311)
    plot(x)
    axis([1 length(x) -1 1])
    ylabel('Speech');
    line([x1*FrameInc x1*FrameInc], [-1 1], 'Color', 'red');
    line([x2*FrameInc x2*FrameInc], [-1 1], 'Color', 'red');
    subplot(312)
    plot(amp);
    axis([1 length(amp) 0 max(amp)])
    ylabel('Energy');
    line([x1 x1], [min(amp),max(amp)], 'Color', 'red');
    line([x2 x2], [min(amp),max(amp)], 'Color', 'red');
    subplot(313)
    plot(zcr);
    axis([1 length(zcr) 0 max(zcr)])
    ylabel('ZCR');
    line([x1 x1], [min(zcr),max(zcr)], 'Color', 'red');line([x2 x2],
[min(zcr),max(zcr)], 'Color', 'red');
```

4. 图 8 图 9 图 10 是窗函数的 FFT

```
    N =51
    求矩形窗的频率响应图
    wn =  rectwin(51) ;  %矩形窗函数
    20*log10(abs(WN))
    [h1,w] = freqz(wn,1);
    figure;
    plot(w/pi,20*log10(abs(h1/max(h1))));
    axis([0 1 -100 0]);
    xlabel('归一化频率 /\pi');
    ylabel('20log_{10}|W(e^{j\omega})| /dB');
    title('矩形窗的傅里叶变换');
```

```
set(gca,'YTick',[-100 -80 -60 -40 -20 0])
set(gca,'XTick',[0 :0.2: 1])
set(gca,'XAxisLocation','top'); %设置 X 轴在上方
set(gca,'YAxisLocation','left'); %设置 Y 轴在左方
text(1,-108,'\pi');%gtext('\pi');
```
求三角窗的频率响应图
```
wn1 = bartlett(51);
[h1,w1] = freqz(wn1,1);
figure(2);
plot(w/pi,20*log10(abs(h1/max(h1))));
axis([0 1 -100 0]);
xlabel('归一化频率 /\pi');
ylabel('20log_{10}|W(e^{j\omega})| /dB');
title('三角窗的傅里叶变换');
set(gca,'YTick',[-100 -80 -60 -40 -20 0])
set(gca,'XTick',[0 :0.2: 1])
set(gca,'XAxisLocation','top');%设置 X 轴在上方
set(gca,'YAxisLocation','left'); %设置 Y 轴在左方
hanning
wn1 = hanning(51) ;
[h1,w1] = freqz(wn1,1);
figure;
plot(w/pi,20*log10(abs(h1/max(h1))));
axis([0 1 -100 0]);
xlabel('归一化频率 /\pi');
ylabel('20log_{10}|W(e^{j\omega})| /dB');
title('Hanning 的傅里叶变换');
set(gca,'YTick',[-100 -80 -60 -40 -20 0]);
set(gca,'XTick',[0 :0.2: 1]);
set(gca,'XAxisLocation','top');%设置 X 轴在上方
set(gca,'YAxisLocation','left'); %设置 Y 轴在左方
hamming
wn1 = hamming(51) ;
[h1,w1] = freqz(wn1,1);
figure;
plot(w/pi,20*log10(abs(h1/max(h1))));
axis([0 1 -100 0]);
xlabel('归一化频率 /\pi');
ylabel('20log_{10}|W(e^{j\omega})| /dB');
title('Hamming 的傅里叶变换');
set(gca,'YTick',[-100 -80 -60 -40 -20 0])
set(gca,'XTick',[0 :0.2: 1])
set(gca,'XAxisLocation','top');%设置 X 轴在上方
set(gca,'YAxisLocation','left'); %设置 Y 轴在左方
```

```
Blackman
wn1 = blackman(51) ;
[h1,w1] = freqz(wn1,1);
figure(5);
plot(w/pi,20*log10(abs(h1/max(h1))));
axis([0 1 -100 0]);
xlabel('归一化频率 /\pi');
ylabel('20log_{10}|W(e^{j\omega})| /dB');
title('Blackman 的傅里叶变换');
set(gca,'YTick',[-100 -80 -60 -40 -20 0])
set(gca,'XTick',[0 :0.2: 1])
set(gca,'XAxisLocation','top');%设置 X 轴在上方
set(gca,'YAxisLocation','left'); %设置 Y 轴在左方
```

5. 图 5 预加重前的时域图、频域的幅值图以及相位图

```
path('D:\Program Files\matlab\toolbox\voicebox',path)
R = audiorecorder(44100,16,1);
disp('Start speaking.')
recordblocking(R,3);
disp('End of Recording.');
回放录音数据
play(R);
获取录音数据
myRecording = getaudiodata(R);
绘制录音数据波形
figure
subplot(3,1,1);
plot(myRecording);title('信号时域图');xlabel('时间');ylabel('幅值');
n=length(myRecording);
fs=44100;
t=0:1/fs:(n-1)/fs;
a=abs(fft(myRecording));
subplot(3,1,2);
plot(t,a);title('信号频谱的幅值图');xlabel('频率');ylabel('幅值');
b=angle(fft(myRecording))*180/pi;
subplot(3,1,3);
plot(b);title('信号频谱的相位图');xlabel('频率');ylabel('相位的大小');
```

6. 图 6 预加重后的时域图、频域的幅值图以及相位图

```
y=filter([1 -0.97],1,myRecording);
figure
subplot(3,1,1);
plot(y);title('信号时域图');xlabel('时间');ylabel('幅值');
```

```
n=length(y);
fs=44100;
t=0:1/fs:(n-1)/fs;
a=abs(fft(y));
subplot(3,1,2);
plot(t,a);title('信号频谱的幅值图');xlabel('频率');ylabel('幅值');
b=angle(fft(y))*180/pi;
subplot(3,1,3);
plot(b);title('信号频谱的相位图');xlabel('频率');ylabel('相位的大小');
```

7.MFCC

```
bank=melbankm(24,256,16000,0,0.4,'m');
```

Mel 滤波器的阶数为 24，FFT 变换的长度为 256，采样频率为 16000Hz，最低的滤波器的低端的起始值为 0，最高的滤波器的高端最高值为 0.4，m 表示 hamming 窗,t 表示三角窗。返回值为一个包含滤波幅度值的稀疏矩阵

归一化 Mel 滤波器组系数

```
bank=full(bank); %full()将稀疏矩阵转换为完整矩阵
bank=bank/max(bank(:)); %归一化 bank 矩阵中的值 %max（bank（：））取出 bank 矩阵中的最大值
```

dct 系数，12*2

```
for k=1:12
n=0:23;
dctcoef(k,:)=cos((2*n+1)*k*pi/(2*24));
end
```

归一化倒谱提升窗口

```
w=1+6*sin(pi*(1:12)./12);
w=w/max(w);
```

声音的预处理（预加重-分帧-加窗）

预加重滤波器，提升高频，突出高频的共振峰

```
xx=double(myRecording);  %将音频信号 x 转换为双精度的值
xx=filter([1 -0.97],1,xx);
```

H(Z)=1-a*z^-1,其中 a 的值取 0.9-1 之间，通常取 0.97

Y = filter(B,A,X) ，输入 X 为滤波前序列，Y 为滤波结果序列，B/A 提供滤波器系数，B 为分子， A 为分母

```
xx=enframe(xx,256,80);%对 xx 256 点分为一帧原始 12 维的 mfcc
for i=1:size(xx,1)%size(xx,1)表示分帧以后的帧数
y=xx(i,:);%提取每一帧数据赋值给 y
```

加窗

```
s=y'.*hamming(256);
```

将每一帧信号乘以汉明窗（）；

对 xx 256 点分为一帧，帧之间的移动样点数为 80，非重叠部分为 80

```
%求一帧信号的能量
```

```
p=0;
p=s'*s;
p1=10*log10(p);

t=abs(fft(s));%FFT 快速傅里叶变换，得到各帧的频谱
t=t.^2;%对语音信号的频谱取模平方，得到各帧的功率谱  %点乘，对应位置的元素相乘
c1=dctcoef*log(bank*t(1:129));
c2=c1.*w';
m(i,:)=c2;
m(i,:)=[c2' p1];
end


求一阶差分系数
dtm=zeros(size(m));
for i=3:size(m,1)-2
dtm(i,:)=-2*m(i-2,:)-m(i-1,:)+m(i+1,:)+2*m(i+2,:);
end
dtm=dtm/3;
求取二阶差分系数
dtmm=zeros(size(dtm));
for i=3:size(dtm,1)-2
dtmm(i,:)=-2*dtm(i-2,:)-dtm(i-1,:)+dtm(i+1,:)+2*dtm(i+2,:);
end
dtmm=dtmm/3;
合并 mfcc 参数和一阶差分 mfcc 参数
ccc=[m dtm dtmm];
去除首尾两帧，以为这两帧的一阶差分参数为 0
ccc=ccc(3:size(m,1)-2,:);
ccc;
subplot(2,1,1);
ccc_1=ccc(:,1);
plot(ccc_1);title('MFCC');xlabel('时间');ylabel('幅值');
plot(ccc);title('MFCC');xlabel('时间');ylabel('幅值');

[h,w]=size(ccc);
A=size(ccc);
subplot(2,1,2);
plot([1,w],A);
xlabel('维数');ylabel('幅值');
title('维数与幅值的关系');
```

8. SVM 函数

```
%% ------------主函数----------------
C = 10;  %约束参数
kertype = 'linear';  %线性核

%% ①------数据准备
n = 30;
%randn('state',6);   %指定状态，一般可以不用
x1 = randn(2,n);    %2 行 N 列矩阵，元素服从正态分布
y1 = ones(1,n);       %1*N 个 1
x2 = 4+randn(2,n);   %2*N 矩阵，元素服从正态分布且均值为 5，测试高斯核可 x2 = 3+randn(2,n);
y2 = -ones(1,n);      %1*N 个-1

figure;  %创建一个用来显示图形输出的一个窗口对象
plot(x1(1,:),x1(2,:),'bs',x2(1,:),x2(2,:),'k+');  %画图，两堆点
axis([-3 8 -3 8]);  %设置坐标轴范围
hold on;    %在同一个 figure 中画几幅图时，用此句

%% ②-------------训练样本
X = [x1,x2];        %训练样本 2*n 矩阵，n 为样本个数，d 为特征向量个数
Y = [y1,y2];        %训练目标 1*n 矩阵，n 为样本个数，值为+1 或-1
svm = svmTrain(X,Y,kertype,C);  %训练样本
plot(svm.Xsv(1,:),svm.Xsv(2,:),'ro');   %把支持向量标出来

%% ③-------------测试
[x1,x2] = meshgrid(-2:0.05:7,-2:0.05:7);  %x1 和 x2 都是 181*181 的矩阵
[rows,cols] = size(x1);
nt = rows*cols;
Xt = [reshape(x1,1,nt);reshape(x2,1,nt)];
%前半句 reshape(x1,1,nt)是将 x1 转成 1*（181*181）的矩阵，所以 xt 是 2*（181*181）的矩阵
%reshape 函数重新调整矩阵的行、列、维数
Yt = ones(1,nt);

result = svmTest(svm, Xt, Yt, kertype);

%% ④--------------画曲线的等高线图
Yd = reshape(result.Y,rows,cols);
contour(x1,x2,Yd,[0,0],'ShowText','on');%画等高线
title('svm 分类结果图');
x1=xlabel('X 轴');
x2=ylabel('Y 轴');

%% -----------训练样本的函数---------
```

```
function svm = svmTrain(X,Y,kertype,C)

% Options 是用来控制算法的选项参数的向量，optimset 无参时，创建一个选项结构所有字段为默认值的选项
options = optimset;
options.LargeScale = 'off';%LargeScale 指大规模搜索，off 表示在规模搜索模式关闭
options.Display = 'off';    %表示无输出

%二次规划来求解问题，可输入命令 help quadprog 查看详情
n = length(Y);  %返回 Y 最长维数
H = (Y'*Y).*kernel(X,X,kertype);
f = -ones(n,1); %f 为 1*n 个-1,f 相当于 Quadprog 函数中的 c
A = [];
b = [];
Aeq = Y; %相当于 Quadprog 函数中的 A1,b1
beq = 0;
lb = zeros(n,1); %相当于 Quadprog 函数中的 LB，UB
ub = C*ones(n,1);
a0 = zeros(n,1);  % a0 是解的初始近似值
[a,fval,eXitflag,output,lambda]                                    =
quadprog(H,f,A,b,Aeq,beq,lb,ub,a0,options);
%a 是输出变量，问题的解
%fval 是目标函数在解 a 处的值
%eXitflag>0,则程序收敛于解 x;=0 则函数的计算达到了最大次数;<0 则问题无可行解，或程序运行失败
%output 输出程序运行的某些信息
%lambda 为在解 a 处的值 Lagrange 乘子

epsilon = 1e-8;
 %0<a<a(max)则认为 x 为支持向量,find 返回一个包含数组 X 中每个非零元素的线性索引的向量。
sv_label = find(abs(a)>epsilon);
svm.a = a(sv_label);
svm.Xsv = X(:,sv_label);
svm.Ysv = Y(sv_label);
svm.svnum = length(sv_label);
%svm.label = sv_label;
end
%% ---------------测试的函数-------------
function result = svmTest(svm, Xt, Yt, kertype)
temp = (svm.a'.*svm.Ysv)*kernel(svm.Xsv,svm.Xsv,kertype);
%total_b = svm.Ysv-temp;
b = mean(svm.Ysv-temp);  %b 取均值
```

```
    w = (svm.a'.*svm.Ysv)*kernel(svm.Xsv,Xt,kertype);
    result.score = w + b;
    Y = sign(w+b);  %f(x)
    result.Y = Y;
    result.accuracy = size(find(Y==Yt))/size(Yt);
    end

    %% --------------核函数---------------
    function K = kernel(X,Y,type)
    %X 维数*个数
    switch type
    case 'linear'   %此时代表线性核
        K = X'*Y;
    case 'rbf'      %此时代表高斯核
        delta = 5;
        delta = delta*delta;
        XX = sum(X'.*X',2);%2 表示将矩阵中的按行为单位进行求和
        YY = sum(Y'.*Y',2);
        XY = X'*Y;
        K = abs(repmat(XX,[1 size(YY,1)]) + repmat(YY',[size(XX,1) 1]) -
2*XY);
        K = exp(-K./delta);
    end
    end
```