
# 语音识别

## 5.1 语音识别概述

语音识别(speech recognition)技术就是让机器识别和理解人类的语言并将其转换为相应的文本或者命令,也就是让机器听懂人类的语音。语音识别过程中,语音会和环境音、噪声等一起被送到机器中,经过去噪后才能够对语音进行采集和翻译,把声音转换成需要的文本内容。

语音识别的应用实际上改变了人机的交互方式。以前的人机交互主要通过键盘或鼠标,键盘和鼠标把人类与计算机绑在一起,需要在距离计算机比较近的地方实施操作。有了语音输入后,我们与计算机或者其他电子设备之间可以保持一定的距离。例如,对室内音箱而言,如果距离音箱 3～5m,通过语音输入也可以实现对音箱的控制;在驾驶的时候,在用手控制方向盘的同时,如果要去操作其他设备,如要调用导航系统,就可以采用语音的方式,这样既实现了控制,还可以保障交通安全。

## 5.2 语音识别流程

本节将从盲源分离、语音识别、语音特征提取和模板匹配等方面展开,声音采集编码后,要针对声音进行识别,后续再转换成文字并应用于各领域。

**1. 盲源分离**

在一个酒会中,会有很多人同时在说话,如果采用音箱放大,音箱中会有多个人的声音,还会夹杂嘈杂的环境音。如何将其中每个人的声音识别出来,分离出每个人的语音信号是进行后续处理的基础,分离每个人的语音信号要求每个人的语音信号有独立分布特性,盲源分离的目的是指在不知道源信号和其传输通道参数的情况下,根据输入源信号的统计特征,仅通过观测信号恢复出信号各个独立成分的过程。

每个人说话的信号用 $\boldsymbol{S}_i$ 表示,之后发出的信号会混合起来,所以需要学习混合信号的

特色，然后针对观测到的信号，将每个人的信号分离出来。中间需要用到混合矩阵和去混合矩阵，常见的分离需要用到随机方法，另一种分离只需要用到声音本身的特性，这里简单介绍只用声音本身的特性进行盲源分离的过程。混合矩阵用 $\boldsymbol{a}_i$ 表示，其中，$\boldsymbol{S}_i$ 是原信号，混合后的信号用 $\boldsymbol{x}_i$ 表示。

在盲源分离中用到了独立成分分析（Independent Component Analysis，ICA），ICA 假设 $\boldsymbol{S}_i$ 在统计学上是独立的，同时独立分量必须满足非高斯分布。在基本模型中，一般是不假定这些分布已知的（如果这些是已知，那问题就更加简化了）。

图 5-1(a)所示的原始信号包含一个正弦波和一个三角锯齿波，混合之后变成了图 5-1(b)所示的信号，进过盲源分离后结果如图 5-1(c)所示。在给定空间的基础上，在没有任何原始图信息的情况下，成功实现了分离，分离出的信号与原始信号基本一致。

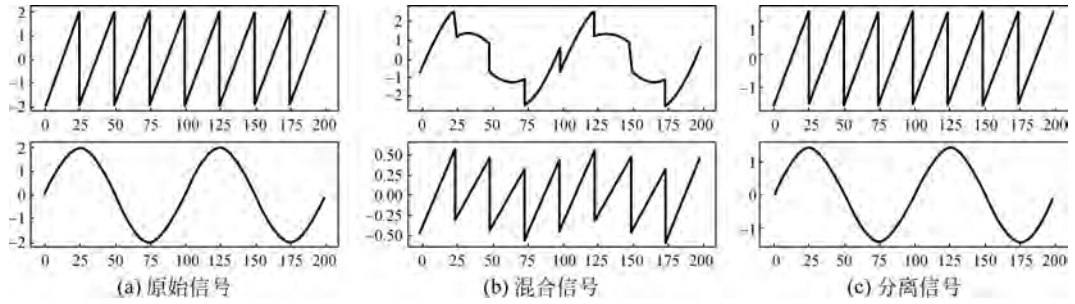



图 5-1　盲源分离

**2. 语音识别**

利用盲源分离把每个人的声音分离出来后，对每个人的声音进行识别，语音识别是将语音识别成相应的文本分析。语音识别流程中，首先需要进行训练，将语音中的一些规律存储起来作为模板，再进行识别，用这个模板进行语音处理。在模板匹配中，需要学习标准模板，最后用标准模板进行匹配，在识别角色中需要用到专家知识。

人体的语音是由人体的发音器官在大脑的控制下做身体运动产生的。人体发音器官由三部分组成：肺、气管和声道，这是语音产生的能源所在。气管连接着肺和喉，是肺与声道的联系通道。喉是由一个软骨和肌肉组成的复杂系统，其中包含着重要的发音器官。声带会产生语音，提供主要的声道是指声门至嘴唇的所有发音器官，包括咽喉、口腔和鼻腔。

语音是声音的一种，是由人的发声器官发出的具有一定语法和意义的声音。大脑对发音器官发出运动神经指令，控制发音器官，各种肌肉运动振动空气，空气由肺进入喉部，经过声带激励进入声道，最后通过嘴唇辐射形成语音。发浊音时，声带的不断开启和关闭将产生间歇的脉冲波，发清音时可等效成随机白噪声。

针对语音形成的规律，可以从几方面进行处理。首先进行预处理，预处理部分包括反混叠滤波、语音增强、去除声门激励和口唇辐射的影响，预处理最重要的步骤是端点检测和特征提取。特征提取的作用是从语音信号中提取一组或几组能够描述语音信号的特征参数，如平均能量过滤、倒谱信息预测系数等。训练和识别参数的选择直接关系着语音识别率的高低，训练是建立模式库的必备过程，其中一个词对应一个参考模式，它由一个词的多种发音训练得到。

模式匹配是整个系统的核心，其作用是按照一定的准则求取代测语音特征信号，参数和

语音信号与模式库中相应模板之间的失真率，最匹配的就是识别结果。预处理中间包含预加重处理，目的是消除低频干扰，尤其在50Hz工作频率可以得到对语音识别更为有用的高频谱部分，使信号变得平坦，以便于频率分析。预加重处理采用高通滤波器实现，是针对声音的发音模型进行的。

语音信号常常假定为短时平稳的，即在10～20ms的小间隔中某些物理参数可以近似看作不变，这样就可以采用图5-2所示的平稳过程分析处理方法进行处理。这种处理的基本方法是将语音信号分割为一些短帧再加以处理，为了减少帧与帧之间的变化，相邻帧之间必须要重叠，第$k$帧和$k+1$帧中间有一部分重叠，第$k+1$帧和$k+2$帧中间也有一部分重合。

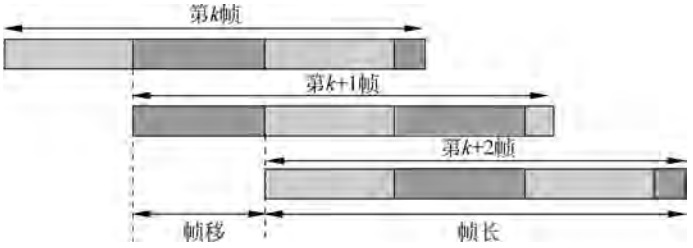



图5-2　平稳过程分析处理

为了减少语音帧的阶段效应，需要进行加窗处理，每帧信号与一个平滑的窗函数相乘，让帧两端平滑地衰减，可以防止傅里叶变换后旁瓣的强度取得更高质量的频谱。加窗处理一般用汉明窗，每一帧成立一个矩形窗，增加其连续性。汉明窗的主瓣最宽，旁瓣最低，可以有效克服泄漏现象，具有更频繁的低通特性，应用也比较广泛。

**3. 语音特征提取**

端点检测的目的是从连续的声音中检测出每一段语音的起始点和终止点，从而达到节省系统资源、方便实时分析的效果，如图5-3所示。单点检测的好坏可以直接影响孤立词识别率的高低，特征选择的好坏也直接影响语音识别的精度。语音特征要包含短时平均能量、短时过零率、线性预测系数(Linear Prediction Coefficient，LPC)、线性预测倒谱系数(Linear Prediction Cepstral Coefficients)和梅尔频率倒谱系数(Mel Frequency Cepstral Coefficents，MFCC)等。

短时平均能量反映了语音振幅或能力随时间缓慢变化的规律。在语音中可以区别出浊音来，因为浊音的短时平均能量为：

$$E_m = \sum_{n+m}^{N+m-1} S_w^2(n-m) \tag{5-1}$$

比轻音短时平均能量

$$Z_0 = \frac{1}{2}\sum_{n=0}^{N-1} \left| S_g n[S_w(n)] - S_g n[S_w(n-1)] \right| \tag{5-2}$$

大很多。

短时过零率表示一帧语音中信号穿过横轴的次数在离散时间信号情况下，当相邻两次初样有不同的代数符号就表示发生了过零。应用短时平均过零率可以得到谱特性的粗略估计。浊音中能量集中于较低频补中有较低的过零率，而轻音使能量集中于较高的频率制度，

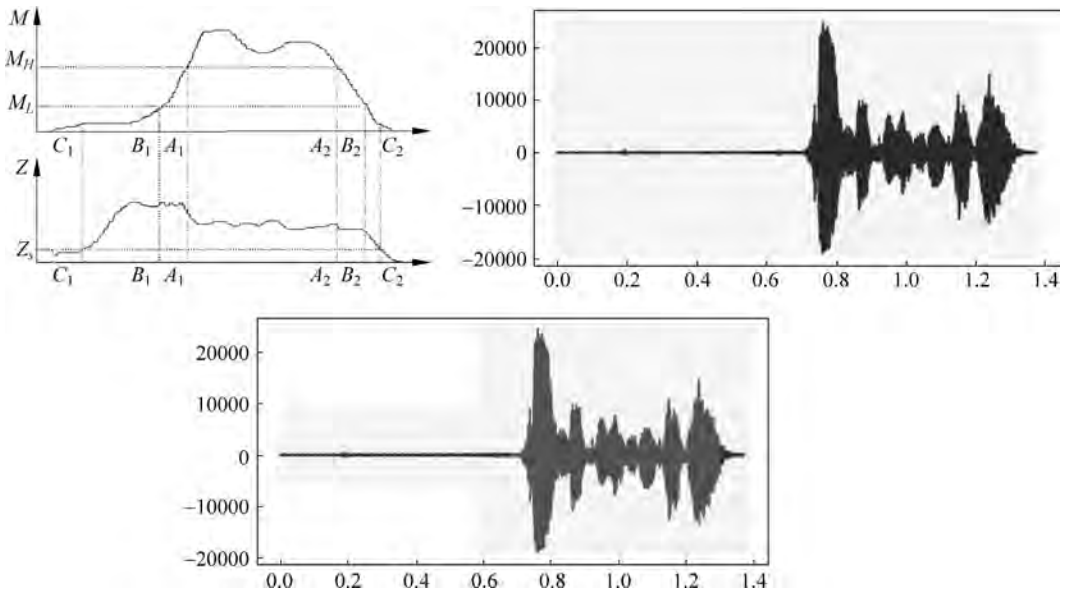



图 5-3 端点检测效果图

有较高的过零率，过零率可以用来区分轻音和浊音，在端点检测中有一定的运用。

语音的线性预测基本思想是语音信号的每个取样值可以用它过去的若干个取样值的线性组合表示，各加权系数的确定原则是使预测误差的均方差最小，如果利用过去 $p$ 个取样进行预测则称为线性预测。根据声音发音的规律看，语音信号可以看作一个线性时变系统，在准周期脉冲训练激励下产生的输出则是一个非时变系统。时间训练的 Z 变换的模的对数，即逆 Z 变换，可以看作信号的倒谱。倒谱分析的基础是假设语音是激励函数与声道冲击响应的卷积，语音的倒谱实际上是将语音的频谱取对数，再进行傅里叶变换：

$$H(z)=\frac{1}{1-\sum_{i=1}^{p}a_i z^{-1}} \tag{5-3}$$

$$c(n)=z^{-1}[\ln|z(x(n))|],\quad c(n)=z^{-1}[\ln H(Z)] \tag{5-4}$$

倒谱(cepstrum)是从频谱引出的新词，倒谱就是信号的傅里叶变频谱经对数运算后再进行傅里叶反变换得到的谱。线性预测系数主要是模拟人的发声模型，未考虑人耳对元音有较好的描述能力但对辅音的描述及抗造性能较差的听觉特性。但线性预测计算量小，易实现。

MFCC 是目前大多数语音识别系统中广泛使用的特征参数，它更符合人耳的听觉特性，因为没有标度，描述人耳频率的非线性特性没有频率尺度，它的值大体上对应实际频率的对数分布关系，MFCC 考虑了人的听觉特性，先将线性频谱映射到基于听觉感知的梅尔非线性频谱(Mel non-linear spectrogram)中，然后转换到倒谱上：

$$X_a(k)=\sum_{n=0}^{N-1}x(n)\mathrm{e}^{-\mathrm{j}2\pi k/N},\quad 0\leqslant k\leqslant N \tag{5-5}$$

$$H_m(k)=\begin{cases}0, & k<f(m-1)\\ \dfrac{2[k-f(m-1)]}{[f(m+1)-f(m-1)][f(m)-f(m-1)]}, & f(m-1)\leqslant k\leqslant f(m)\\ \dfrac{2[f(m+1)-k]}{[f(m+1)-f(m-1)][f(m)-f(m-1)]}, & f(m)\leqslant k\leqslant f(m+1)\\ 0, & k\geqslant f(m+1)\end{cases} \tag{5-6}$$

其中，$x(n)$代表语音信号；$N$ 代表傅里叶变换的点数；式(5-6)用到的三角滤波器的频率响应定义如图 5-4 所示。

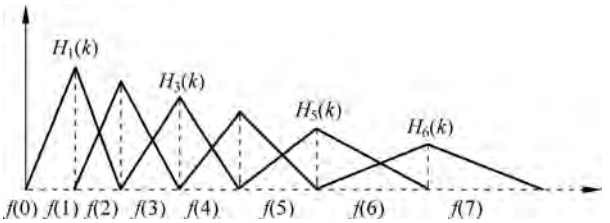



图 5-4　三角滤波器的频率响应

MFCC 的计算流程可以分为预加重、分帧、加窗、快速傅里叶变换、MEL 滤波器组、计算对数能量、经离散余弦变换、对数能量、动态差分参数的提取等步骤。

MFCC 突出的优点是在噪声中表现出更强的健壮性，在非特定人语音识别方面减少了因说话人不同带来的影响，不足之处是算法复杂度比较大。由于信号在时域上的变化很难看出其特性，所以通常将它转换为频域分析，其中不同的能量分布就代表不同的语音特性，所以在乘以汉明窗后，还必须对分帧加窗后的各帧信号进行快速傅里叶变换，得到各帧的频谱并对语音信号的频谱取模平方，得到语音信号的功率谱。经过傅里叶变换后的频谱包括了声音的很多频率信息，是处理频率信息的一个很重要的工具。

**4. 模板匹配**

有了特征就需要进行特征匹配，也就是模板匹配方法的应用，在语音识别中，常用的模板匹配方法有动态时间规整(Dynamic Time Warping，DTW)、HMM、向量量化(Vector Quantization，VQ)技术、人工神经网络(Artificial Neural Network，ANN)等。

语音信号具有很强的随机性，不同的发音习惯、不同的发音环境或不同的心情都会使一个字的发音持续时间发生变化，所以参考模板和测试模板中的向量长度不同，且其中每个发音单元的长度也不同。DTW 是一种最优算法，算法的思想就是把未知量均匀拉长或缩短，直到参考模式长度一致。在这一过程中，未知语音识别信号的时间轴进行不均匀扭曲使其模板特征对齐，并在两者之间不断地进行两个向量距离最小的路径匹配计算，从而获得两个向量匹配时累积距离最小的规整函数。

DTW 是较早提出解决发音长短不匹配问题的，测试语音参数共有 $i$ 帧向量，而参数模型共有 $j$ 帧，$i$ 和 $j$ 不相等，这是使用 DTW 的基本条件。这时需要寻找一个时间规整函数 $J=w(i)$，它将测试向量的时间轴 $i$ 非线性地映射到模板的时间轴上，比如发音“hello”或者“he～llo”这两个的长度是不一样的，那么把前面的“ha-”跟后面“ha-”的对应上，把“-llo”跟“-llo”对应上，就是 DTW 需要解决的问题。定义两个时间序列 $Q$ 和 $C$，长度分别为 $n$ 和 $m$，

对这两个序列构造一个 $n \times m$ 的矩阵网络，$q_i$ 和 $c_j$ 描述两个点的距离。最后用路径寻找法找到一条最优的路径，这条路径不是随意选择的，需要满足以下几个约束。

(1) 边界条件任意语音发音快慢都可能变化但其先后顺序不会变，因此首选路径必定是左下角出发，右上角结束。

(2) 连续性不能跨过某个点去匹配，只能和自己相邻的点对齐，这样可以保证序列 $Q$ 和 $C$ 中的每个坐标都在 $w$ 出现单价性限制，$w$ 必须是随着时间单调进行的连续性和单调性的约束。每一格的路径就只能是三个方向，即 $(i+1,j)$、$(i,j+1)$、$(i+1,j+1)$ 满足这些约束条件的路径可以有指数个，取其最小的路径：

$$\mathrm{DTW}=(Q,C)=\min\left\{\frac{\sum_{k=1}^{k} w_k}{K}\right\} \tag{5-7}$$

这就是相应的优化函数。通过该优化函数最终得到两个时间序列 $Q$ 和 $C$ 的匹配过程，完成语音的配准和识别。

语音作为观测信号，文字作为隐性信号，隐性信号是指外界不便观测或者观测不到的状态信号。文字和信号之间存在一个状态迁移的概率矩阵，描述由文字到语音信号转换时文字的发音概率、观测概率、字词之间的转换频率。最后根据具体的语音信号计算出相应的文字训练，这是 HMM 的主要思路。HMM 库不是预先存好的模式样本，而是通过反复的训练用迭代算法形成的与训练输出信号吻合概率最佳的 HMM 模型参数。

虽然 HMM 方法在训练过程中的处理比 DTW 方法要复杂，但识别过程比 DTW 方法简单。在孤立词和小词汇的汉语识别中，HMM 的识别率要高于 DTW 方法，而且解决了 DTW 无法实现的连续语音识别的应用问题。因此，在汉语语音识别中 HMM 方法不仅可以用于孤立词识别系统，而且在连续语音识别和说话人识别等方面也得到了广泛的应用，是目前汉语语音识别技术的主流。

VQ 技术用一个 $K$ 维向量表示一个原类，用 $K$ 个标量表征语音信号的波形帧或者参数帧，然后对向量进行整理量化。VQ 技术需要将无限的质量空间划分为 $M$ 个有限的区域边界，而每个区域有一个中心向量值及码字，所以总共有 $M$ 个码字。各码字的下标或者序号的集合，则构成了一本反应训练时 $K$ 维向量的码书。在语音识别时，将 $K$ 作为待处理量，与已有的码书中的 $M$ 个区域边界进行比较，即可得到识别结果。

ANN 是用于模拟人的组织结构和思维过程的一个前沿研究领域，基于 ANN 的语音识别系统通常由神经元、训练算法和网络结构等构成。ANN 采用非线性信息处理机制、信息分布和存储机制等多种现代信息技术工具，具有高速的信息处理能力，并有较强的适应和自动调节能力。在训练过程中，ANN 能不断调整自身参数权重和拓扑结构，以适应环境和系统性能优化的需求，在模式识别中有选择快速、识别率高等特点。深度学习是神经网络的进一步延伸，可以通过端到端的方式进行，也就是通过学习特征自动完成多种任务，可以通过学习讲话人的特征识别讲话人，也可以通过学习语音的特征进行语音识别。

## 5.3 语音识别案例实现

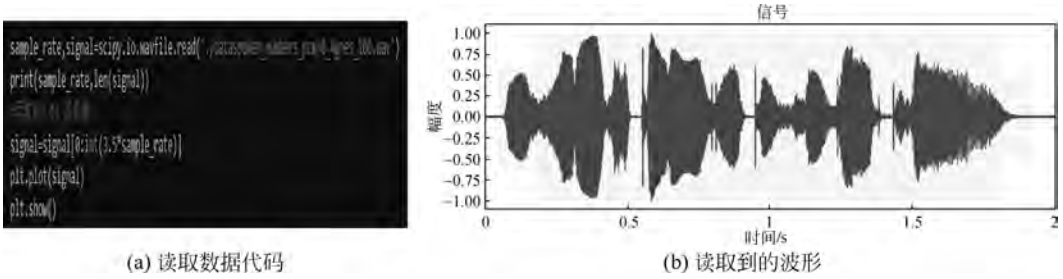
本节基于 Amazon Ehco 介绍语音识别的案例实现，首先进行特征识别然后用模板匹配

方法进行语音的识别，主要涉及 MFCC 和神经网络模板匹配的代码实现，并给出了完整的硬件设计。

### 5.3.1 MFCC

Amazon Ehco 是一个典型的语音音箱，其 MFCC 实现的具体步骤如下所述。

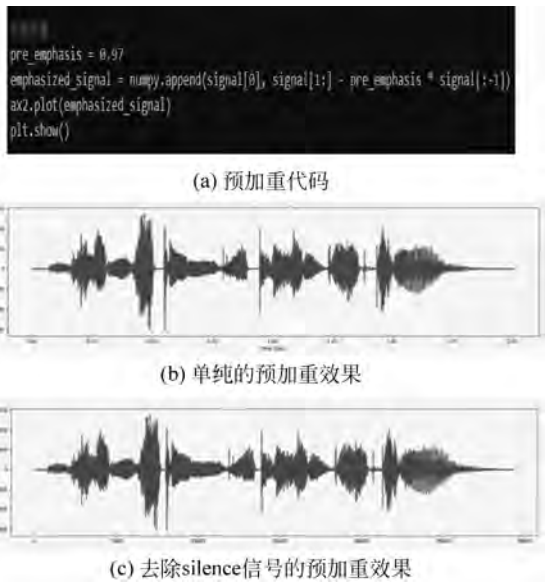
(1) 读取数据和展示数据，如图 5-5 所示。音频文件格式有很多种，如 MP3、WAV 等，图 5-5(a)所示的代码中读取的是 WAV 格式的文件。




(a) 读取数据代码　　(b) 读取到的波形

图 5-5　读取数据

(2) 读取了部分数据后，通过图 5-5(b)所示的波形，可以发现波形是毫无规律的。对于没有规律的波形，可以采取预加重的方法进行处理，具体代码如图 5-6(a)所示。此处提供了两种预加重的方法，一种是单纯的预加重方法，处理后的波形如图 5-6(b)所示；另一种是去除 silence 信号后的预加重，对于同样的输入波形，采用此方法的波形图发生了一些变化，如图 5-6(c)所示。




(a) 预加重代码

(b) 单纯的预加重效果

(c) 去除silence信号的预加重效果

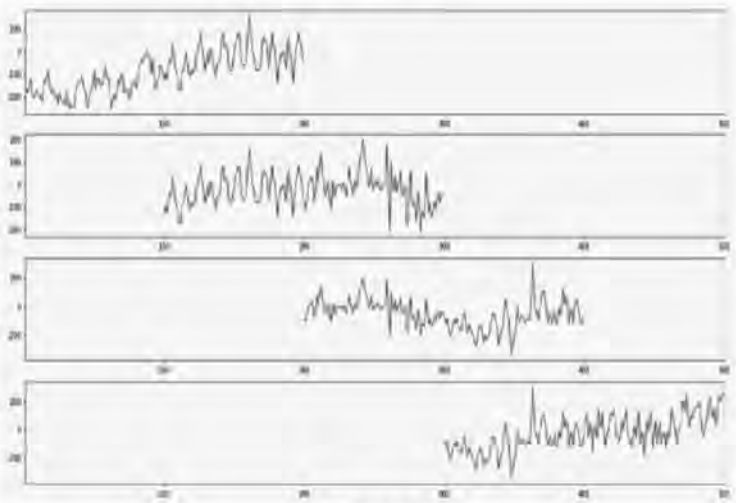
图 5-6　对数据进行预加重处理

(3) 语音处理范围里的典型帧(即分帧)大小是 20～40ms，那么一个长时信号的连续帧之间会出现重叠，相对应的分帧有一部分会与前一帧或后一帧有重叠。为了消除重叠，对长

时信号进行分帧处理，分解为有重叠的短时信号，供后面做进一步处理，具体实现代码如图 5-7(a)所示，信号分帧的效果如图 5-7(b)所示。

```
frame_size=0.025
frame_stride=0.01
frame_length,frame_step=frame_size*sample_rate,frame_stride*sample_rate
signal_length=len(emphasized_signal)
frame_length=int(round(frame_length))
frame_step=int(round(frame_step))
num_frames=int(numpy.ceil(float(numpy.abs(signal_length-frame_length))/frame_step))

pad_signal_length=num_frames*frame_step+frame_length
z=numpy.zeros((pad_signal_length-signal_length))
pad_signal=numpy.append(emphasized_signal,z)

indices = numpy.tile(numpy.arange(0, frame_length),
                     (num_frames, 1)) + numpy.tile(numpy.arange(0, num_frames * frame_step, frame_step),
                      (frame_length, 1)).T

frames = pad_signal[numpy.mat(indices).astype(numpy.int32, copy=False)]

ax3=f1.add_subplot(2,2,3)
ax3.plot(pad_signal)
plt.show()
```

(a) 代码实现



(b) 分帧效果

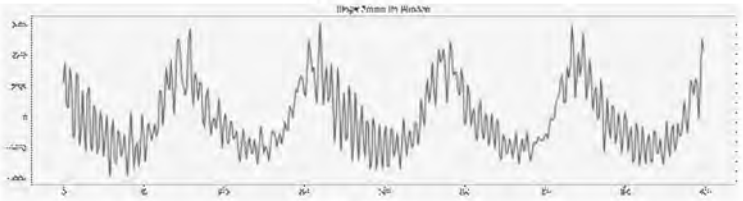
图 5-7　信号分帧

(4) 把一段长时信号分帧得到短时信号后，还需要对分帧进行加窗处理，通常采用的是汉明(Hamming)窗，实现代码如图 5-8(a)所示。对加窗后的每一帧做短时傅里叶变换(Short-Time Fourier Transform，STFT)，变换前后的信号分别如图 5-8(b)和图 5-8(c)所示。变换后把每一帧的结果沿另一个维度堆叠起来，得到类似于一幅图的二维信号形式，输入的声音信号通过 STFT 展开得到的二维信号就是所谓的声谱图，如图 5-8(d)所示。

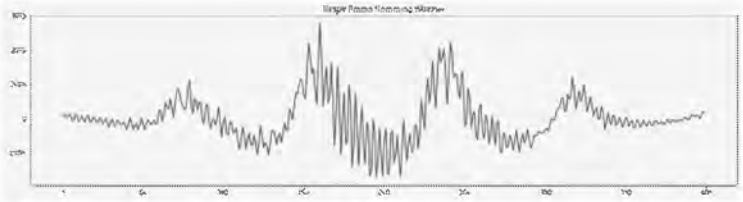
(5) 通过 STFT 得到的声谱图中的颜色就表示了幅值，通过滤波器组对颜色进行分析，就可以得到声音中频谱的信息，代码实现如图 5-9(a)所示。使用三角滤波器提取的功率谱如图 5-9(b)所示。

```
'''
    加上汉明窗
    frames *= 0.54 - 0.46 * numpy.cos((2 * numpy.pi * n) / (frame_length - 1))
'''
frames *= numpy.hamming(frame_length)

#傅立叶变换和功率谱
NFFT = 512
mag_frames = numpy.absolute(numpy.fft.rfft(frames, NFFT))  # 幅度
pow_frames = ((1.0 / NFFT) * ((mag_frames) ** 2))  # 功率谱
```

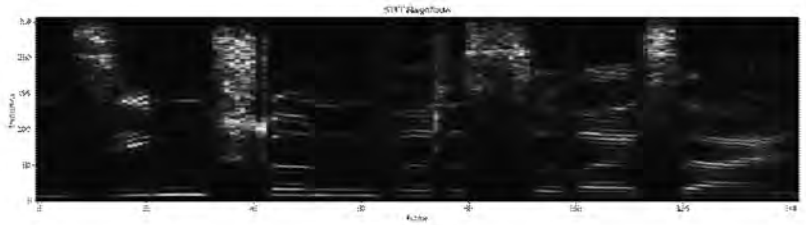
(a) 代码实现



(b) 未加窗



(c) 加窗



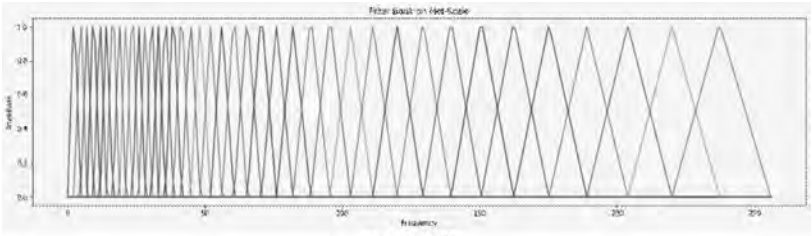
(d) 声谱图

图 5-8　加窗及 STFT 处理

```
for m in range(1, nfilt + 1):
    f_m_minus = int(bin[m - 1])   # left
    f_m = int(bin[m])             # center
    f_m_plus = int(bin[m + 1])    # right
    for k in range(f_m_minus, f_m):
        fbank[m - 1, k] = (k - bin[m - 1]) / (bin[m] - bin[m - 1])
    for k in range(f_m, f_m_plus):
        fbank[m - 1, k] = (bin[m + 1] - k) / (bin[m + 1] - bin[m])
filter_banks = numpy.dot(pow_frames, fbank.T)
filter_banks = numpy.where(filter_banks == 0, numpy.finfo(float).eps, filter_banks)
filter_banks = 20 * numpy.log10(filter_banks)
```

(a) 代码实现

图 5-9　滤波器代码及功率谱



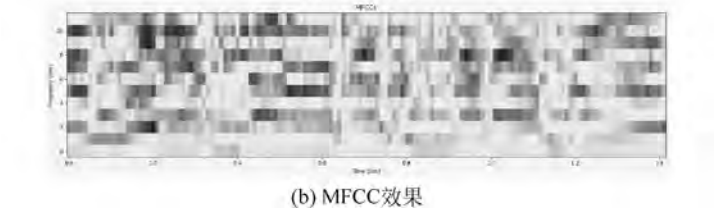
(b) 功率谱

图 5-9 （续）

（6）对功率谱做平均标准化，从所有帧中间减去每个系数的平均值，最后就可以得到相应的 MFCC 结果，代码实现如图 5-10(a)所示，MFCC 效果如图 5-10(b)所示。

```
num_ceps = 12
mfcc = dct(filter_banks, type=2, axis=1, norm='ortho')[:, 1 : (num_ceps + 1)]
(nframes, ncoeff) = mfcc.shape

n = numpy.arange(ncoeff)
cep_lifter =22
lift = 1 + (cep_lifter / 2) * numpy.sin(numpy.pi * n / cep_lifter)
mfcc *= lift

#filter_banks -= (numpy.mean(filter_banks, axis=0) + 1e-8)
mfcc -= (numpy.mean(mfcc, axis=0) + 1e-8)

print(mfcc.shape)
plt.plot(filter_banks)

plt.show()
```

(a) 代码实现



(b) MFCC效果

图 5-10 MFCC

利用实现的 MFCC 就可以进行后续处理，如识别说话人等。

### 5.3.2 神经网络模板匹配

基于 5.3.1 节提取的 MFCC 特征，可以利用神经网络对其进行分类。以下通过一个简单的实例进行说明，对简单的 0～9 进行分类，识别语音中的数字，具体实现步骤如下。

（1）载入文件，并把文件分为训练集和测试集，如图 5-11 所示。

（2）把训练集或者测试集的文件分为语音数据和标签，把标签转换成 One-Hot 格式，One-Hot 格式中每一位向量对应着 0～9 的一个数。例如，测试后，0～9 中数字“5”的概率比较高，那么就可以认为数值是“5”，图 5-12 是具体的代码实现。

（3）使用训练集进行训练，利用测试集进行验证。在神经网络中，优化函数使用了 Adam()函数，损失函数使用了交叉熵函数，得到训练集的特征，最后获取的标签如图 5-13 所示。


图 5-11　载入文件

```
train_set, test_set = load_files()
train_features, train_labels = read_files(train_set)
test_features, test_labels = read_files(test_set)

model = keras.models.Sequential()
model.add(Flatten())
model.add(Dense(512,input_dim=1600))
model.add(Activation('relu'))
model.add(Dense(10,input_dim=512))
model.add(Activation('softmax'))

tensorboadr = TensorBoard()

model.compile(optimizer='Adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(train_features,train_labels,epochs=3,
          validation_data=(test_features,test_labels),
          validation_freq=1,batch_size=1,callbacks=[tensorboadr])
```

图 5-12　训练集代码

```
192/192 [==============================] - 1s 6ms/sample - loss: 0.0819 -
accuracy: 0.9948 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 2/3
192/192 [==============================] - 1s 3ms/sample - loss: 0.0000e+00 -
accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 3/3
192/192 [==============================] - 1s 3ms/sample - loss: 0.0000e+00 -
accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
```

图 5-13　获取标签

（4）最后是进行训练，把提取好的特征送到神经网络进行训练和分类，并用测试集进行测试。经过训练后，损失值已经很小，则说明准确率比较高。如图 5-14 所示是每个 Epoch 的损失值的变化情况。如果说特征具备比较充分，则训练就比较容易，测试也比较准确。

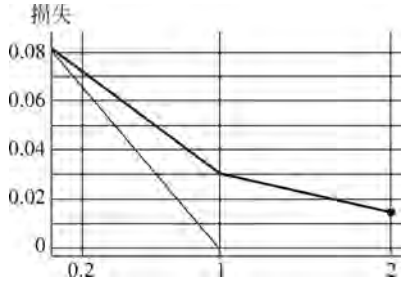



图 5-14　损失值的变化

### 5.3.3 Amazon Echo

Amazon Echo 是 Amazon 推出的一款智能控制设备，而 Alexa 是预装在 Amazon Echo 内的个人虚拟助手，可以接收相应的语音命令。Amazon Echo 的硬件实现如图 5-15 所示。

Amazon Echo 的硬件实现比较复杂，在这样一个小的圆盘中包含了 7 个定向麦克风。在播放音乐的时候，也始终可以听到用户的声音。Amazon Echo 可以用 Alexa 实现唤醒，Alexa 是一种非常复杂的信息处理层的人机交互界面，将人的声音流转换为文本。在每一次交互的过程中，Alexa 都在进行训练，以便可以更好地聆听人的声音，然后更准确地触发并映射用户命令的动作。

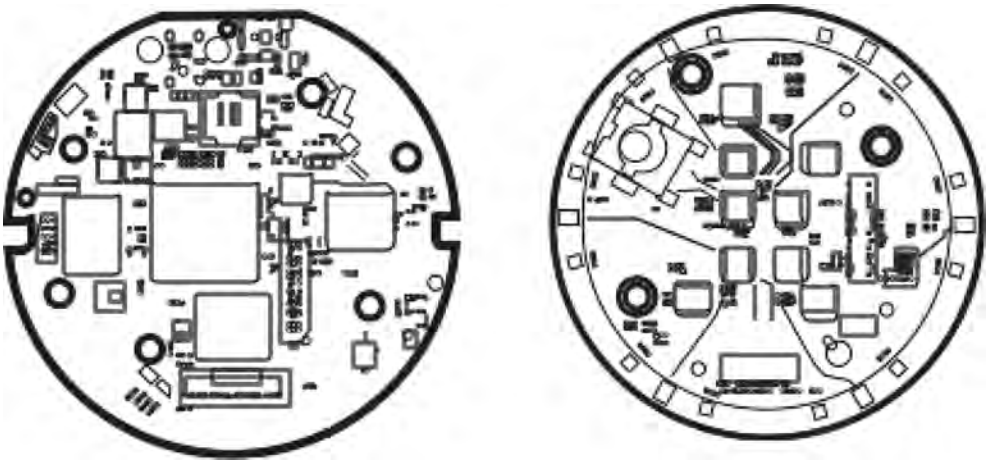


图 5-15 Amazon Echo 示意图