基于时域分析技术的语音识别

摘要：

语言是人类最重要的交流工具，自动语音识别技术起源于 20 世纪 50 年代，最早的商用系统是 IBM 在 90 年代推出的 ViaVoice。经过半个多世纪的发展，语音识别技术目前已日趋成熟并成功应用到人们的日常生活之中，如苹果手机的 Siri 体验、科大讯飞的迅速崛起等。

语音是一种典型的、易于获取的一维时序信号，语音信号处理及识别技术也是数字信号处理课程绝佳的实践途径。时间序列分析、快速傅里叶变换、滤波器设计等多项数字信号处理的教学内容在语音识别核心技术中均占有重要地位。本系列实验即面向语音识别基本任务，由浅入深，循序渐进地设计完善语音识别系统，包括时域法、频域法及说话人识别。

关键词：

语音识别 信号处理

1. 实验目的

熟悉语音数据的基本形式及特点，理解并应用离散时间信号的基本分析、处理方法，理解语音识别技术的概貌，为后续实验打好基础。

1.1 掌握语音信号采集，理解语音信号格式。

1.2 熟悉语音信号预处理方法。

1.3 了解语音信号分帧与加窗的重要性和必要性；掌握常用的窗函数和加窗分帧处理的原理；根据原理能编程实现分帧及加窗处理。

1.4 了解语音短时时域分析的原理；掌握短时时域分析的一些参数计算方法；根据原理能编程实现短时时域分析的参数计算。

1.5 了解基于双门限法的端点检测原理。

1.6 掌握并编程实现基于时域分析技术实现孤立字语音识别方法。

2. 实验原理

2.1 语音信号的采集

采集“0”、“1”、…、“9”这 10 个语音的 wav 文件，每个类别应采集 10 组以上的样本。

使用 MATLAB 的 audiorecord(fs,n,ch)函数录制，使用 wavwrite 函数进行保存。

2.2 语音信号格式的理解

wav 格式

偏移地址	大小字节	数据块类型	内容
00H~03H	4	4 字符	资源交换文件标志（RIFF）
04H~07H	4	长整数	从下个地址开始到文件尾的总字节数
08H~0BH	4	4 字符	WAV 文件标志（WAVE）
0CH~0FH	4	4 字符	波形格式标志（fmt ），最后一位空格。
10H~13H	4	整数	过滤字节（一般为 00000010H），若为 00000012H 则说明数据头携带附加信息（见“附加信息”）。

14H~15H	2	整数	格式种类（值为 1 时，表示数据为线性 PCM 编码）
16H~17H	2	整数	通道数，单声道为 1，双声道为 2
18H~1BH	4	长整数	采样频率
1CH~1FH	4	长整数	波形数据传输速率（每秒平均字节数）
20H~21H	2	整数	DATA 数据块长度，字节。
22H~23H	2	整数	PCM 位宽
随后 2 字节	2	整数	附加信息（可选，由上方过滤字节确定）
随后	…	不定长度字符	“fact”,该部分是可选部分，一般当 WAV 文件是由某些软件转换而来时，包含该部分。 若包含该部分： （1）该部分的前 4 字节为数据头，一般为 4 个字母。 （2）随后 4 个字节表示长度，即除去头（4 字节）和长度（4 字节）之后，数据本身的长度。 （3）最后的字节为数据本身。

			例如：“66 61 73 74 04 00 00 00F8 2F 14 00”。 “66 61 73 74”是 fact 字段的数据头， “04 00 00 00”是数据本身的长度，“F8 2F 14 00”是数据本身。 （注意是 little-endian 字节顺序）
随后 4 字节	4	4 字符	数据标志符（data）
随后 4 字节	4	长整型	DATA 总数据长度字节
随后	…		DATA 数据块

2.3 语音信号的预处理

对语音原始数据实现端点检测等基本的预处理任务，为后续的时域分析做好准备。端点检测的含义为将数据的实际发声部分从静音及背景噪声中分割出来，如图 1.3 所示。后续计算将仅针对分割出的部分进行。该部分可以采用交互式的手工方式来实现，也可编程自动化地实现。

2.4 短时时域特性分析

2.4.1 短时能量与短时平均幅度

设第 n 帧语音信号 () n xm 的短时能量用 n E 表示，则其计算公式如下：

$$E_n = \sum_{m=0}^{N-1} x_n^2(m)$$

E_n 是一个度量语音信号幅度值变化的函数，但它有一个缺陷，即它对高电平非常敏感 （因为它计算时用的是信号的平方)。为此，可采用另一个度量语音信号幅度值变化的函数， 即短时平均幅度函数 M_n ，它的定义为：

$$M_n = \sum_{m=0}^{N-1} |x(m)|$$

M_n 也是一帧语音信号能量大小的表征，它与 E_n 的区别在于计算时小取样值和大取样 值不会因取平方而造成较大差异，在某些应用领域中会带来一些好处。

2.4.2 短时过零率

短时过零率表示一帧语音中语音信号波形穿过横轴（零电平）的次数。对于连续语音信号，过零即意味着时域波形通过时间轴；而对于离散信号，如果相邻的取样值改变符号则称为过零。因此，过零率就是样本改变符号的次数。过零率实质上是信号频谱分布在时域的一种最简单的体现，即高频分量丰富的信号其过零率也一般较高。 设第 n 帧语音信号$X_n(m)$）的短时过零率用Z_n 表示，则其计算公式如下：

$$Z_n = \frac{1}{2}\sum_{m=0}^{N-1} |\mathrm{sgn}[x_n(m)] - \mathrm{sgn}[x_n(m-1)]|$$

式中，sgn[.]是符号函数，即

$$\mathrm{sgn}[x] = \begin{cases} 1, & (x \geq 0) \\ -1, & (x < 0) \end{cases}$$

其具体计算步骤如下：

a）首先对信号进行去直流化；

b）然后按照时间顺序统计采样点数值符号变号的次数；

c）将上述计数出的次数针对序列时长进行归一化操作，即得到过零率。

2.5 基于双门限法的端点检测

根据语音的统计特性，可以把语音段分为清音、浊音以及静音（包括背景噪声）三种。

语音端点检测本质上是根据语音和噪声的相同参数所表现出的不同特征来进行区分。在双门限法中，短时能量可以较好地区分出浊音和静音。对于清音，由于其能量较小，在短时能量检测中会因为低于能量门限而被误判为静音；短时过零率则可以从语音中区分出静音和清音。

将这两种检测结合起来，就可以检测出语音段（清音和浊音）及静音段。在基于短时能量和过零率的双门限端点检测算法中首先为短时能量和过零率分别确定两个门限，一个为较低的门限，对信号的变化比较敏感，另一个是较高的门限。当低门限被超过时，很有可能是由于很小的噪声所引起的，未必是语音的开始，到高门限被超过并且在接下来的时间段内一直超过低门限时，则意味着语音信号的开始。

3. 实验内容

3.1 语音信号的采集

使用 MATLAB 采集 0 到 9 共十个数字的 wav 格式语音，每个数字重复 10 遍。

3.2 语音信号的处理

以步长 900 帧处理语音信号，计算短时能量，并以短时能量为依据加窗分离出声音信号，加窗时使用双门限法，而后计算平均过零率。

4. 实验数据处理

4.1 计算短时能量

虽然语音信号具有时变特性，但是在一个短时间范围内（一般认为在10~30ms的时间内），其特性基本保持不变即相对稳定，因而可以将其看作是一个准稳态过程，即语音信号具有短时平稳性。所以任何语音信号的分析和处理必须建立在“短时”的基础上，即进行“短时分析”，将语音信号分为一段一段来分析其特征参数，其中每一段称为“一帧”，帧长一般取10~30ms。这样，对于整体的语音信号来讲，分析出的是由每一帧特征参数组成的特征参数时间序列。

信号的短时能量定义为：设语音波形时域信号为x(l)、加窗分帧处理后得到第n帧语音信号为$x_n(m)$，则$x_n(m)$满足下式：

$$x_n(m) = w(m)x(n+m)$$

$$w(m) = \begin{cases} 1, & 0 \le \mathrm{m} \le \mathrm{N}-1 \\ 0, & m = 其他值 \end{cases}$$

其中，n=0,1T,2T,……，其中N为帧长，T为帧移长度。

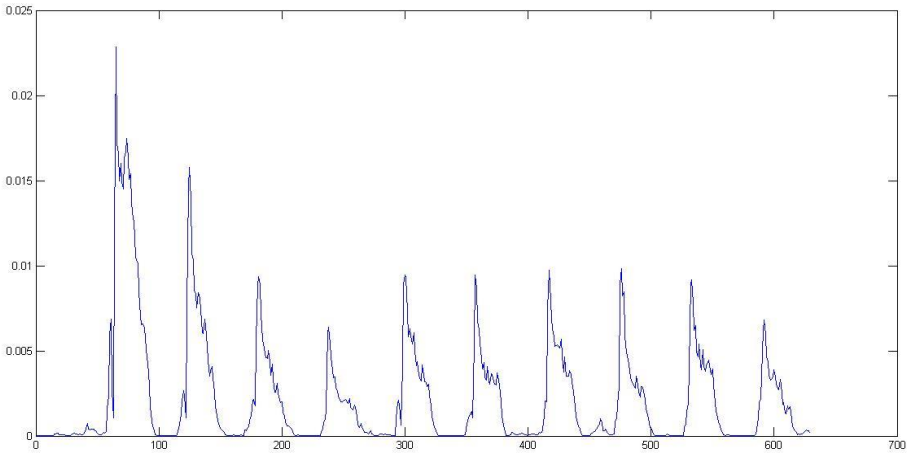
设第n帧语音信号$x_n(m)$的短时能量谱用E_n表示，则其计算公式如下

$$E_n = \sum_{m=0}^{N-1} x_n^2(m)$$

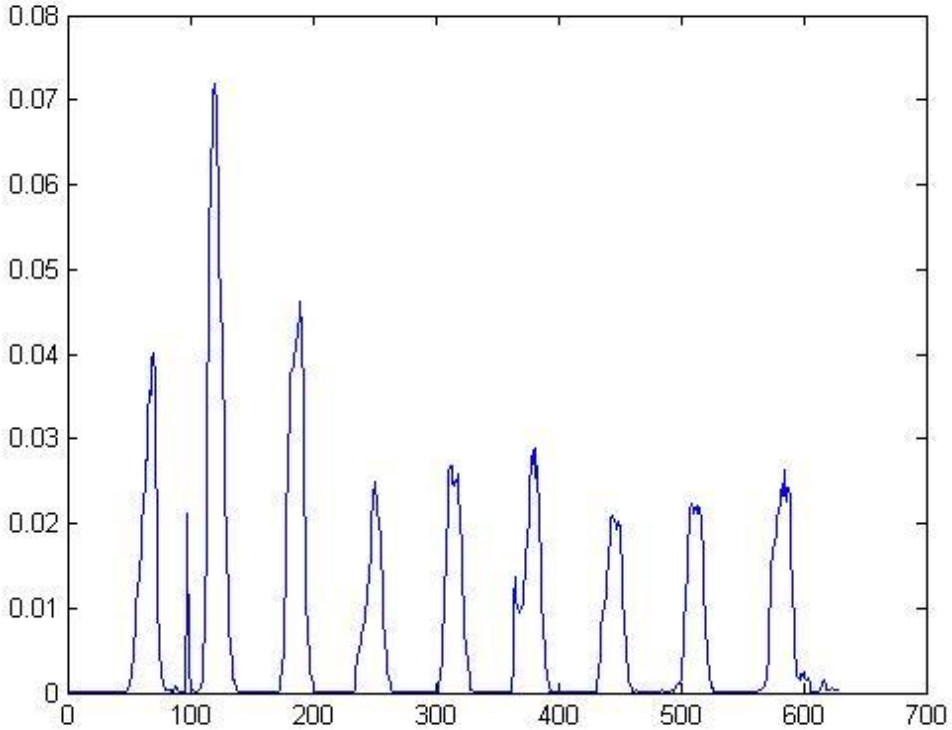
在本次实验中，我们将音频的采样率设置为44100Hz，帧长设为900，帧移为200，即1秒49帧，每帧约20.4ms。由音频数据的总长度（通过length函数得到）即可计算得语音的总帧数，再通过for循环对一帧内的音频信号进行短时平均能量计算（利用上述公式）。

4.2 每个数字的短时能量

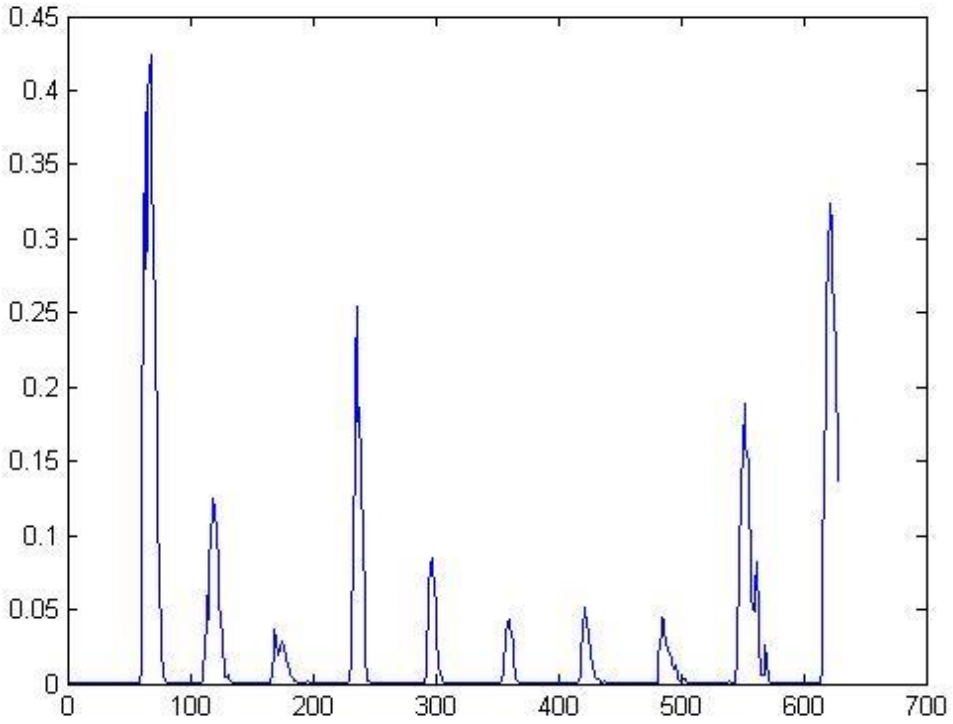
数字 0：



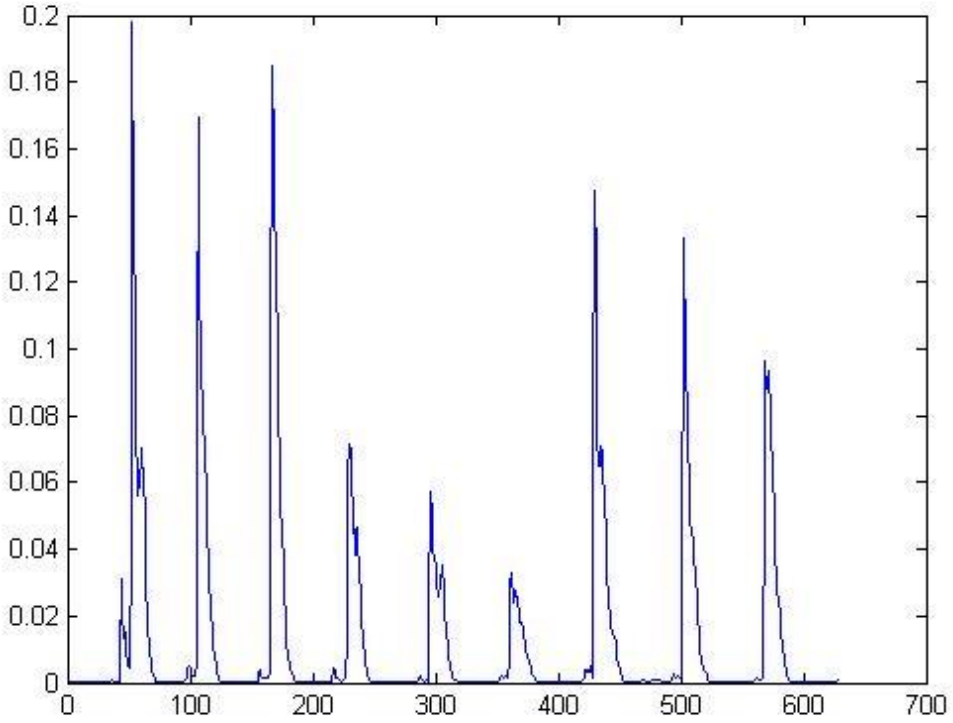
数字 1：



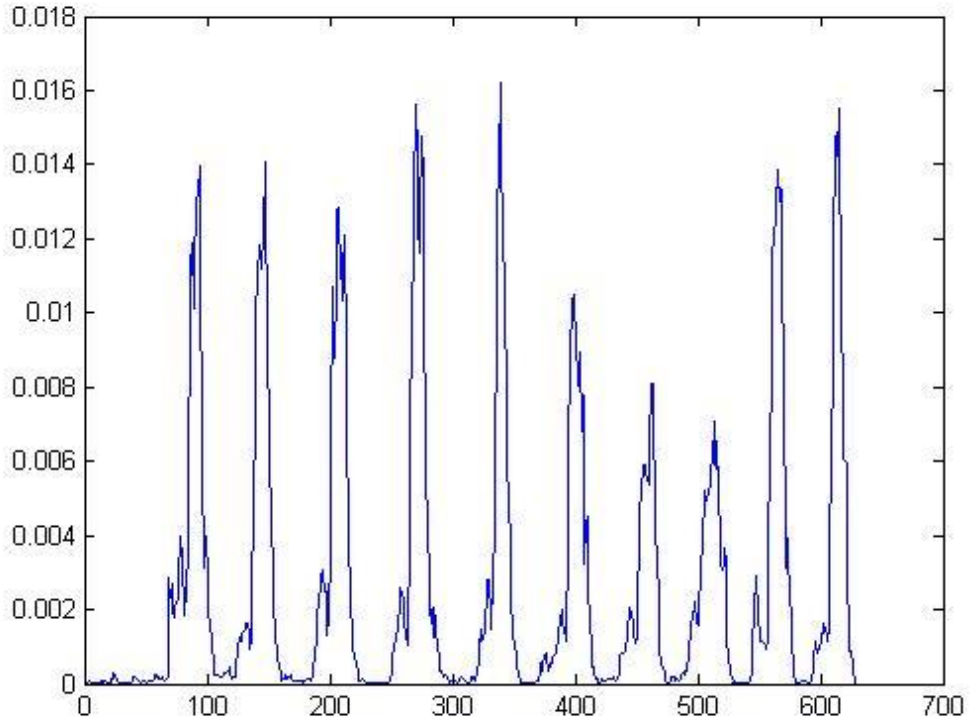
数字 2：


0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
100
200
300
400
500
600
700

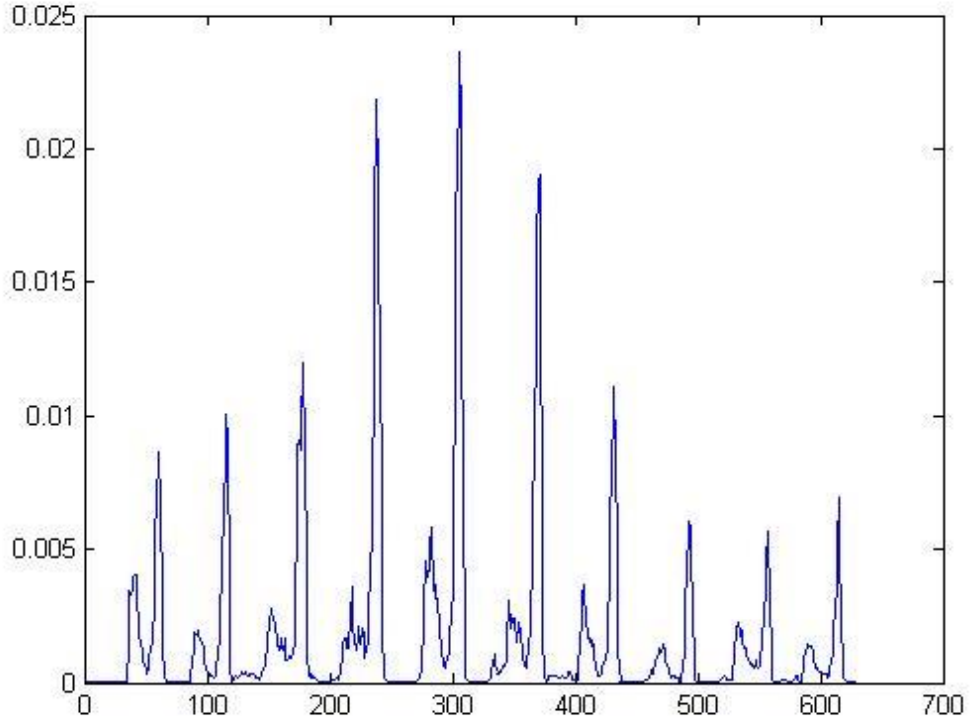
数字 3：


0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
100
200
300
400
500
600
700

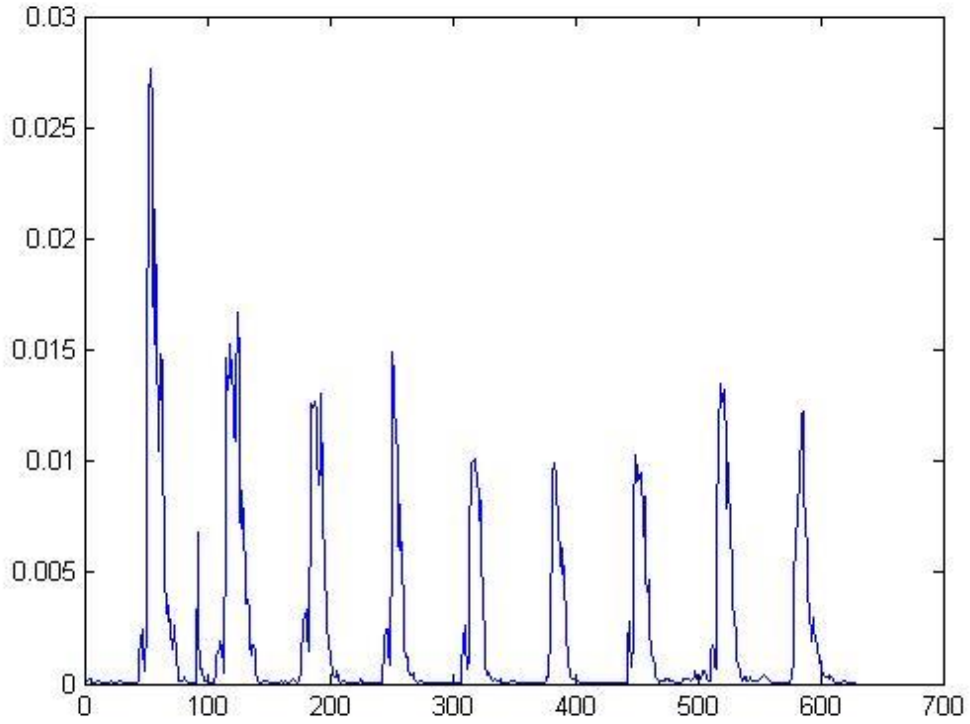
数字 4：


0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
100
200
300
400
500
600
700

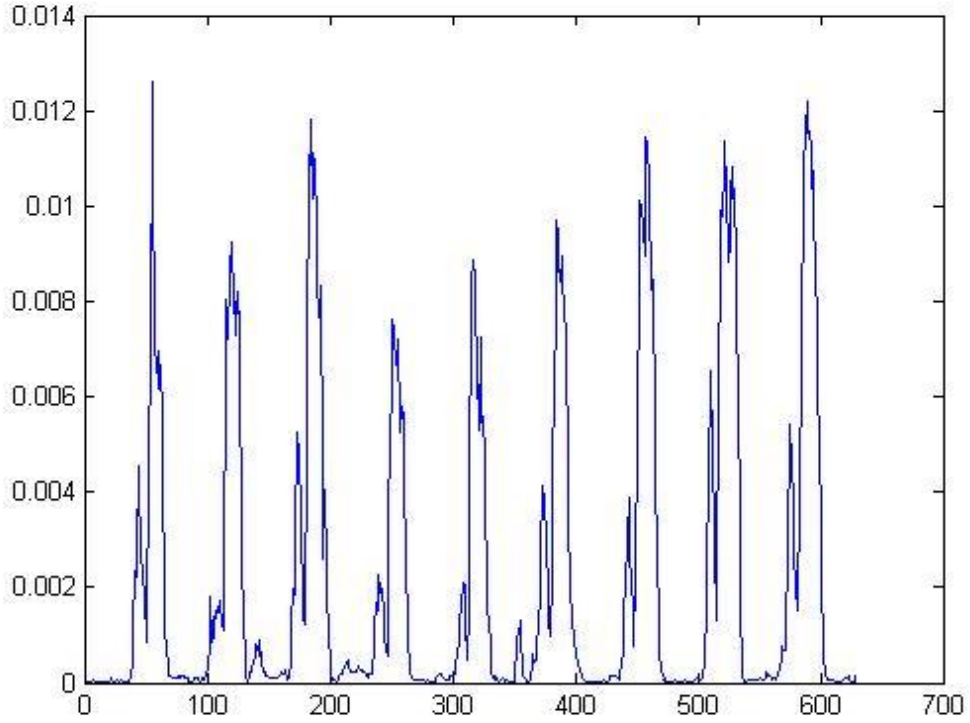
数字 5：


0.025
0.02
0.015
0.01
0.005
0
0
100
200
300
400
500
600
700

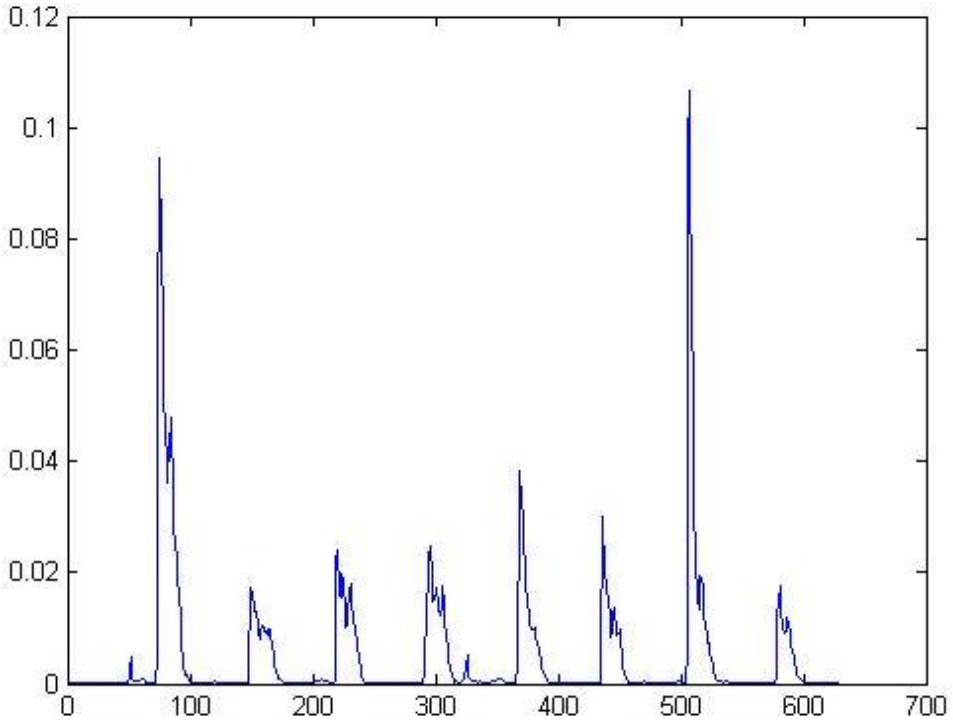
数字 6：


0.03
0.025
0.02
0.015
0.01
0.005
0
0
100
200
300
400
500
600
700

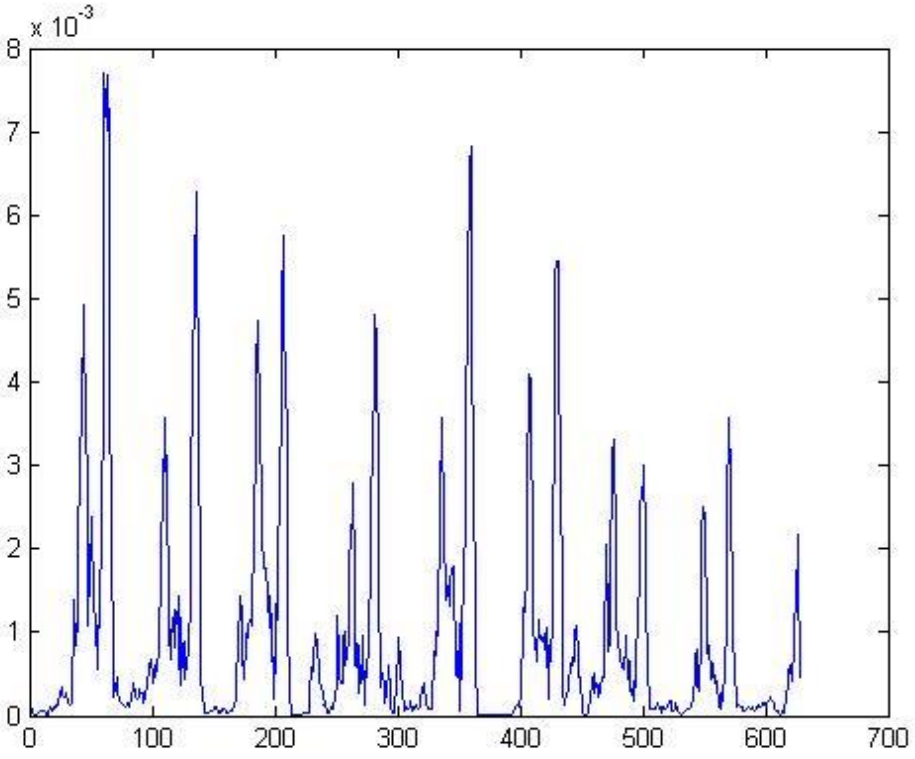
数字 7：


0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
100
200
300
400
500
600
700

数字 8：

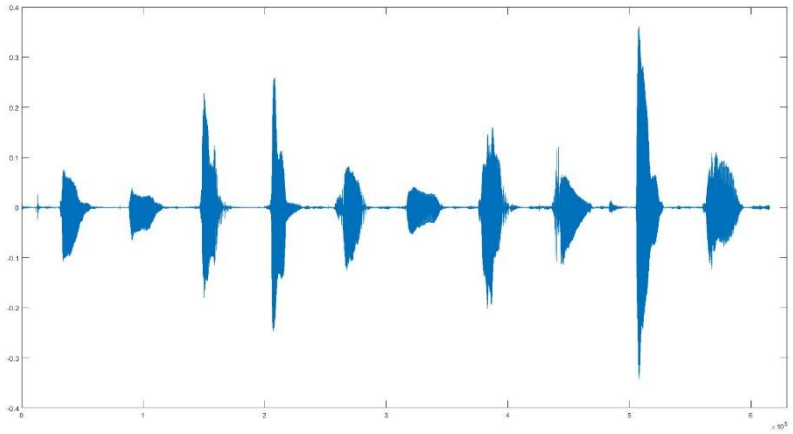

0.12
0.1
0.08
0.06
0.04
0.02
0
0
100
200
300
400
500
600
700

数字 9：

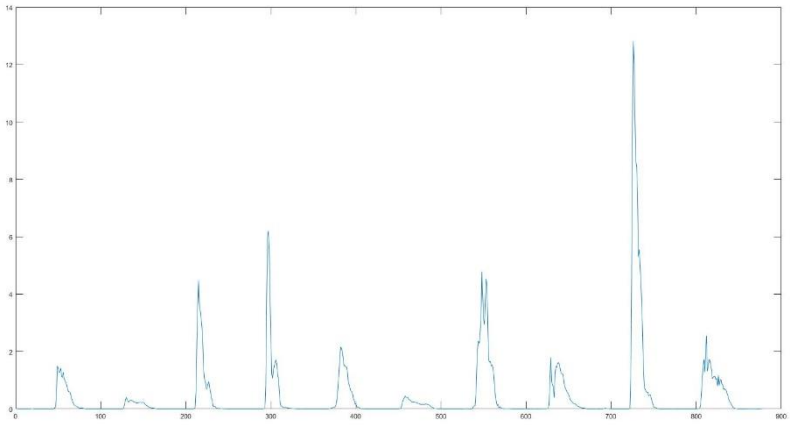

x 10^-3
8
7
6
5
4
3
2
1
0
0
100
200
300
400
500
600
700

4.3 语音0 ~ 9的波形时域信号与短时能量

语音信号0 ~ 9的时域波形：



语音信号0 ~ 9的短时能量：



4.4 平均过零率统计

语音和噪声的区别可以体现在他们的能量上，语音段的能量比噪声段的能量大，如果环境噪声和系统输入的噪声比较小，只要计算输入信号的短时能量就能够把语音段和噪声背景区分开，除此之外，还可以借助短时过零率来判断语音段的音节。

短时过零率表示一帧语音中语音信号波形穿过横轴（零电平）的次数。过零率实质上是信号频谱分布在时域的一种最简单的体现，即高频分量丰富的信

号其过零率也一般较高，因此它可以用来区分清音和浊音。

定义语音信号$x_n(m)$的短时过零率Z_n为

$$Z_n = \frac{1}{2}\sum_{m=0}^{N-1}|sgn[x_n(m)] - sgn[x_n(m-1)]|$$

式中，$sgn[.]$是符号函数，即$sgn[x] = \begin{cases} 1, (x \geq 0) \\ -1, (x \leq 0) \end{cases}$

下表为两组数据，分别来源于两位同学的0～9这10个数字的平均过零率统计。

表1：

数字	第一次	第二次	第三次	第四次	第五次	第六次	第七次	第八次	第九次	第十次	平均过零
0	19.806	21.786	21.440	20.520	20.600	18.393	25.480	24.296	22.192	20.435	21.495
1	16.385	21.714	20.750	17.379	19.370	20.720	15.917	16.480	15.375	15.960	18.005
2	33.111	35.846	35.875	38.905	38.684	39.412	33.818	31.500	30.333	35.636	35.312
3	23.048	25.261	27.385	26.696	24.762	34.484	35.250	25.708	26.241	26.360	27.519
4	20.400	22.800	22.813	24.688	21.267	23.438	21.267	24.105	24.467	22.000	22.724
5	19.313	20.962	19.269	18.577	17.346	17.348	18.130	17.214	19.071	17.818	18.505
6	24.696	24.167	25.583	23.520	24.136	24.417	22.478	23.792	24.769	28.364	24.592
7	48.310	69.885	77.259	72.458	59.000	57.917	78.462	63.960	50.615	70.280	64.815
8	39.080	37.308	33.680	34.167	33.087	32.346	33.962	29.708	36.083	31.125	34.055
9	26.194	28.853	27.926	26.130	25.300	26.789	27.143	25.333	24.429	22.765	26.086

表2：

数字	第一次	第二次	第三次	第四次	第五次	第六次	第七次	第八次	第九次	第十次	平均过零
0	20.806	22.033	17.000	16.462	15.750	18.000	19.464	17.519	16.148	18.269	18.145
1	37.607	2.333	22.071	29.000	19.241	23.680	19.241	28.815	30.654	26.229	23.887
2	37.320	35.625	37.524	16.000	34.800	34.688	28.278	34.632	41.280	30.500	33.065
3	100.310	120.276	121.129	218.429	60.696	106.875	95.613	94.882	30.222	106.188	105.462
4	120.583	117.029	127.343	116.564	130.576	127.500	133.706	86.029	102.088	134.406	119.582
5	11.833	14.455	17.400	13.333	17.615	14.056	19.857	13.813	18.000	14.692	15.505
6	13.500	21.846	67.000	21.240	19.333	9.800	20.500	10.667	17.714	17.063	21.866
7	95.429	93.433	81.323	86.467	202.500	16.684	25.000	78.152	77.613	79.800	83.640
8	32.333	29.192	28.000	29.182	30.360	34.250	27.458	28.476	29.000	27.850	29.610
9	26.813	20.417	29.000	20.444	45.667	22.545	26.600	19.889	29.357	21.500	26.223

5. 单变量分类结果分析

通过对录音处理所得数据观察，我们发现同样的数字的过零率是相近的，而不同数字的过零率则有一定的区别，在数据分布上有可以看出一定的聚类倾向。如果我们构建一个用大量语音过零率构建的数据库，而对所有需要分类的语音在库内进行匹配，认为语音库中与该语音过零率相差最小的即认为是测试语音所属

的分类。

我们另外录音了数字 0~9 的语音各一份作为测试集，将之前处理的 0~9 每个数字的语音各 10 份作为数据库，以两个语音片段过零率差的绝对值作为相似程度评价标准，进行这 10 个语音片段的识别测试，我们将语音片段的分类的测试结果和实际真实类别记录如下：

表 3 额外声音测试结果

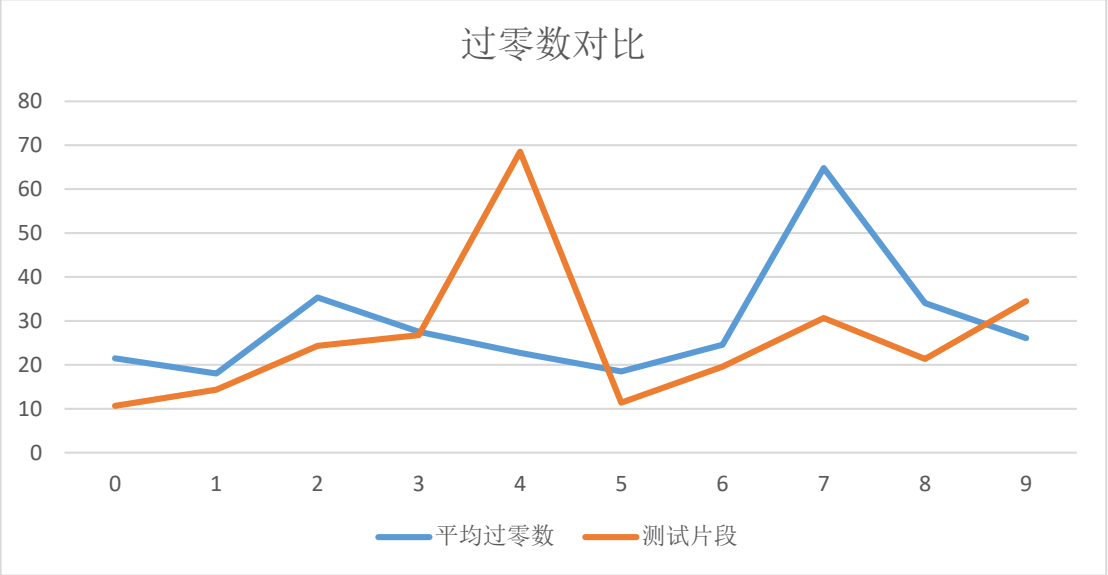
实际分类	0	1	2	3	4	5	6	7	8	9
测试结果	1	1	0	9	7	1	0	2	4	3

我们发现，实际分类的效果非常低，在十个测试片段中只正确了一个片段。我们比较了一下数据集不同数字的平均过零率与测试片段的过零率，如下表。可以看出测试片段与数据集中的语音片段在过零率的整体分布和趋势上还是存在着整体的偏差。我们认为这可能是由于不同人的说话习惯与音色导致的不同，每个人在说话时发音方式及时长都不同。因此我们决定用同一个人同次录音所得的声音进行测试。

表 4 数据集与测试片段过零率

	0	1	2	3	4	5	6	7	8	9
平均过零数	21.49	18.00	35.31	27.51	22.72	18.50	24.59	64.81	34.05	26.08
测试片段	10.67	14.33	24.32	26.76	68.51	11.37	19.59	30.66	21.33	34.48

我们将数据集中 0~9 每个语音的十个片段拆分成比对集与测试集两部分，并且采用交叉验证的方法以进行多次测试。每次测试我们随机从每个语音的十个片段中选择一个作为测试片段，即每次测试有 90 个比对片段和 10 个测试片段，通过随机划分测试即可。我们进行了 1000 次随机划分，共测试了 100000 次，其



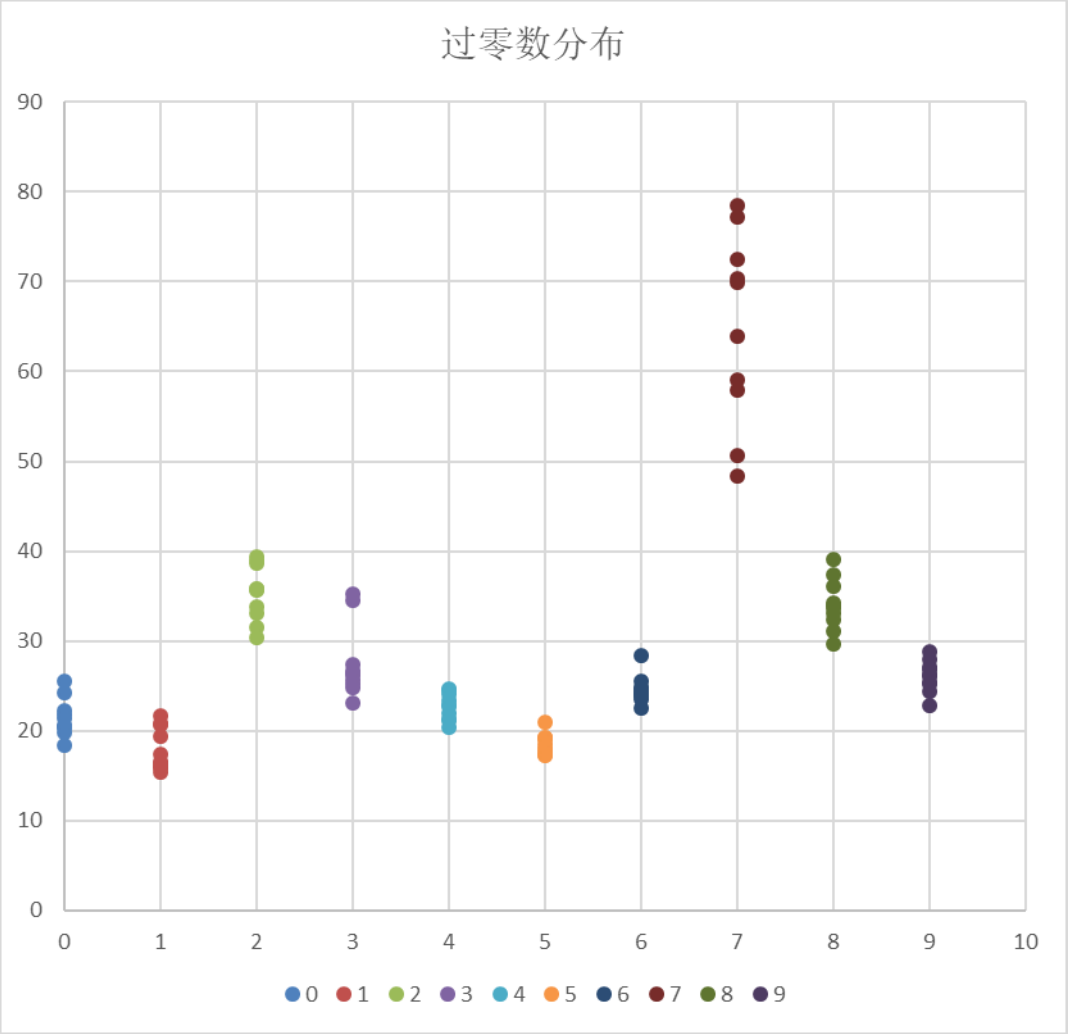
中有 46344 次测试结果准确，正确率达到了 46.3%，由此可见，由于说话人和录音场合导致的实验误差还是相当地大的。我们记录下了每种数字语音的识别准确率如下表

表 5 交叉测试准确率

	0	1	2	3	4	5	6	7	8	9
准确率(%)	31.72	70.00	42.01	10.41	40.27	71.41	31.96	100	32.71	32.95

由于 0，2，8，9 等数字的过零率均集中在一起，所以这几个数字之间的相互混淆致使其分类准确率低，7 虽然在数据集中有着较大的方差，但由于其过零率较高，分布距离其他数字较远，所以和其他数据能很好地分离开，分类准确率甚至达到了 100%，1 和 5 是由于声母和韵母发音相同，其波形一致性比较高，而

声母韵母不同的数字在发音上可能变化会比较大，这也可能是解释其高准确率的原因。我们曾经考虑过将数据集中有明显偏离的数据筛除，但是考虑到各类样本的均衡性，我们依然将这些样本保留了下来。



6. 算法优化

通过上述实验结果，可以发现，仅依靠一段语音的整体平均过零率来进行语音识别，存在较大误差，并不能很好地区分各个数字之间的差异。为了提高识别的准确率，我们优化了算法，在识别过程中加入了更多的特征。我们将语音分为若干段，并计算每一段的平均过零率和能量，将过零率向量和能量向量作为特征，通过 KNN 算法进行语音识别。并通过交叉验证的方式扩充实验次数，实验结果如下：

当去过零数和平均能量的权重为（1， 0）时（即只考虑过零数的影响，而忽略能量的影响），10000 次语音识别正确了 5309 次，正确率为 53.09%，各数字识别率如下：

	0	1	2	3	4	5	6	7	8	9
准确率(%)	48.9	31.3	62.9	10.4	31.1	82.3	100	74.4	70.0	19.6

当去过零数和平均能量的权重为（1，10）时，10000 次语音识别正确了 6482 次，正确率为 64.82%，各数字识别率如下：

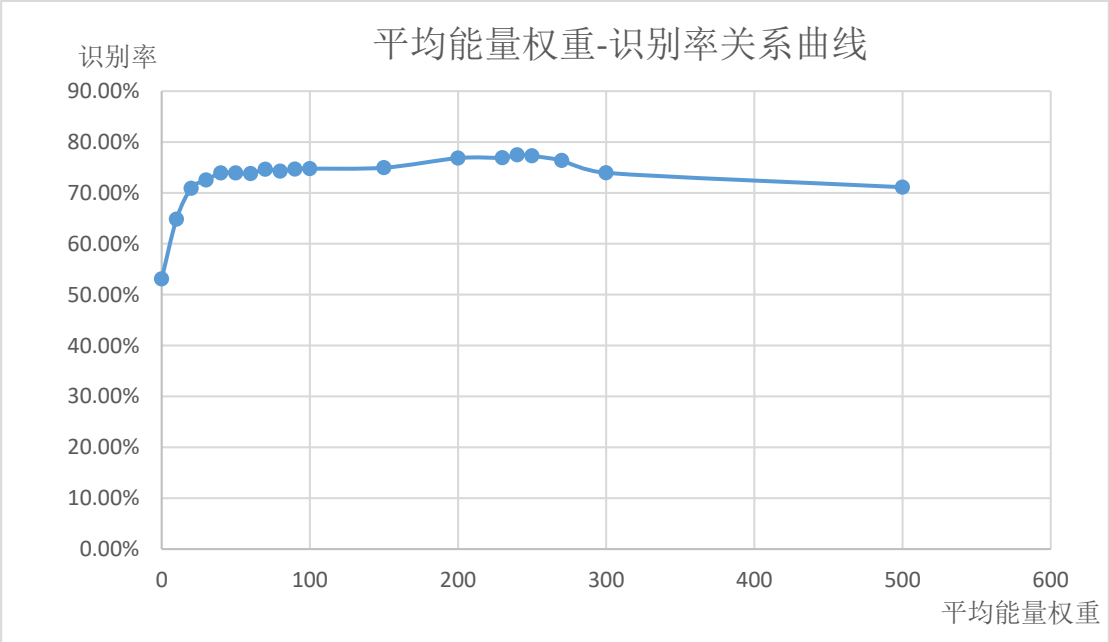
	0	1	2	3	4	5	6	7	8	9
准确率(%)	70.1	52.6	79.9	14.2	33.9	83.6	100	72.3	84.6	57.0

当去过零数和平均能量的权重为 (1, 100) 时，10000 次语音识别正确了 7477 次，正确率为 74.77%，各数字识别率如下：

	0	1	2	3	4	5	6	7	8	9
准确率(%)	90.3	93.5	90.7	32.2	34.2	69.4	92.6	72.2	100	74.4

通过对比可以得出，将语音分为前后若干段，可以更好地提取出语音的时序特征，相较于之前的单变量分类，语音识别率有明显提升。

经过测试不同的权重，我们发现权重对语音识别的准确率有着明显的影响，我们固定过零数的权重为 1，改变平均能量的权重，测试结果如下：



可以看到，当平均能量权重约为 250 时，有最高的识别率为 77.28%。我们认为，当取（1，250）权值时，过零数和平均能量处于相当的数量级，分类器可同时考虑这两种特征，实验结果较为理想。

通过对比我们发现，相较于只考虑过零数的情况下，加入了平均能量这一特征，使得绝大部分数字的识别率都有了明显的提升，但 5、6 这两个数字的识别率却有所下降，我们认为这两个数字在能量上并不具有区分性。此外，3、4 两个数字的识别率始终在 40%以下，需要考虑其他特征。

然而，由于在考虑能量特征时未进行归一化处理，因此能量特征与音量具有显著关系，若用其他语音片段进行测试，可能结果不会很理想，需要进一步改善算法。

附录：

1. 语音信号采集 Matlab 代码

```
recObj = audiorecorder(44100,16,1);
disp('Start speaking.')
recordblocking(recObj, 10);     % 录音录 10 秒钟
disp('End of Recording.');
%play(recObj);     % 回放录音数据
myRecording = getaudiodata(recObj);       % 获取录音数据
plot(myRecording);    % 绘制录音数据波形
wavwrite(myRecording,44100,16,'1');
```

2. 单特征语音信号处理 Matlab 代码

```
framelength=900;
step=700;
[data,fs]=audioread('1.wav');
flag=0;
now=1;
%mean(data)
%alt=data-mean(data);
framenumber=fix((length(data)-200)/step);
energy=ones([framenumber,1]);
for i=1:framenumber
    energy(i)=0;
    for j=(i-1)*step+1:(i-1)*step+framelength
        energy(i)=energy(i)+data(j)*data(j);
        %zeros(i)=zeros(i)+abs(sign(alt(j+1)-sign(alt(j))));
    end
    if flag==0&&energy(i)>0.025
        flag=1;
        for k=i:-1:1
            if energy(k)<0.002
                starting(now)=k+1;
                break
```

```
            end
        end
    end
    if flag==1&&energy(i)<0.002
        flag=0;
        ending(now)=i;
        now=now+1;
    end
    %zeros(i)=zeros(i)/2;
end
plot(energy);

for i=1:10
    zeroing=zeros([ending(i)-starting(i),1]);
    ending(i),starting(i)
    for j=starting(i):ending(i)-1
        alt=sign(data((j-1)*step+1:(j-1)*step+901)-mean(data((j-
1)*step+1:(j-1)*step+901)));
        for k=1:framelength
            zeroing(j-starting(i)+1)=zeroing(j-
starting(i)+1)+abs(alt(k+1)-alt(k))/2;
        end
```

```
    end
    hahaa(i)=mean(zeroing);
end
```

3. 单变量测试结果 Matlab 代码

```
%单独测试结果
%testing是测试用数据
%data是数据集


for i=1:10
    now=testing(i);
    dis=abs(now-data(1,1));
    pre(i)=1;
    for j=1:10
        for k=1:10
            if abs(now-data(j,k))<dis
                pre(i)=j;
                dis=abs(now-data(j,k));
            end
        end
    end
end
pre
```

4. 单变量交叉测试 Matlab 代码

```
%本程序用于对语音数据集内部进行交叉测试
%data为一二维数组存储所有过零数
right=0; %正确数量
all=0; %测试总数
everyone=[0,0,0,0,0,0,0,0,0,0]; %每个数字的正确率
for yyy=1:10000
    index=unidrnd(10,10,1);
    for i=1:10
        tested=data(i,index(i));
        dis=abs(tested-data(1,1));
        pre(i)=1;
        for j=1:10
            for k=1:10
                if index(j)==k
                    continue;
                end
                if abs(tested-data(j,k))<dis
                    pre(i)=j;
                    dis=abs(tested-data(j,k));
                end
            end
```

```
        end
    end
    all=all+10;
    for j=1:10
        if pre(j)==j
            right=right+1;
            everyone(j)=everyone(j)+1;
        end
    end
end
```

5. 多特征信号处理 Matlab 代码

```
clear all
framelength=900;
step=700;
for shu=1:10
    [data,fs]=audioread( [num2str(shu-1),'h.wav']);
    flag=0;
    now=1;
    %mean(data)
    %alt=data-mean(data);
    framenumber=fix((length(data)-200)/step);
    energy=ones([framenumber,1]);
    for i=1:framenumber
        energy(i)=0;
        for j=(i-1)*step+1:(i-1)*step+framelength
            energy(i)=energy(i)+data(j)*data(j);
        end
        if flag==0&&energy(i)>0.025
            flag=1;
            for k=i:-1:1
                if energy(k)<0.002
                    starting(now)=k+1;
```

```
                    break
                end
            end
        end
        if flag==1&&energy(i)<0.002
            flag=0;
            ending(now)=i;
            now=now+1;
        end
        %zeros(i)=zeros(i)/2;
    end
    %plot(data);
    %xlim([0,630000]);
    %plot(energy);

    Segs=4;%每段语音分段数
    for i=1:10
        zeroing=zeros([ending(i)-starting(i),1]);
        %ending(i),starting(i)
        for j=starting(i):ending(i)-1
            alt=sign(data((j-1)*step+1:(j-1)*step+901)-mean(data((j-
1)*step+1:(j-1)*step+901)));
```

```
            for k=1:framelength
                zeroing(j-starting(i)+1)=zeroing(j-
starting(i)+1)+abs(alt(k+1)-alt(k))/2;
            end
        end
        for l=1:Segs
            part=floor((ending(i)-starting(i))/Segs);
            %以（第i个数，第l段）的形式建立向量
            Crossing(shu,i,l)=mean(zeroing(part*(l-1)+1:part*l));%每段平
均过零数
            AvgEnergy(shu,i,l)=mean(energy(starting(i)+part*(l-
1)+1:starting(i)+part*l));%每段平均能量
        end
    end
end
```

6. 多特征分类交叉验证 Matlab 代码

```
function [all1,right1,everyone1] =test3(data1,data2,lambda1,lambda2)
%本程序用于对语音数据集内部进行交叉测试
%data为一二维数组存储所有过零数
right=0; %正确数量
all=0; %测试总数
everyone=[0,0,0,0,0,0,0,0,0,0]; %每个数字的正确率
for yyy=1:1000
    index=unidrnd(10,10,1);
    pre=zeros(10,1);
    for i=1:10
        tested1=data1(i,index(i),:);
        tested2=data2(i,index(i),:);
        dis=ones(5,1)*4444444;
        indexs=zeros(5,1);
        labels=zeros(5,1);
        %dis=abs(tested-data(1,1));
        for j=1:10
            for k=1:10
                if index(j)==k
                    continue;
                end
```

```
                for in=5:-1:1
                    %data1(j,k,:)-tested1
                    %normest(data1(j,k,:)-tested1)*lambda1+normest(data2(j,k,:)-
tested2)*lambda2
                    tempdis=norm(squeeze(data1(j,k,:)-
tested1))*lambda1+norm(squeeze(data2(j,k,:)-tested2))*lambda2;
                    if tempdis<dis(in)
                        if in>1
                            indexs(in)=indexs(in-1);
                            labels(in)=labels(in-1);
                            dis(in)=dis(in-1);
                        else
                            indexs(in)=k;
                            labels(in)=j;
                            dis(in)=tempdis;
                        end
                    else
                        if in<5
                            indexs(in+1)=k;
                            labels(in+1)=j;
                            dis(in+1)=tempdis;
                        end
```

```
                    break;
                end
            end
        end
    end
    [M,F,C]=mode(labels);
    if F==1
        pre(i)=M;
    else
        for la =labels'
            hhhhh=0;
            for aa=C{1}
                if aa==la
                    pre(i)=la;
                    hhhhh=1;
                    break;
                end
            end
            if hhhhh==1
                break;
            end
        end
```

```
            end
        end
        all=all+10;
        for j=1:10
            if pre(j)==j
                right=right+1;
                everyone(j)=everyone(j)+1;
            end
        end
    end
    right1=right;
    all1=all;
    everyone1=everyone;



end
```