西安交通大學
XI'AN JIAOTONG UNIVERSITY

《数字信号处理》
语音识别系列实验

小组成员:

1. 李欣阳 自动化 43 2140504066
2. 任自强 自动化 41 2140504018
3. 苗鸿易 自动化 41 2140504017

基于时域分析技术的语音信号识别

李欣阳，任自强，苗鸿易

（西安交通大学，自动化科学与技术系，陕西，西安，710049）

摘 要：语言是人类交流信息的主要手段之一，计算机的语音识别技术正引起越来越多的关注。本文基于语音信号的时域特征，在信号已分帧加窗的基础上，根据信号的短时能量和短时过零率，实现孤立词语音的端点检测，并成功推广到连续多词汇的语音信号。在语音信号的特征提取上，选用平均短时能量、平均短时过零率、前后半程能量比、前后半程过零率比四个特征，借助最近邻算法（KNN）和 BP 神经网络实现对语音信号的模式匹配，并取得了较好的效果。

关键词：时域特征 端点检测 短时过零率 短时能量 KNN 算法 BP 神经网络

1 引言

1.1 研究背景

随着计算机技术的迅速发展，人机交互成了人们关注的热点问题。作为人机交互的重要组成部分，人机对话技术具有贴近人类自然语言交流、使用方便快捷的特点，一直处于蓬勃发展当中。所谓语音信号识别技术就是指使用机器通过识别和理解人类的语音后，将语音信号转变为相应的文本或命令的技术，包括语音采集处理、语音特征提取和识别等部分，是人机对话技术的主要方面之一，具有很高的研究价值。

语音识别起始于 20 世纪 50 年代，这是语音识别技术的开端。60 年代末的语音信号线性预测技术和动态时间规的发展解决了语音的特征提取和不等长匹配问题，其研究特点以孤立字语音识别为主，把孤立字作为一个整体来建立模板。20 世纪 80 年代初，线性预测和动态时间规技术催生矢量量化(VQ)技术和隐马尔柯夫模型(HMM)理论，实现了基于线性预测倒谱和动态时间规技术的特定人孤立小词汇量语音识别系统。此时语音识别研究的重点之一是连接词语音识别，各种连接词语音识别算法被开发，如多级动态规划语音识别算法。另一重要发展是语音识别算法从模板匹配技术转向基于统计模型技术，研究从微观转向宏观，不再追求细化语音特征，而是从统计的角度来建立最佳的语音识别系统。隐马尔柯夫模型(HMM)是其典型，它能很好的描述语音信号的时变性和平稳性。统计语言模型也开始取代基于规则语言模型的方法。HMM 的研究使大词汇量连续语音识别系统的开发成为可能。80 年代中期，在实践中成功应用了 HMM 模型和人工神经网络(ANN)，20 世纪 90 年代之后，人工神经网络技术的应用成为语音识别的一条新途径，它具有自适应性、并行性、非线性、鲁棒性、容错性和学习特

性，在结构和算法上都显示出了很大的潜力。此时，在细化模型的设计、参数提取和优化以及系统的自适应技术上取得了关键进展，语音识别技术进一步成熟，语音识别系统从实验室走向实用[1]。

语音识别技术目前广泛应用于语音拨号、语音导航、工业设备控制、语音文档检索、简单的听写设备录入等领域，它在机器人控制、保密系统等领域正成为关键技术。随着语音识别技术的发展，孤立词、小词汇量的语音识别系统在日常生活中已经得到广泛应用。随着研究的深入，语音识别技术将会拥有更为广阔的应用前景。

1.2 语音识别的基本原理

目前，大多数语音识别系统都采用了模式匹配的原理。根据这个原理，未知语音的模式要与已知语音的参考模式逐一进行比较，最终最佳匹配的参考模式被作为识别结果。图 1 就是根据模式匹配原理构成的自动语音识别系统的原理方框图。

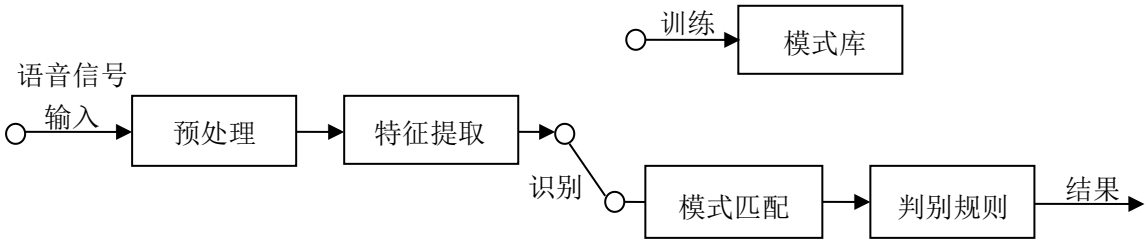

图 1 模式匹配法原理框图

待识别语音经过话筒变换成图中语音信号后作为系统输入，经过预处理后进行特征提取，常用的特征有短时平均能量或幅度、短时平均过零率、短时自相关函数、线性预测系数、倒谱、共振峰等。训练在识别之前进行，是通过让讲话者多次重复语音，从原始语音样本中去除冗余信息，保留关键数据，再按规则对数据加以聚类，形成模式库。模式匹配是整个语音识别系统的核心，是根据一定的准则以及专家知识，计算机输入特征与库存模式之间的相似度，判断出输入语音的语意信息。

2 语音信号的时域特征

对于孤立词、小词汇量的语音信号，从时域角度看，最重要的特征主要有两个——短时平均能量和短时平均过零率，数字 0-9 的汉语语音在这两个特征上会有或多或少的区别，这些区别就是识别相应语音信号的主要依据。

2.1 短时能量

对于经过加窗分帧的语音信号，短时平均能量描述了信号在某一帧的平均或者总能量的大小，由于信号的幅度即是能量的直接反应，所以也可以用平均或者

总幅度来表示能量的相对大小，这两种表示方法具有相同的意义。假设表示数字语音信号序列，N 为窗长，也就是每帧的序列长度，那么短时总能量和短时平均能量可以表示为

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

或者用幅度表示为

$$E_n=\sum_{m=0}^{N-1}\left|x_n(m)\right| \quad \text{或者} \quad E_n=\frac{1}{N}\sum_{m=0}^{N-1}\left|x_n(m)\right|$$

2.1 短时过零率

信号的幅度值从正值到负值要经过零值，称其为过零。统计信号在一帧内过零次数就称之为短时过零率。对于连续语音信号来说，过零就是信号波形穿过横轴的次数，对离散信号来说则是指相邻两个点的取样值符号相反。由于语音信号具有短时平稳性，要对其进行分帧处理，所以在这种条件下得到的过零率又被称为短时过零率，其定义为

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

短时平均过零率是语音信号在时域的重要特征，它在一定程度上反映了信号频率在时间轴上的分布情况。如果某一帧过零率较大，则说明该处信号的幅值符号发生了剧烈的变化，也就表现为高频分量在该处较为集中，可以说短时过零率是频域特征在时域的反映。

3 语音信号的预处理

3.1 语音信号的分帧加窗

语音信号具有短时平稳性，较短时间内可以认为语音信号近似不变，这样就可以把语音信号分为一些短片段来进行处理，这就是分帧。语音信号的分帧是采用可移动的有限长度的窗口进行加权的方法来实现的，本文首先对数字语音信号进行幅度归一化处理，然后取每帧长度为 256，由于 wav 格式的语音文件采样率为 44100，所以每帧时间长度约为 5.8ms，在 Matlab 下编写函数 enframe 实现对语音信号的分帧加窗。

3.2 基于短时能零特征的端点检测方法

语音信号一般可分为无声段、清音段和浊音段[2]。无声段是背景噪声段，平均能量最低；浊音段为声带振动发出对应的语音信号段，平均能量最高；清音段为空气在口腔中的摩擦、冲击或爆破而发出的语音信号段，平均能量居于两者之间。采用基于幅度的算法来检测浊音通常是可行也是可靠的，但对清音而言，除非信号具有极高的信噪比，否则，采用基于幅度算法从背景噪声中鉴别出清音就

不够可靠了。此时，需要用到语音信号的另一重要特征，即过零率，它指一定时间内信号穿越零电平的次数。清音段与无声段的波形特点有明显不同，无声段信号变化比较缓慢，清音段信号由气流摩擦产生，在幅度上的变化比较剧烈，穿越零电平次数较多[3]。大量实验表明，通常清音段过零率最大，浊音段过零率的变化范围较小。可见，振幅特征适合检测浊音，而过零率则适合检测清音，为了同时检测两者，一般综合利用两种特征，采用双门限法进行端点检测[4]。

首先为短时能量和过零率分别确定两个门限 amp1、amp2 和 zcr1、zcr2，amp1 和 zcr1 是低门限数值较小，对信号的变化比较敏感，很容易超过；amp2 和 zcr2 是高门限数值较大。低门限被超过未必是语音的开始，有可能是很短的噪声引起的，高门限被超过并且接下来的自定义时间段内的语音超过低门限，意味着信号开始。

此时整个端点检测可分为四段：静音段、过渡段、语音段、结束。实验时使用一个变量表示当前状态，最初在静音段，如果能量或过零率超过低门限，就开始标记起始点，进入过渡段。当两个参数值都回落到低门限以下，就将当前状态恢复到静音状态。而如果过渡段中两个参数中的任一个超过高门限，即被认为进入语音段。在处于语音段时，如果两参数降低到门限以下，而且总的计时长度小于最短时间门限，则认为是一段噪音，继续扫描以后的语音数据，否则标记结束端点。具体使用该算法时，需要反复调试门限参数以使端点检测的性能最佳，下图给出对于单独语音“1”的端点检测结果。

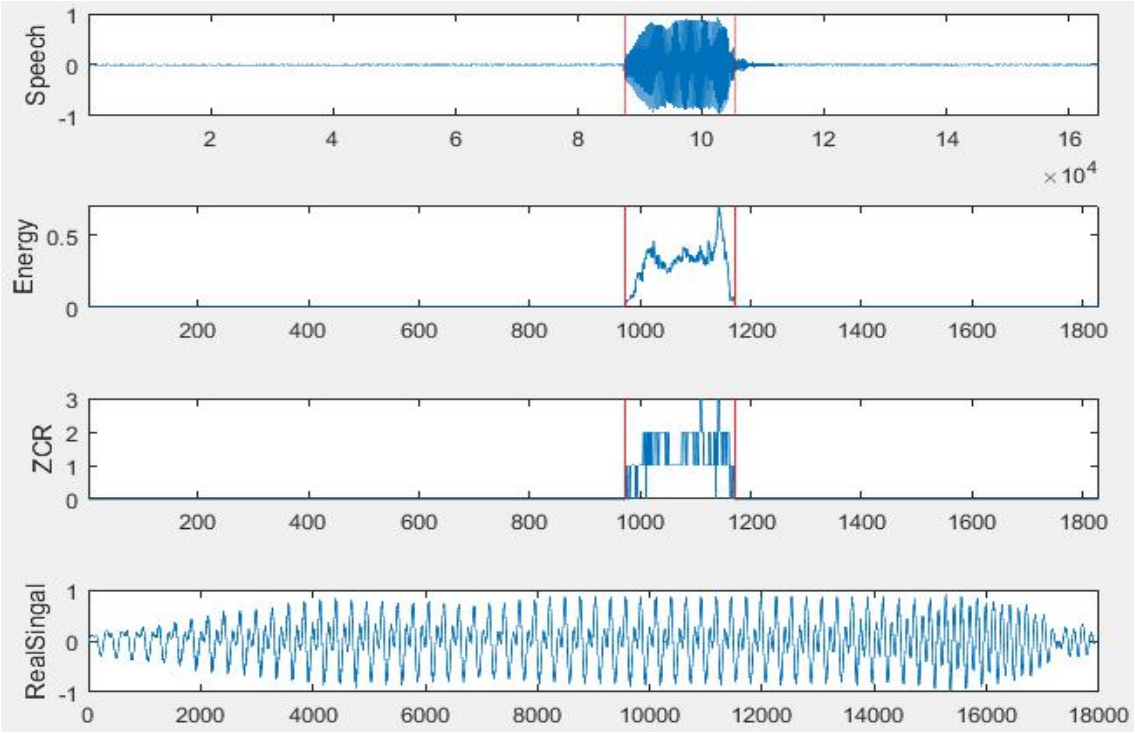

图 2 语音信号“1”的端点检测结果

上图表明该算法用于孤立词汇语音的端点检测时具有较好的效果，但是仍然具有一定的误差，主要发生在语音末尾的清音段，此时声音由发音的末尾气流同

口腔唇齿摩擦产生，在正常语音交流时凭借人耳听力很难察觉该部分的声音，大多数情况下是湮没在背景噪声中，和背景噪声一起成为应该被滤去的部分。所以就人的听觉而言，该部分语音并不影响人耳对语音内容的辨识，忽略之后也不影响语音信号识别的准确度。

该算法同样可应用于连续多词汇语音信号的端点检测，但为了保持较好的检测效果，需要做出部分参数和终止标志的调整，使程序运行时能顺利的完成整个语音信号的检测，而不是在检测出第一个词语或者仅仅去除首末噪声后终止。具体的修改办法是：调用时采用循环调用，检测到一段语音信号后，计算其短时过零率和短时能量，随后将其单独保存起来并从原来的整段信号中删除，继续进行下一段语音信号的端点检测。由于算法核心未变，仍然是基于短时能量和短时过零率，所以为了使两个词汇之间分界鲜明，要求词与词之间具有足够的静音时间，即录入语音信号时尽量词与词之间停顿恰当、间隔鲜明。图 3 是对连续多词汇语音信号的一组检测结果，结果显示该端点检测算法具有较好的适用性，但仍然具有微小误差。

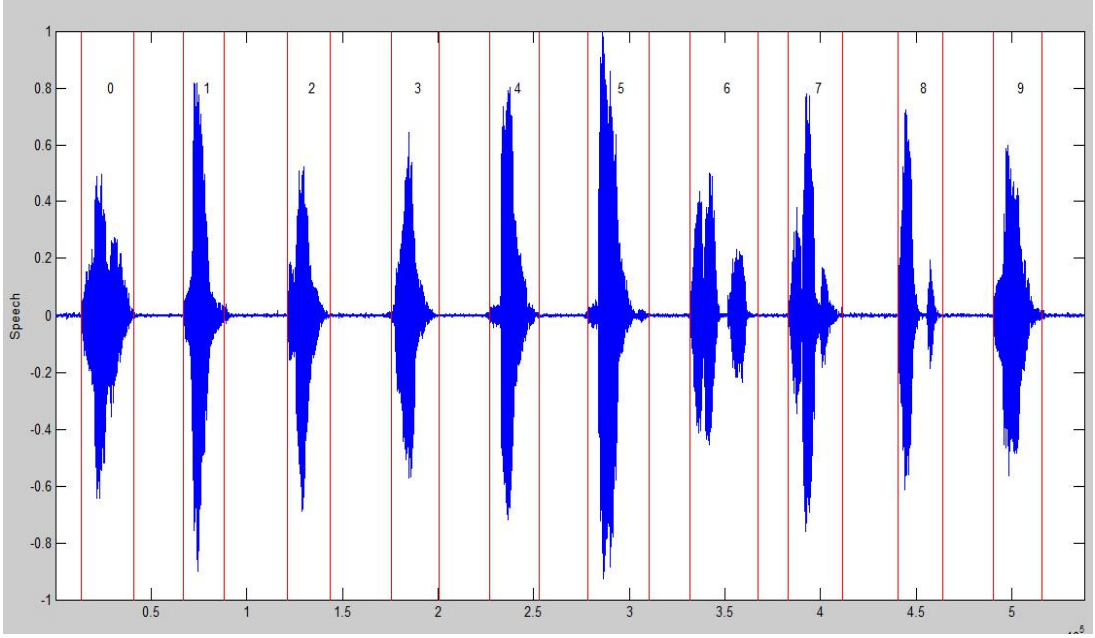

图 3 连续多词汇语音的端点检测

4 分类器的训练及模式匹配

4.1 语音信号特征的选取

对于训练分类器而言最重要的就是特征的选取，选取恰当的特征对于正确区分不同的语音信号具有很大的帮助。就时域特征而言，为了克服序列长度对信号短时能量和短时过零率的影响，本文选取平均短时能量和短时平均过零率作为信号特征。此外，考虑到平均短时能量和平均短时过零率仅仅反映了信号的总体特征，而信号内部能量分布和过零率的相对分布情况则被忽略了，这对于语音信号

而言是个很大的不利，因为信号内部能量及过零率的分布同样是区分语音信号的重要特征，因此为了克服特征选取上的这个弊端，本文额外添加前后半程能量比和前后半程过零率比，作为反映信号能量和过零率大致分布情况的特征参数。

综上所述，本文选取的语音信号特征参数有四个，即平均短时能量、平均短时过零率、前后半程能量比、前后半程过零率比，0-9 十个数字 10 组样本的具体特征参数值见附录表 1。

4.2 基于最近邻算法（KNN）的语音信号识别

4.2.1 分类器原理

KNN 算法原理简单说来，就是从训练集中找到和新数据最接近的 k 条记录，然后根据他们的主要分类来决定新数据的类别。根据先前确定的四个特征，将计算好的特征矩阵带入 KNN 模型，每行是一个特征向量，列数是特征的维数，行数是特征的个数；总共是 0-9 十个类别，每个类别有十个训练样本，最后生成模型用于语音信号的模式匹配。

4.2.2 语音信号的匹配结果

训练完成后把事先录制好的 10 组测试样本带入 KNN 模型，得到相应的模式匹配结果，详细情况见表 1。

表 2 最近邻方法的匹配结果

检测序号	0	1	2	3	4	5	6	7	8	9
一	1	1	2	3	4	5	8	8	8	6
二	1	0	2	3	4	5	6	7	8	6
三	1	1	2	3	4	5	8	8	8	9
四	0	0	2	3	4	5	6	1	8	9
五	0	1	4	3	2	6	6	7	8	9
六	1	0	2	3	4	6	6	7	8	6
七	0	1	8	3	4	6	6	7	8	6
八	0	1	2	3	2	5	6	7	8	9
九	0	0	2	3	4	5	6	1	8	9
十	1	0	2	3	5	5	6	7	8	6
识别率	50%	50%	80%	100%	70%	70%	80%	60%	100%	50%
综合识别率	65%									

分析上表可以发现，数字“3”和“8”的识别率最高，均达到 100%，其他数字的识别率则不甚理想，且在不同数字之间识别率表现出较大差异。KNN 算法的综合识别率在 65%左右，准确度还有待提高，说明算法和特征提取上仍然有待改进。

4.3 基于 BP 神经网络的语音信号识别

4.3.1 BP 神经网络的训练

结合本文研究对象，用于训练 BP 神经网络的特征参数矩阵是一个 100×4 的矩阵，其中每行代表一个特征总共四行，每列代表一个语音样本，十个数各十组总共 100 个；训练采用有监督的学习方式，目标输出是一个 100×10 的由 10 个单位阵构成的矩阵，其中每行代表一个类别，每一列元素“1”所在的位置就是该列样本所应在的类别。

采用 Matlab 自带的神经网络工具箱，采用 newff 函数构建神经网络，由于三层神经网络可以逼近任意连续函数[5]，因此本文采用三层前馈网络，各层传递函数均采用最常用的对数 *S* 型传递函数 *logsig*[6]，即

该 BP 网络的最大训练次数设置为 50000，最小执行梯度设为，目标性能，每隔 10 次反馈一次当前训练情况。

4.3.2 语音信号的匹配结果

训练完成后把事先录制好的 10 组样本的特征参数带入神经网络，得到相应的匹配结果，输出矩阵每一列最大的数所在的位置就是当前语音信号所在的类别，详细情况见表 2。

表 2 BP 神经网络的匹配结果

检测序号	0	1	2	3	4	5	6	7	8	9
一	0	1	2	9	4	1	1	4	8	7
二	5	1	5	3	4	5	1	3	8	5
三	5	1	2	3	7	1	6	4	8	7
四	0	1	5	3	4	1	6	7	8	9
五	0	1	8	3	3	5	6	7	8	9
六	0	1	2	3	4	5	6	7	8	9
七	5	1	2	3	4	5	6	4	5	9
八	0	1	8	3	3	5	6	7	8	7
九	0	1	8	3	4	5	6	7	8	9
十	0	1	2	3	4	5	6	7	8	3
识别率	70%	100%	50%	90%	80%	70%	80%	60%	90%	50%
综合识别率	74%									

分析以上结果可以发现，“1”、“3”、“8”的识别率最高，分别达到 100%、90%和 90%，而“2”和“9”的识别率则不太乐观，仅为 50%，综合识别率为 74%左右。考虑到神经网络的特性，由于只使用了 10 组训练样本，增大训练集

对提高识别率很有帮助，此外特征的选取仍然有待改善，需要寻找更多更有代表性的特征。

5 评价与改进

在语音信号的端点检测方面，本文采用短时能量和短时过零率相结合的方法，短时能量用于区别浊音和背景噪声，短时过零率用于区别清音和背景噪声，在孤立词的端点检测上取得了很好的效果，并且在推广到连续多词汇语音时，仍能保持其准确度。

在分类器的选取上，本文选取两种分类器——KNN 和 BP 神经网络，它们表现出各自的优缺点：KNN 算法简单有效，重新训练的代价低，但是需要大量样本且准确度较低；BP 神经网络分类精度高，具有较好的容错能力，但是同样需要较大的训练集且训练耗时长、重新训练代价高。

总结下来，想要实现语音信号的精确识别，仅仅依靠时域特征是远远不够的，还需要引入更多更有效的特征，时域频域相结合才能使识别精度最大化。

参考文献

[1]高新涛，陈乖丽，语音识别技术的发展现状及应用前景[J]，甘肃科技纵横，2007，36(4)：13-13.
[2]宋建华，包玉花，梁跃， 一种基于Matlab的语音信号端点检测方法[J]，黑龙江大学工程学报，2008，35(4)：74-76.
[3]刘庆升，徐霄鹏，黄文浩，一种语音端点检测方法的探究[J]，计算机工程，2003，29(3)：120-121.
[4]顾亚强，赵晖，吴波，一种语音信号端点检测的改进方法[J]，计算机仿真，2010，27(5)：340-343.
[5]陈敏歌，基于神经网络的语音信号识别及其实现[D]，陕西科技大学，2008.
[6]高燕，神经网络方法在语音信号检测中的应用分析[J]，工业，2016(6)：00249-00249.

附录

附录一：图表

附表 1 训练样本的特征参数

样本序号	averageZCR	divZCR	averageAMP	divAMP
0.1	1.2443	1.3505	0.1873	0.9810
0.2	1.6488	0.9103	0.3280	0.7205
0.3	2.3390	1.3590	0.2773	2.7427
0.4	9.5281	8.3263	0.3013	1.2051
0.5	14.2167	7.9907	0.5354	3.2443
0.6	0.9669	1.2479	0.1445	1.0699
0.7	1.1132	1.1852	0.2701	2.0021
0.8	8.3072	8.9347	0.5300	1.7876
0.9	3.1890	1.0430	0.4857	1.6223
0.10	9.2963	13.3429	0.3985	2.7968
1.1	1.4052	1.2797	0.1997	1.0089
1.2	1.5148	0.8551	0.3197	0.7098
1.3	2.8138	1.4286	0.7225	4.0127
1.4	22.4638	0.9796	0.2090	1.4816
1.5	16.1327	5.6097	0.7248	2.9363
1.6	0.9476	0.6832	0.1364	0.6030
1.7	1.0000	1.0494	0.2441	2.2308
1.8	9.1770	9.3700	0.8008	2.0506
1.9	2.9623	1.3550	0.5772	2.7955
1.10	3.2475	11.1266	0.1319	2.3314
2.1	1.2366	1.1457	0.2464	1.9327
2.2	1.9171	0.6372	0.3746	0.8661
2.3	2.7939	1.4730	0.4431	3.1050
2.4	22.9348	1.0095	0.6242	0.8885
2.5	17.3301	3.2966	1.8227	2.7629
2.6	1.2647	0.8786	0.1988	1.0206
2.7	1.0000	1.2651	0.2011	2.3233
2.8	8.9877	10.3161	0.5340	1.4584
2.9	3.4702	1.0630	0.4564	1.7401
2.10	7.9521	18.0328	0.3143	1.7821
3.1	0.8309	0.6042	0.1465	0.5711
3.2	1.1702	0.7054	0.2962	0.6733
3.3	2.7122	1.4481	0.3617	2.9685
3.4	7.6762	0.9487	0.1547	0.8464
3.5	15.9223	4.1530	0.6962	2.5151
3.6	0.5090	0.3148	0.1067	0.4201
3.7	1.0462	1.4578	0.2160	2.1607

3.8	9.2607	9.8894	0.5786	2.1043
3.9	2.4605	1.5616	0.4288	3.8245
3.10	7.6136	17.6111	0.2947	2.1401
4.1	1.2439	0.9402	0.1591	0.5793
4.2	1.2841	0.7937	0.3133	0.8509
4.3	2.3333	1.7870	0.3459	3.6226
4.4	9.8982	11.0919	0.2012	1.0085
4.5	16.7043	4.7537	0.8401	4.3287
4.6	0.8618	0.9587	0.1235	0.6735
4.7	1.1505	1.8158	0.2782	2.9601
4.8	9.4068	11.1978	0.4687	1.4281
4.9	3.3289	1.1350	0.5937	2.2745
4.10	6.0507	15.7000	0.2749	2.3136
5.1	1.1100	1.0296	0.1790	0.6663
5.2	1.3706	0.7647	0.3081	0.7173
5.3	2.1533	1.2180	0.2498	1.9603
5.4	9.2130	9.2864	0.2183	0.8443
5.5	13.5991	15.8629	0.4948	2.8040
5.6	0.4799	0.4444	0.1134	0.4073
5.7	0.9538	1.9815	0.2455	2.6844
5.8	8.9453	7.9453	0.5829	1.5628
5.9	2.9787	1.1538	0.3940	2.1311
5.10	8.0339	16.5556	0.3968	2.5486
6.1	1.1906	0.5551	0.1592	0.5151
6.2	1.7207	0.6500	0.3108	0.8416
6.3	2.4692	1.4318	0.3064	3.3539
6.4	8.5598	8.3163	0.1909	0.5888
6.5	15.1435	5.8211	0.6175	2.3743
6.6	0.3746	0.1474	0.0924	0.4236
6.7	1.2246	1.3367	0.2738	2.0250
6.8	8.9573	11.1860	0.4805	1.1386
6.9	3.5664	1.0158	0.3869	2.1943
6.10	3.9222	10.7667	0.1796	2.6485
7.1	1.1404	1.3617	0.1937	0.1319
7.2	1.4556	1.0154	0.3641	0.8621
7.3	2.8276	1.2937	0.6288	3.0311
7.4	11.7200	10.2213	0.3125	1.3510
7.5	14.0622	9.9867	0.7935	2.9535
7.6	0.7161	0.7197	0.1103	0.4890
7.7	1.1793	1.2143	0.2525	1.1993
7.8	7.5766	8.7358	0.4269	1.4182
7.9	3.4000	1.5888	0.5491	2.7081
7.10	5.4275	11.4737	0.2552	1.4715

8.1	1.4013	1.5394	0.1922	0.9687
8.2	1.5628	0.8735	0.3379	0.7283
8.3	2.4091	1.4462	0.2922	2.5502
8.4	9.4589	8.7982	0.1632	0.6951
8.5	18.6986	2.7092	1.3979	1.7405
8.6	0.8497	1.0472	0.1220	0.7080
8.7	1.2881	1.3030	0.2861	2.2901
8.8	7.7262	8.6303	0.5419	1.4810
8.9	3.6815	0.9461	0.4385	1.8645
8.10	5.3671	11.2899	0.2124	1.6187
9.1	1.4103	1.0553	0.1792	0.5648
9.2	1.6938	0.8342	0.3752	0.9454
9.3	2.1807	1.3529	0.3721	2.2878
9.4	8.7195	12.6624	0.1966	1.1622
9.5	16.2579	3.1135	0.6242	1.5787
9.6	0.5569	0.8788	0.1001	0.7092
9.7	1.0378	1.8657	0.2892	1.8483
9.8	9.0759	9.8636	0.4353	1.2433
9.9	3.0131	1.5191	0.4192	1.8392
9.10	4.9220	11.6182	0.1840	1.6684

附录二：源程序

1. 分立组件

1.1 语音信号的分帧加窗函数

```
function f=enframe(x,win,inc)
nx=length(x(:));              % 取数据长度
nwin=length(win);             % 取窗长
if (nwin == 1)                % 判断窗长是否为 1，若为 1，即表示没有设窗函数
   len = win;                 % 是，帧长=win
else
   len = nwin;                % 否，帧长=窗长
end
if (nargin < 3)               % 如果只有两个参数，设帧 inc=帧长
   inc = len;
end
nf = fix((nx-len+inc)/inc); % 计算帧数
f=zeros(nf,len);              % 初始化
indf= inc*(0:(nf-1)).';       % 设置每帧在 x 中的位移量位置
inds = (1:len);               % 每帧数据对应 1:len
f(:) = x(indf(:,ones(1,len))+inds(ones(nf,1),:));    % 对数据分帧
if (nwin > 1)                 % 若参数中包括窗函数，把每帧乘以窗函数
    w = win(:)';              % 把 win 转成行数据
    f = f .* w(ones(nf,1),:);  % 乘窗函数
```

```
end
```

1.2 连续多词汇的端点检测（调整参数后也可用于孤立词）

```
FrameInc = 200;%帧移为 80 点
amp1 = 12; amp2 = 6; %初始短时能量门限
zcr1 = 10; zcr2 = 2; %初始短时过零率门限
maxsilence = 8;  % 8*10ms  = 80ms
minlen  = 15;    % 15*10ms = 150ms
status  = 0;     %初始状态为静音状态
count   = 0;     %初始语音段长度为 0
silence = 0;     %初始静音段长度为 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);
%计算短时能量并调整能量门限
%amp = sum(abs(enframe(filter([1 -0.9375], 1, x), FrameLen, FrameInc)), 2);
amp = sum(abs(enframe(x, FrameLen, FrameInc)), 2);
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;
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');
 xx1=x1*FrameInc;
 xx2=x2*FrameInc;
 figure;
```

1.3 BP 神经网络的训练

```
%averageZCR
p11=[1.2443 1.6488 2.3390 9.5281 14.2167 0.9669 1.1132 8.3072 3.1890 9.2963];
p12=[1.4052 1.5148 2.8138 22.4638 16.1327 0.9476 1 9.1770 2.9623 3.2475];
p13=[1.2366 1.9171 2.7939 22.9348 17.3301 1.2647 1 8.9877 3.4702 7.9521];
p14=[0.8309 1.1702 2.7122 7.6762 15.9223 0.5090 1.0462 9.2607 2.4605 7.6136];
p15=[1.2439 1.2841 2.3333 9.8982 16.7043 0.8618 1.1505 9.4068 3.3289 6.0507];
p16=[1.1100 1.3706 2.1533 9.2130 13.5991 0.4799 0.9538 8.9453 2.9787 8.0339];
p17=[1.1906 1.7207 2.4692 8.5598 15.1435 0.3746 1.2246 8.9573 3.5664 3.9222];
p18=[1.1404 1.4556 2.8276 11.7200 14.0622 0.7161 1.1793 7.5766 3.4000 5.4275];
p19=[1.4013 1.5628 2.4091 9.4589 18.6986 0.8497 1.2881 7.7262 3.6815 5.3671];
p110=[1.4103 1.6938 2.1807 8.7195 16.2579 0.5569 1.0378 9.0759 3.0131 4.9220];
%divZCR
p21=[1.3505 0.9103 1.3590 8.3263 7.9907 1.2479 1.1852 8.9347 1.0430 13.3429];
p22=[1.2797 0.8551 1.4286 0.9796 5.6097 0.6832 1.0494 9.3700 1.3550 11.1266];
p23=[1.1457 0.6372 104730 1.0095 3.2966 0.8786 1.2651 10.3161 1.0630 18.0328];
```

```
p24=[0.6042 0.7054 1.4481 0.9487 4.1530 0.3148 1.4578 9.8894 1.5616 17.6111];
p25=[0.9402 0.7937 1.7870 11.0919 4.7537 0.9587 1.8158 11.1978 1.1350 15.7000];
p26=[1.0296 0.7647 1.2180 9.2864 15.8629 0.4444 1.9815 7.9453 1.1538 16.5556];
p27=[0.5551 0.6500 1.4318 8.3163 5.8211 0.1474 1.3367 11.1860 1.0158 10.7667];
p28=[1.3617 1.0154 1.2937 10.2213 9.9867 0.7197 1.2143 8.7358 1.5888 11.4737];
p29=[1.5394 0.8735 1.4462 8.7982 2.7092 1.0472 1.3030 8.6303 0.9461 11.2899];
p210=[1.0553 0.8342 1.3529 12.6624 3.1135 0.8788 1.8657 9.8636 1.5191 11.6182];
%averageAMP
p31=[0.1873 0.3280 0.2773 0.3013 0.5354 0.1445 0.2701 0.5300 0.4857 0.3985];
p32=[0.1997 0.3197 0.7225 0.2090 0.7248 0.1364 0.2441 0.8008 0.5772 0.1319];
p33=[0.2464 0.3746 0.4431 0.6242 1.8227 0.1988 0.2011 0.5340 0.4564 0.3143];
p34=[0.1465 0.2962 0.3617 0.1547 0.6962 0.1067 0.2160 0.5786 0.4288 0.2947];
p35=[0.1591 0.3133 0.3459 0.2012 0.8401 0.1235 0.2782 0.4687 0.5937 0.2749];
p36=[0.1790 0.3081 0.2498 0.2183 0.4948 0.1134 0.2455 0.5829 0.3940 0.3968];
p37=[0.1592 0.3108 0.3064 0.1909 0.6175 0.0924 0.2738 0.4805 0.3869 0.1796];
p38=[0.1937 0.3641 0.6288 0.3125 0.7935 0.1103 0.2525 0.4269 0.5491 0.2552];
p39=[0.1922 0.3379 0.2922 0.1632 1.3979 0.1220 0.2861 0.5419 0.4385 0.2124];
p310=[0.1792 0.3752 0.3721 0.1966 0.6242 0.1001 0.2892 0.4353 0.4192 0.1840];
%divAMP
p41=[0.9810 0.7205 2.7427 1.2051 3.2443 1.0699 2.0021 1.7876 1.6223 2.7968];
p42=[1.0089 0.7098 4.0127 1.4816 2.9363 0.6030 2.2308 2.0506 2.7955 2.3314];
p43=[1.9327 0.8661 3.1050 0.8885 2.7629 1.0206 2.3233 1.4584 1.7401 1.7821];
p44=[0.5711 0.6733 2.9685 0.8464 2.5151 0.4201 2.1607 2.1043 3.8245 2.1401];
p45=[0.5793 0.8509 3.6226 1.0085 4.3287 0.6735 2.9601 1.4281 2.2745 2.3136];
p46=[0.6663 0.7173 1.9603 0.8443 2.8040 0.4073 2.6844 1.5628 2.1311 2.5486];
p47=[0.5151 0.8416 3.3539 0.5888 2.3743 0.4236 2.0250 1.1386 2.1943 2.6485];
p48=[0.1319 0.8621 3.0311 1.3510 2.9535 0.4890 1.1993 1.4182 2.7081 1.4715];
p49=[0.9687 0.7283 2.5502 0.6951 1.7405 0.7080 2.2901 1.4810 1.8645 1.6187];
p410=[0.5648 0.9454 2.2878 1.1622 1.5787 0.7092 1.8483  1.2433 1.8392 1.6684];
%特征矩阵
p=[p11 p12 p13 p14 p15 p16 p17 p18 p19 p110;p21 p22 p23 p24 p25 p26 p27 p28 p29 p210;p31
p32 p33 p34 p35 p36 p37 p38 p39 p310;p41 p42 p43 p44 p45 p46 p47 p48 p49 p410];
pr=minmax(p);          %输入参数范围
goal = [eye(10,10) eye(10,10) eye(10,10) eye(10,10) eye(10,10) eye(10,10) eye(10,10)
eye(10,10) eye(10,10) eye(10,10)];
net=newff(pr,[3,10],{'logsig','logsig'});
net.trainParam.show = 10; %显示的间隔次数
net.trainParam.lr = 0.05; %
net.trainparam.min_grad = 1e-15; %最小执行梯度
net.trainParam.goal = 1e-5; %性能目标值
net.trainParam.epochs = 50000;%最大训练次数
net = train(net,p,goal);
y0=sim(net,p);
```

2.集成检测程序（仅包含孤立词识别）

2.1 KNN 算法识别

```
function [xx1,xx2] = voice_segment(x)
x = double(x);
x = x / max(abs(x));
%参数设置
FrameLen = 400;%帧长为 240 点
%input：训练和测试数据，存于 mat 文件中；output：KNN 算法的测试结果
function[]=KNN()
close all;
load('train_data_chinese.mat','total_last');
load('test_data_chinese.mat','total_last_test');
train_data = total_last;
test_data = total_last_test;
piecesPerclass=5;
class_num=10;
for i=1:class_num
   train_label(:,i)=zeros(1,piecesPerclass)+i;
end
X=train_data;%训练的特征，每行是一个特征向量，列数是特征的维数，行数是特征的个数
Y=train_label(:);
%训练生成模型-1NN
mdl =ClassificationKNN.fit(X,Y,'NumNeighbors',1);
characterClass= predict(mdl,test_data);  %得到最佳的分类。
characterClass=characterClass-1;
disp('k 近邻识别结果：')
disp(characterClass')
```

2.2 BP 神经网络识别

```
clc;clear;
% 录音录 2 秒钟
recObj = audiorecorder(44100,16,1);
disp('Start speaking.')
recordblocking(recObj, 2);
disp('End of Recording.');
myRecording = getaudiodata(recObj); %获录音结果
x = myRecording;
x = x / max(abs(x));%幅度归一化到[-1,1]
%参数设置
FrameLen = 256;      %帧长
inc = 90;            %未重叠部分
amp1 = 1 ;           %短时能量阈值
amp2 = 0.1;
zcr1 = 10;           %过零率阈值
```

```
zcr2 = 5;
minsilence = 6;  %用无声的长度来判断语音是否结束
minlen  = 15;    %判断是语音的最小长度
status  = 0;     %记录语音段的状态
count   = 0;     %语音序列的长度
silence = 0;     %无声的长度
%计算过零率
tmp1  = enframe(x(1:end-1), FrameLen,inc);
tmp2  = enframe(x(2:end)  , FrameLen,inc);
signs = (tmp1.*tmp2)<0;
diffs = (tmp1 -tmp2)>0.02;
zcr   = sum(signs.*diffs,2);
amp = sum((abs(enframe(filter([1 -0.9375], 1, x), FrameLen, inc))).^2, 2);
amp1 = min(amp1, max(amp)/4);
amp2 = min(amp2, max(amp)/8);
%开始端点检测
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); % 记录语音段的起始点
         status  = 2;
         silence = 0;
         count   = count + 1;
      elseif amp(n) > amp2 || zcr(n) > zcr2 % 可能处于语音段
         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 ||zcr(n) > zcr2     % 保持在语音段

         count = count + 1;
      else                      % 语音将结束
         silence = silence+1;
         if silence < minsilence % 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;              %记录语音段结束点
disp('length of effective voice sequence')
head = x1*inc; tail = x2*inc;
EVL = x2*inc-x1*inc;
y = x(x1*inc:x2*inc,:); % 得到经过端点检测剪切的语音序列
%计算过零率
tmp11  = enframe(y(1:end-1), FrameLen,inc);
tmp21  = enframe(y(2:end)  , FrameLen,inc);
signs = (tmp11.*tmp21)<0;
diffs = (tmp11 -tmp21)>0.02;
zcrm   = sum(signs.*diffs,2);
length(zcrm);
totalZCR = sum(zcrm);
averageZCR = totalZCR/length(zcrm)
%前后半程过零率之比
zcr11=zcrm(1:floor(0.5*length(zcrm)),:);
length(zcr11);
zcr12=zcrm(floor(0.5*length(zcrm))+1 :end ,:);
length(zcr12);
divZCR = sum(zcr11)/sum(zcr12)
%平均能量
ampm = sum((abs(enframe(filter([1 -0.9375], 1, y), FrameLen, inc))).^2, 2);
length(ampm);
totalAMP = sum(ampm);
averageAMP = totalAMP/length(ampm)
%前后半程能量之比
amp11=ampm(1:floor(0.5*length(ampm)),:);
amp12=ampm(floor(0.5*length(ampm))+1 :end ,:);
divAMP=sum(amp11)/sum(amp12)
%构造特征向量并带入检测
eigenvector=[averageZCR;divZCR;averageAMP;divAMP]
y_result=sim(net,eigenvector)
[a,b]=max(y_result);
disp('the number is')
b-1
```