数字信号处理实验报告

目　录

摘要

语音识别是信号处理领域一个十分重要的议题，声音的复杂多变和人耳的精巧结构都使其在技术上有着很大的挑战性，而利用大量语音作为模板进行训练和匹配也是被广泛采用的技术方法。考虑到 Mel 滤波器对人耳的模拟性，加上 KNN 算法在分类上的广泛应用，我们选择 MFCC 系数作为语音信号的特殊向量，并利用 KNN 进行分类测试。我们共采集了 1900 段语音信号并用其组建了多个不同的数据集，在 Matlab 平台上实现了特征提取、数据集组建和分类测试的功能。

我们在单人数据集上进行交叉验证，达到了 42.6%的识别准确率，并在平衡测试集上达到了 26%的识别率。

一、 语音识别技术简介

语音识别是信号处理中一个重要的研究方向，近二十年来，语音识别技术已经取得了显著进步，我们在工业、通信、消费电子产品等多个领域中都有着极为广泛的应用。然而语音识别的过程也相对繁杂，从音频信号的处理到特征提取再到分类往往需要进行大量的工作，我们小组希望从 0~9 数字语音信号分类入手，找到语音分类的一个高效的方法。

目前的语音分类技术多采用模式识别策略，在训练阶段提供大量的数据并将这些数据作为训练集提取特征，而在识别时将输入语音的特征与训练数据进行比对，从而确定输入语音的具体分类。声学特征提取是其中一个重要环节，因为原始信号是时域波形，不能很好地代表声音特征，目前常用的声学特征包括线性预测系数 LPC、Mel 滤波器倒谱系数 MFCC 等。

Mel 滤波器倒谱系数可以很好地从频域上描述语音的特征，而它相比较于直接使用语音 FFT 快速傅立叶变换能够更好地压缩特征量。Mel 滤波器在低频部分有密集的刻度，而高频时刻度稀疏，与人耳更容易识别低频信号这一点特征相符，也更好地模仿人耳对音频的识别。

KNN 是相对简单的分类方法，其方法的主要优势是训练过程计算量简单，在提取出数据特征之后即可进行分类，但其空间开销和测试时间随着数据集的增大就十分明显。我们认为 KNN 是一个比较合理的分类方法。

在本次实验中，我们提取了语音的 MFCC 系数并利用 KNN 对语音段进行了分类。我们将在第 2 节中详细阐述 MFCC 特征提取的具体步骤，第 3 节中讲述我们对于数据集准备的一些思考，关于分类算法我们则是放在了第 4 节中。为了使用方便和用户友好，我们开发了一个用户界面，这个将在第 5 节中进行简单介绍，而第 6 节中则给出了不同数据集和测试参数下的测试结果，并对算法性能进行一个总体的评估。

二、 特征提取

1. 加窗

因为我们采样到的一个语音文件通常包含了多个语音段，为获得每一个语音段单独的数据，我们进行了加窗的操作。实验指导书中的预加重操作是早于加窗操作的，但经过实际操作，我们发现在噪声比较大的片段中，先进行预加重会使噪声和真实语音之间的界限变得模糊不利于区分。因此我们先进行了加窗操作：计算短时能量，而后用双门限法加窗。

2. 预加重

由于语音的高频分量对于识别具有特别的意义，然而高频分量又通常能量较弱，因此需要预加重，此处我们使用的滤波器是$H(z) = 1 - az^{-1}$，a取 0.97，实际绘图验证，发现高频信号增强，但同时也很小幅度地增强了背景噪声，这也说明了预加重操作应晚于加窗。

3. 快速傅里叶变换

本实验想要提取的是频域上的特征，因此要计算出语音信号的离散傅里叶变换频率谱，在算法实现上使用快速傅里叶变换（FFT）算法，分别对每一帧数据做 FFT。

$$X(i,k) = FFT\ (x_i(m))$$

此处我们直接使用了 MATLAB 的 FFT 库函数，尽管我们也自行实现了 FFT 运算，但经过实际操作发现使用库函数能加快运算速度。

此处需要注意的是，从 FFT 这一步开始一直到最终得出 MFCC 系数，都是对每一帧语音信号分别来做的，最后可以用多维数组把提取到的特征存在一起。

4. Mel 滤波器

通过下式由实际频率计算 Mel 频率：

$$Mel(f) = 2595\lg(1 + f/700)$$

式中，f 为频率，单位为 Hz。

设划分的带通滤波器为 $H_m(k)$，$0 \le m < M$，M为滤波器的个数。每个带通滤波器的传递函数为：

$$f(x) = \begin{cases} 0, & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)}, & f(m-1) \le k \le f(m) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)}, & f(m) \le k \le f(m+1) \\ 0, & k > f(m+1) \end{cases}$$

其中，$\sum_{m=0}^{N-1} H_m(k) = 1$ 。

Mel 滤波器的中心频率$f(m)$定义为：

$$f(m) = \frac{N}{f_s} F_{Mel}^{-1}(F_{Mel}(f_l) + m\frac{F_{Mel}(f_h) - F_{Mel}(f_l)}{M+1})$$

5. Mel 滤波能量

计算每一帧 FFT 后的数据谱线能量：$E(i,k) = [X_i(k)]^2$

把求出的每帧谱线能量通过 Mel 滤波器，并计算在该 Mel 滤波器中的能量。

$$S(i,m) = \sum_{k=0}^{N-1} E(i,k)H_m(k), \qquad 0 \le m < M$$

实际上此处可以忽略一些频率较高的滤波窗，因为实际频率与 Mel 频率的对数关系是从约 1000Hz 开始的，而当频率远远大于 1000Hz 时滤波器的影响可以选择忽略。

6. 经离散余弦变换（DCT）得到 MFCC 系数

$$mfcc(i,n) = \sqrt{\frac{2}{M}} \sum_{m=0}^{M-1} ln[S(i,m)]cos[\frac{\pi n(2m-1)}{2M}]$$

此处我们原本取 MFCC 阶数为 12，而后在老师建议下尝试了 14。由这个式子计算，相当于从每一帧数据中计算出 14 个 MFCC 向量，这些向量即为这一帧的语音特征。

三、 数据集准备

我们将处理过的语音存储作为实验的数据集，每段语音我们存储每一帧的 1~14 阶 MFCC 系数作为每一段语音的特征，则每一段语音的特征为一个 n*m 的矩阵，其中 n 表示这一段语音的帧数，而 m 表示 mfcc 系数的最高阶数，在我们的程序中取的是 14。为方便程序调用，我们再将不同段语音的 mfcc 系数放在一起，以语音类别和语音片段索引作为高两维，使数据最终采用类似于四维数组状的表示方法。

由于不同段语音的帧数长短有所区别，会导致每一段语音的 MFCC 系数矩阵大小不一，而在高维上无法进行拼合，因此我们选用 cell 元胞来存储，每一个元胞里面是一段语音的 MFCC 系数，而把若干个元胞组成一个 n*m 的矩阵，其中 n 是语音类别数，此处为 10，m 是每类语音所包含的数量。

由于语音信号需求量大，而采集成本相对较高，我们与其他小组交换了语音数据，并通过前文所示的算法进行了特征提取。考虑到每个人在发音习惯、音色上的差别，我们认为要想更好地识别出每个人所说的数字，需要较为多样的语音，因此我们组内在采集数据时注重了语音数量的平衡。加上与其他小组的交换，我

们共有 1900 组语音数据，其中每种数字各 190 组，我们设计了如下几种数据集进行了效果比对。

1. 单人语音数据集。我们将单人的语音单独作为一个数据集，以观察不受音色和讲话人干扰的情况下语音识别的情况。我们取了三个人的语音做成了三个单人数据集，每个人有 10*20 段语音片段。
2. 全部数据集。我们将所有的语音数据片段均作为训练集。由于需要留 200 个片段作为测试片段，所以我们训练集中仅有 1700 段语音片段。当然，当我们进行自交叉测试的时候，就会直接使用全部 1900 段语音片段。
3. 平衡数据集。由于在与其他组交换语音时，存在有数据不平衡即某一个人贡献了大部分的语音的情况。为防止某个人的音色或者发音过分影响数据集的效用，我们平衡了每个人的语音在数据集中所的比例，制成了一个 1100 段语音构成的数据集，其中每个人的片段数均在 100~200 之间，多余的片段当作测试集处理。

我们同时在这三个数据集上进行测试，以期望观察不同的数据集对于分类结果所产生的影响。

四、 分类算法

我们使用 KNN 算法进行语音段的分类。KNN 算法，即 K 最近邻分类算法通过比对测试数据与训练集中每一个样本的相似度来进行分类。我们的程序以 mfcc 系数作为特征，通过动态时间规划（DTW）算法获得测试样本与训练样本的距离，选择距离最近的 k 个点中的众数作为识别结果。

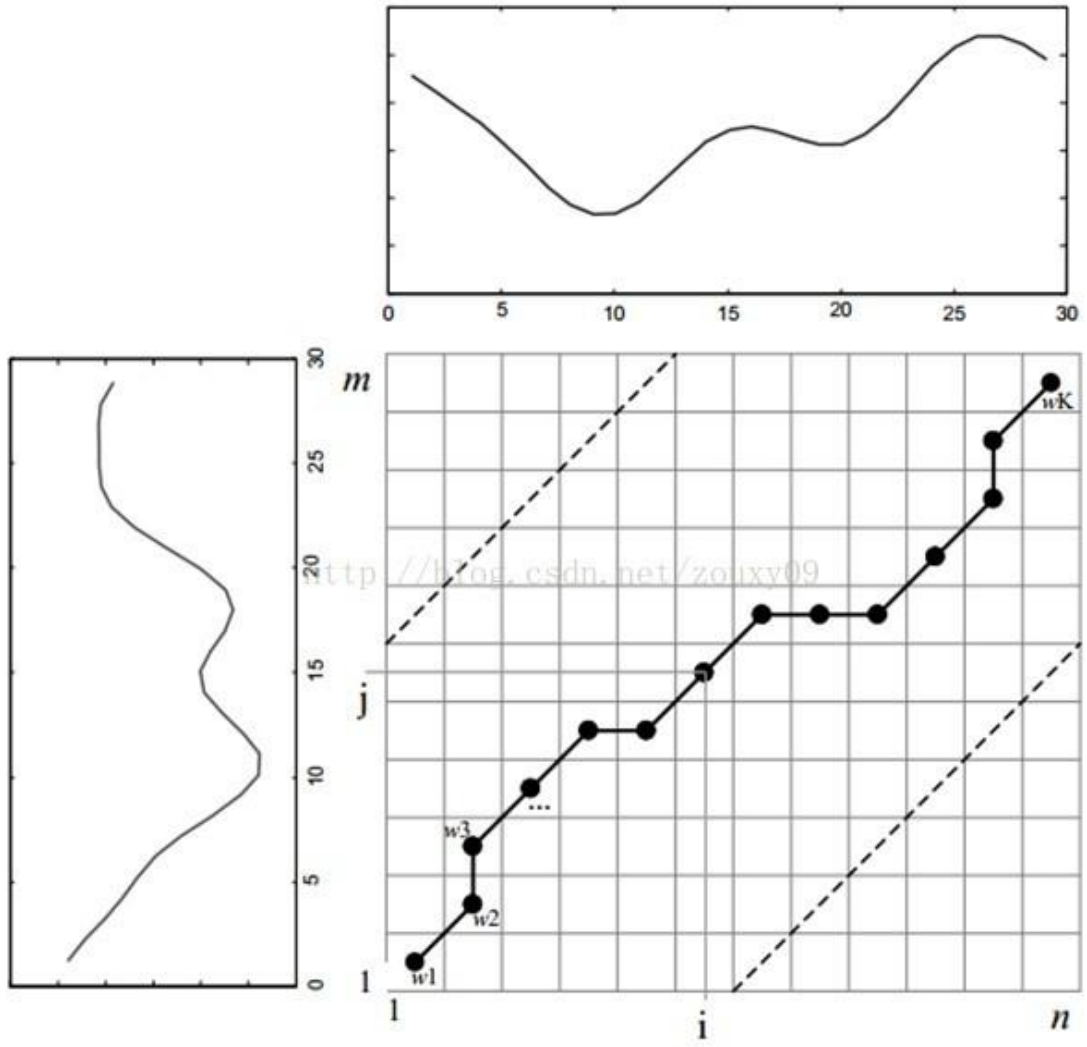
1. 算法 DTW

要进行 KNN 分类，首要的一点是对两组数据相似性的计算。对于定长向量之间的相似性可以直接用差向量的模作为两个向量之间的距离，而我们提取的变量尺寸是根据语音长度不同有所变化的，所以我们需要一种其他的方法来定义特征之间的距离。这里我们采取动态时间规划 DTW 算法。

我们认为每一阶 MFCC 系数向量是一个特征，我们需要计算两个不同语音段不同阶 MFCC 向量之间的距离然后把它们做加和作为总的向量距离。每一阶 MFCC 系数向量长度不一，通过将较短的一个向量进行邻近插值较长的向量从而与另外一个同长度的向量计算距离。选取的方法是使选取出来的向量与另一个向量有尽可能小的常规意义上的距离。我们定义第一个向量 a 前 i 个分量和第二个向量前 j 个分量之间的距离为：

$$d(i,j)=\begin{cases} D(i,j)+min\{d(i-1,j),d(i,j-1),d(i-1,j-1)\} & ,i>0 \ \ and \ \ j>0 \\ 0 & ,i=0 \ \ or \ \ j=0 \end{cases}$$

如下图，每一格上代表两个向量对应位的距离，DTW 即画出一条从图的左下角到右上角的路径，使这条路径经过的距离之和最短。在语音片段距离中可以认为是在任何的局部拉伸压缩变换所能得到的两段语音的距离最短值（差异最小值）。



2. KNN 参数

KNN 中唯一的参数是 k，也即对于每一段测试语音我们我们在实现过程中，考虑到了 KNN 中参数 k 的取值不同可能对实现结果产生不同的影响。我们在测试过程中对于 k 的取值不同进行多次测试，并给出相应结果。

3. KNN 评价与改进

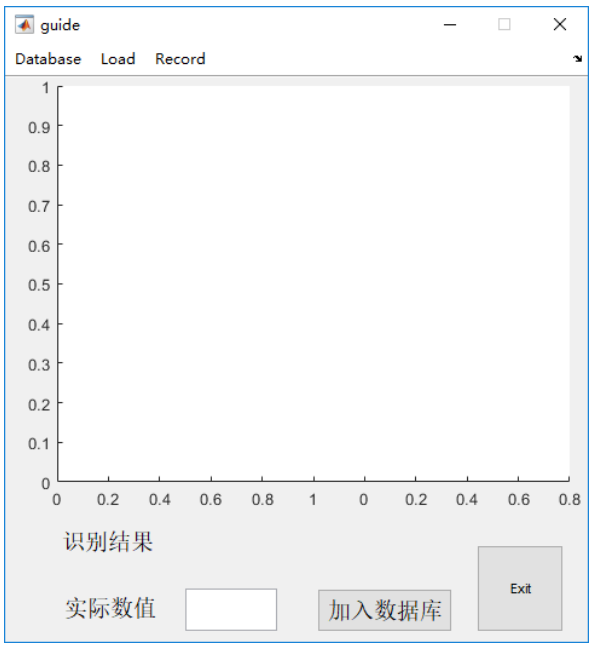
我们选择 KNN 算法有以下几个原因：结构较为简洁，易于理解，易于实现，无需估计参数，无需训练。但在实际使用过程中，我们也发现了 KNM 算法的存在的一些问题，如当样本不平衡时，容易得到容易得到样本容量较大的结果，这个问题可以通过平衡样本数量解决；另一个不足之处是计算量较大，因为对每一

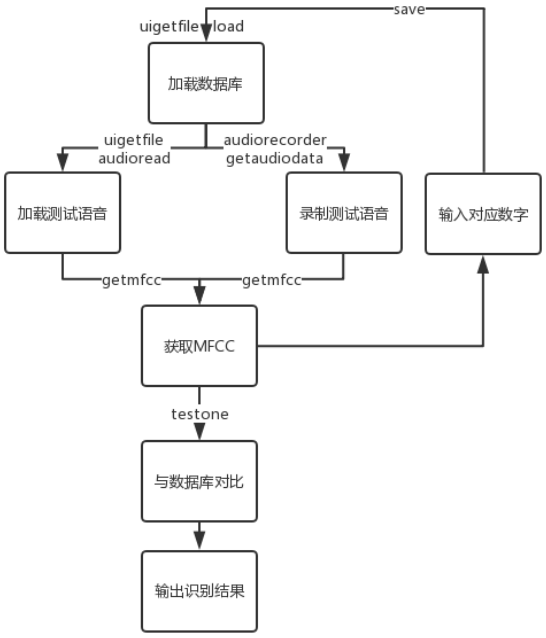
个待分类的文本都要计算它到全体已知样本的距离，才能求得它的 K 个最近邻点。

对此，我们以分类效率作为改进方向，事先对样本属性进行约简，删除对分类结果影响较小的属性，快速的得出待分类样本的类别。因此我们分别求出每个样本到其他同类样本的距离并求和，我们认为这代表了其在同类样本中的代表性。然后取那个距离之和最小的样本数据，如此便得到了 0-9 每个数字的 1 个典型样本，并以此进行识别。

五、 图形用户界面设计与实现

为方便用户使用，我们计划开发一个界面友好的程序，而最常使用的方法就是使用图形用户界面 GUI。建立这样一个界面友好、占用资源少、高性能、便于移植、可配置的 GUI 界面设计，能够使用户的学习和使用更为方便容易。用户不需要知道后台的应用程序究竟是怎样执行各种命令的，而只需了解可见界面组件的使用方法；只要通过与界面交流就可以使指定的行为得到正确的执行，对输入的信息进行一系列的处理。因此我们利用 Matlab 设计了“基于 mfcc 的语音识别”的 GUI 界面，能够方便直观地将我们输入的数据及识别的结果很好地表达出来。我们设计的界面极其逻辑流程如下：





可以看见，在菜单上有 Database、Load 和 Record 三个按钮，通过 uigetfile 函数获取文件名，再由相应的函数完成文件的读取，分别实现了加载数据库，加载测试语音段进行识别和录制语音进行识别的功能。在程序的主体界面中，有一个坐标系，可以用来显示加载的数据库或语音的波形。在录制或加载完语音后，将调用 testone 及 getmfcc 函数获取识别结果并在窗体上显示出来。

由于一开始我们的样本集较小，我们希望我们的程序有不断学习的能力，

因此还加入了"加入数据库"这一按钮，在录制完语音后，可以手动输入对应的数字并保存到我们的数据库中，以此不断扩充我们的数据库，提高识别率。

六、 测试结果

1. 近邻分类测试

我们利用 KNN 近邻分类器对语音进行分类测试，主要测试 KNN 分类参数以及不同数据集对分类效果的影响。我们取 KNN 分类器的 k 分别为 5、7、11 以查看不同近邻数量对分类器性能的影响，同时观察在不同数据集上的表现。

首先我们对单人数据集进行测试，我们在单人数据集上进行交叉测试。单人测试集内有 10*20 段语音，我们每次随机选择 10*1 段语音当作测试集，其余 10*19 段当作训练集，并通过 KNN 分类器进行分类，每次共对 100 段语音进行分类并输出分类结果。我们取 train_c 和 train_x 两个单人数据集进行交叉测试，KNN 取 5、7、11。测试结果如下：

测试集	K 取 5 时准确率	K 取 5 时准确率	K 取 11 时准确率
Train_C	39．0%	37．6%	42．6%
Train_X	35．6%	39．3%	38．9%

然而如果我们用一个人的语音作为训练集对另外一个人的语音进行测试，可以得到如下的测试结果。此处我们用 train_c 作为训练集，而 train_x 作为测试集。共进行了 200 次测试

数字	0	1	2	3	4	5	6	7	8	9	总
K=7	0%	0%	75%	0%	0%	80%	0	0%	0%	40%	19．5%
K=11	0%	0%	80%	0%	0%	55%	0%	0%	0%	65%	20%

显然如果用一个语音来对另外一个人的语音进行分类是不合适的，因为每个人的语音特征是不同的，这个结果是我们所有数据集中最差的分类结果，一方面说明了特征提取需要具有充分性，也说明训练数据需要有广泛性。

然后我们观察其在平衡数据集上的测试结果。我们取了同一个人的 10*20 段语音作为测试集。我们观察其在 k 取 5、7、11 时的分类效果。

数字	0	1	2	3	4	5	6	7	8	9	总
K=5	55%	0%	0%	70%	15%	45%	0%	10%	15%	0%	21%
K=7	60%	0%	0%	75%	20%	50%	0	5%	15%	5%	23%
K=11	60%	0%	0%	80%	20%	55%	0%	15%	25%	5%	26%

我们可以看出，平衡测试集比单人测试集有更差的测试结果，我们认为这是由于各人音色给训练产生了较大的干扰。同时我们也看出识别结果对于不同数字有很明显的偏差，这也是各个数字发音的特性所决定的。但是依然可以看出，k 取值增大能显著改善分类效果。

最后我们使用全数据集进行测试。使用全数据集时我们随机选取了这 1900 段语音中的 200 段作为测试集而其他作为测试集，设置 k 值分别为 5、7、11，查看分类结果如下：

数字	0	1	2	3	4	5	6	7	8	9	总
K=5	45%	35%	5%	35%	0%	20%	0%	25%	20%	10%	19．5%
K=7	45%	35%	0%	35%	0%	20%	5%	35%	15%	10%	20%
K=11	50%	50%	0%	35%	5%	20%	5%	40%	5%	15%	22．5%

和之前的测试一样我们发现 K 增大能够显著地提升分类效果。但是通过和之前的对比我们也发现完全数据集的分类效果没有平衡数据集好。我们认为该数据集可能过拟合于其中贡献数据量较大的人的音色特点，而无法专注于对数字读音的分类，从而使系统的性能有所下降。实际上此分类数据集中有超过一半的语音是同一个人贡献的，也因此我们发现了数据平衡对于分类的重要性。

虽然前面的测试结果都表明增大 k 的取值能显著改善分类性能，但是增大 k 的取值会增加时间和空间开销，所以我们需要在效率和效果之间进行折中，取得对最佳的分类方案。

2. 典型分类测试

正如前文所言，KNN 分类算法在计算距离时会花费大量的时间，因为它每次都要和所有训练样本进行比对。如果希望简化运算量的话，必须要减少训练集中的数据量，因此我们采用了 4.3 节中的方法，生成典型数据来进行分类。

首先是对于同一个人的 100 个样本使用该方法，识别结果如下，综合正确率约为 48%，与未进行简约分类样本时结果相近，但运行速度大幅提高。

数字	0	1	2	3	4	5	6	7	8	9
准确率	30%	50%	20%	50%	90%	80%	40%	60%	30%	30%

但对完全数据集中的 1900 个样本简约数据集进行识别时，识别的正确率却明显的降低了，只有 19.63%，我们推测的原因是对于每个数字 190 个样本而言，仅保留 1 个中心样本使得删除的样本过多，而这个样本不足以很好地代表这个数字的类域，而且这批数据中包含有个人音色等信息量太大，典型样本难以学习。

为提高其学习能力我们改为对 1900 个样本取每个数提取 10 个典型样本并重新进行了测试，识别的正确率提高到了 24.9%，得到了与之前相近的结果但大大提高了执行效率。

数字	0	1	2	3	4	5	6	7	8	9
每个数取 1 个中心点/%	18.9	21.1	2.6	58.9	24.2	27.4	4.7	12.6	25.8	0.0
每个数取 10 个中心点/%	16.3	33.7	8.9	57.4	17.9	26.3	7.9	15.8	36.8	27.9

七、 总结

在本次实验中，我们利用 MFCC 系数和 KNN 分类器对语音信号进行了分类，并集成成了一个应用阶段的用户应用。我们比对了不同的数据集和 knn 参数，发现采用既具有广泛性又在各种数据来源中找到平衡的平衡数据集能有最好的表现。同时，k 越大分类效果越好，但是却也带来了更大的时间和空间开销。典型数据分类方法能减少大部分的资源占用，但是却对分类结果有很大影响。

我们认为我们的分类算法在时间和准确率上都有很大的提升空间，还没有办法做到实时地给出分类结果，而这在需要及时性和准确性的时候就有很大的弊病。我们的数据集大小也需要扩充，每种类别只有 190 条语音数据量过少不足以完全学习。我们组建测试集时没有用训练集之外的人员的语音进行测试也是漏洞之一，这些都是在日后的实践中需要改进的方面。