Audio Environment Classification for Hearing Aids

by

Cecille Freeman

A thesis
presented to the University of Guelph
in fulfilment of the
thesis requirement for the degree of
MSc.(Eng)
in
Engineering Systems and Computing

Guelph, Ontario, Canada, 2009

I hereby declare that I am the sole author of this thesis.

I authorize the University of Guelph to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Guelph to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

The University of Guelph requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

This thesis examines background classification systems for hearing aids in four stages. In the first stage, the K-nearest neighbours classifiers (KNN), hidden Markov models (HMM), artificial neural networks (ANN) and ANNs with windowed input (WANN) are assessed for functionality. All of these classifiers appear to be suitable for this task. However, the WANN gives the best results. In the second stage K-means clustering and self-organizing maps (SOMs) are used to find appropriate classes. The classes selected are in-car, traffic, birds, water washing, water running, office noise, restaurant noise, shopping and music. In the third stage feature selection is used on a large candidate set from literature. Feature selection is done using sequential forward floating search (SFFS) and using Euclidean distance as the distance metric. In the final stage, the classifiers are tested using the classes and feature vector selected. These are tested against a simpler feature vector from literature. The simple feature vector is able to give better results that the selected feature vector in most cases. The WANN has an average accuracy of 65.6% and a best-run accuracy of 80.0% using a sample window size of three. Using a selected nine-feature vector, the ANN is able to attain an average accuracy of 68.2% and a best-run accuracy of 74.4%. Better results may be able to be attained through the use of a different distance metric. The ANN and the WANN both appear to be the good classifiers for this task.

Acknowledgements

A wise and experienced grad student once told me that he was unsure how one could thank all those who required thanking in an acknowledgements section that was only a few paragraphs long. And now that I have come to write this section, I, too, declare that it cannot be done. So I give up all pretence and I will be taking the space I need to do it right. What are a few more pages tacked onto this massive tome? And to those who would say this section is too fat, to you I say, it's just big-boned.

First and foremost I would like to thank my advisors, Dr. Robert Dony and Dr. Shawki Areibi. This thesis would have been a horrendous mess were it not for your support, direction and editing. Dr. Dony, your advice was always helpful, your corrections and suggestions well explained. Every time I came to you for help or editing, I always felt that what I came out with was an improvement over what I had before. And although I seemed to come quite a lot, your door was always open, whether it was for thesis work, class work, or any of my various other problems, which never seemed to be in short supply.

Dr. Areibi, thank you not only for the help with my thesis, but also for being a fantastic person to TA for, and for being a great instructor. I always looked forward

to you classes, even if I tried my best never to actually speak in them.

Thanks also to everyone else who worked on this project. Ahmed and Shaw, your feedback was always helpful and your support was very much appreciated. To everyone at AMI, thanks very much for all of your support, for attending all of our meetings and providing excellent feedback on all of our work. It is always nice to know that there's a chance that this work won't live alone in the library forever.

Not everyone who requires thanking was so directly involved in the project. Thanks especially to everyone in the engineering office. Lucy, Mary and Susan, you are all so knowledgeable and friendly, you make everything so much easier.

I have always thought that one of the best things about engineering at Guelph is the professors. We have many incredible profs that we have at this university, but several have really stood out. To Dr. Brown, an excellent research co-op term where I started to learn how to put together a real research project. You were always so supportive and calm, even as I was causing major ecological disasters in the enviro lab, mere hours from your exam. Thanks also to Dr. Calvert, who isn't actually in the engineering school, but as I've been told many times, we're not all perfect. You are a great instructor and I learned a lot in your class, much of which I used either directly or indirectly in this thesis. Many thanks to Dr. Davidson as well. I never actually had you as a professor, but every time I came to you with a problem you were there. So many of the great opportunities I had were due to your guidance or influence. You have opened more doors than I even knew were there.

Thanks also to all the profs I have TA'd for: Dr. Moussa, Dr. Lubitz, Dr. McBean and Eyad. Every class was a fantastic experience, and I learned a lot

from every one of you. Thanks especially to Dave McCaughan, for making my first TA experience so enjoyable, both inside and outside the classroom. I learned a lot about teaching in that class, and, actually, a lot about PHP too.

None of my TA positions would have gone as smoothly were it not for the other TAs I have worked with. Thanks to Darshna for being awesome during a crazy semester. Thanks also to Christina and John for great semesters and for being my CUPE buddies. Thanks to awarren for never minding that I can't stop calling by his email, and for helping me get my feet wet in my first TA semester. Thanks especially to Patterson for being a great TA, but also a great friend. Together with Fisher and Gus, you guys form the nerdiest group of crazies I know and I love it.

To my amazing office roomies, Zoe and Jiggar for putting up with me all that time. You guys are fantastic. Thanks also to the folks on the GES: Carlos, Maeghan, Ang, Maryam, and Sarah for all your hard work and also for just generally being great. Also, thanks to all the other amazing grad students for always being there for me, especially when I wanted to procrastinate. Anna, Jeff, Mo and Heather, you guys area amazing. Thanks especially to Alicia and co. for all of our curricular and extra-curricular good times, and to Danny, instigator of all things fun.

Also, thanks to Tej both for being the best first-year TA ever, and for being the first brazen person to just let the acknowledgement section grow as it may. Your teachings are wise, great lifelong-student master. Thanks also to the other crazy grad students John and Mattheson, and Al, their partner in crime. Many good times were had by all.

I also owe a special thanks to all the people who helped me on the way to this degree. My undergrad friends were great, from helping me understand classes I didn't quite get, to simply being there for me when I was stressed, I couldn't have done it without you guys. To everyone in the cluster and whom I sat with on engsoc, you all made every day enjoyable, and a special thanks to Baxter, Driver and Jeannine, the craziest of the crazies. Also thanks to Kim, for introducing me to the crazy circus that is student government. It was incredibly strange, but also fun.

To Ramy, my most organized friend. You always seem to know exactly what you want and exactly how to get it. And yet, instead of viewing people as competition, you managed to pull people up with you. Thanks also To Graver, for always being there for me, right from the beginning. You always challenged me to stretch myself and I for that I am very grateful.

Many thanks to my great roommates. Darryl, not only are you fun, but your crazy work ethic is actually scary. Thanks especially to Marcy, Meghan and Morri. You were the best roommates ever and always put up with my craziness. I can't possibly imagine what school would have been like without you guys. You are all amazing and my sappy acknowledgements section cannot possibly do justice to your mighty awesomeness.

Jon, an extra-special thanks to you for putting up with my absolute insanity during a stressful 41x semester and all through my grad degree. From my .NET inspired sailor mouth to my many MATLAB-induced mood swings, it's a miracle you arrived unscathed, and yet you were always there at the end of the day strategically

ducking and ready to help.

Last, but certainly not least, thanks to my awesome family. Lily and Tooty, you've never been anything but supportive and I'm sure you're well aware that I would never have gotten this far without your help. Thanks to my slightly nuts, but still awesome brother, Jake. You're like my clone and my exact opposite all wrapped up into one. And thanks to Mom and Dad, for all your support these seven long years. I know you thought I was crazy at times, but although you were occasionally reduced to nodding and smiling, you were always interested and ready to help with everything both school-related and otherwise. Your help and support is very much appreciated.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Hearing loss affects more than one million Canadians [1] and the prevalence of hearing loss increases with age [2]. Health Canada suggests that more than 30% of Canadians over the age of 65 are affected by hearing loss, making it the most common sensory impairment for people in this age category [3]. Health Canada also suggests that many teenagers and young adults are also starting to suffer some permanent hearing loss due to exposure to excessive noise [4].

There are three major types of hearing loss: sensorineural, conductive and mixed. Conductive hearing loss occurs when the transmission of sounds from the outer ear canal to the inner ear is blocked. Sensorineural loss occurs from damage to the inner ear or the auditory nerves. Mixed hearing loss is a combination of sensorineural and conductive. While conductive loss can usually be corrected medically or surgically, sensorineural loss is permanent and cannot be corrected surgically [5]. Additionally, conductive hearing loss is often flat across the spectrum, or has more loss occurring in the lower frequencies [6]. This is easier to correct because it can be

solved by simple amplification. Sensorineural hearing loss, however, is often worse in the higher frequencies. Since correcting conductive loss is trivial, most current hearing aid research focuses on sensorineural hearing loss.

Hearing loss is a problem that is quite prevalent, and is likely going to become worse as our exposure to excessive noise increases. Unfortunately, many people with hearing loss either do not know about their loss or choose not to wear corrective devices for their loss. There are a number of reasons for this, including social factors; however, one of the most common reasons is the perception that hearing aids are not effective or can actually make hearing more difficult.

Patients with hearing aids often complain of a number of different problems. One of the most common is difficulty hearing in areas with background noise. The amplification of hearing aids works for both foreground and background noises, which can make hearing foreground noises more difficult in noisy environments. Additionally, people with hearing loss already require a higher signal to noise ratio (SNR) to hear correctly, and the combination of these two factors can greatly reduce the ability to actually distinguish sounds [7]. Using noise reduction techniques can also cause problems with distinguishing speech, as some techniques can also eliminate speech cues. This results in an increase in SNR, but no increase in intelligibility [8, 9].

As algorithms and processing techniques become more complex, more and more signal processing systems are moving towards digital signal processing. Digital signal processing offers a number of advantages over traditional analog processing techniques. Because digitally implemented filters can be fitted more closely to

the peaks and valleys required in the frequency spectrum for each patient, the speech could be made more crisp. Digital filters are more easily adjusted, which would allow the use of a single, general filter bank that can be customized for each patients frequency loss. Digital filters are also able to handle more complex algorithms, such as adaptive and statistical filtering, which can be used to decrease noise and feedback and enhance speech signals.

Digital hearing aids can easily handle and switch between the multiple data inputs required for microphone array and binaural processing (see Section 2.1.3), and handle the multiple programs and functions required for environment specific programming and environment classification (see Chapter 3).

Research on manual multi-program hearing aids has shown that patients benefit from using different hearing aid program in different locations [10]. However, these require the user to adjust the hearing aid manually, usually with a small switch on the hearing aid. This can be difficult for elderly patients who may also suffer from reduced dexterity due to arthritis or other conditions. This also requires the user to first identify that the hearing aid is not working properly and test different programs to find the correct settings. Using an automatic adjustment system would be much easier for users and would result in fewer incidents where the user was unable to properly hear. It would also allow the hearing aid to incorporate more complex adjustable systems, such as incorporating multiple noise reduction programs for different noise environments. An example of a noise reduction system using an environment classifier is shown in Figure 1.1. In this diagram, the dashed box indicates the portion of the system that is being investigated in this thesis.

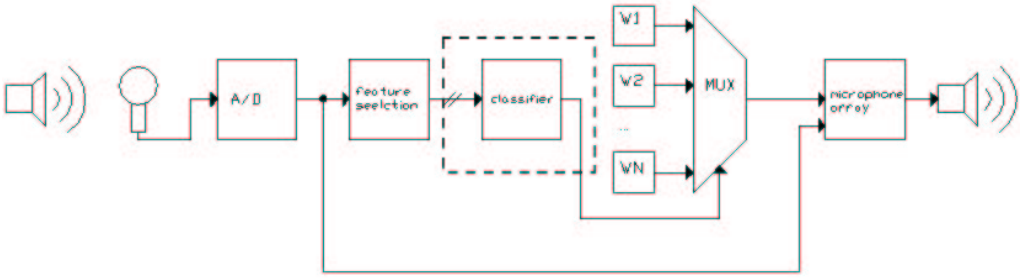


Figure 1.1: An illustration of a sample microphone-array-based noise reduction system that could be used in a hearing aid. The dashed box indicates the portion of the system that is considered in this thesis, including the selection of the classifier itself as well as the inputs and the outputs

In order for the hearing aid to adjust automatically, an audio environment classification system is required, which can identify the current audio environment from the incoming noise. This thesis investigates audio environment classifiers for hearing aids. The thesis examines the type of classifier that should be used, as well as the audio features that should be used for input and different types of audio environments that should be classified in this system. The classifiers tested in this thesis are the K-nearest neighbours classifier, the multi-layer perceptron and the hidden Markov model. Additionally, this thesis investigates the effect of using a multi-layer perceptron with windowed input. This technique has been used for prediction applications, but has not been applied to the classification of audio samples.

Although a number of researchers have examined different classification algorithms for audio environment classification and different related problems, few have used a formal feature selection technique to select a set of input features. This thesis uses a formal feature selection technique to select the inputs from a large set of candidate features from literature. Additionally, the majority of the studies in

this area simply assume the audio classes that are used as outputs. Some studies have logically justified the choices, but no studies have examined the output classes in a formal manner. This thesis investigates the types of output classes using a clustering technique.

The remainder of this thesis is structured as follows. Chapter 2 presents some background information on hearing aids and the algorithms used in this research, including information on pattern classification algorithms, clustering algorithms and feature selection algorithms. Chapter 3 presents a literature review of recent environment classification studies including suggested classifiers, classes and input features. Chapter 4 presents and discusses the methods and results from the initial tests of the classifiers for functionality. Chapter 5 presents and discusses the methods and results used to select a set of audio environments to use as output classes. Chapter 6 presents the approach taken to audio feature selection and presents and discusses the results. Chapter 7 discusses the methods used to test the entire system together and the discusses the results of the tests. Chapter 8 presents the conclusions and recommendations from this thesis.

Chapter 2

Background

Hearing impairment is characterized not only by reduction in loudness at different frequencies (attenuation loss), but also by a difficulty in separating signals in closely spaced frequencies. Hence, patients with hearing loss require a higher SNR than a person with normal hearing in order to achieve the same level of intelligibility [11, 7]. In a study of hearing impaired patients with hearing aids, the largest complaint was the difficulty of hearing in areas with background noise [7]. To compensate for this, noise reduction techniques are often implemented in hearing aids.

This chapter will give an overview of the current techniques used for noise reduction. It will also present the basic algorithms for the techniques used in this thesis, including the algorithms for the classifiers, clustering and feature selection.

2.1 Filtering

Many different noise reduction techniques have been investigated, but no one technique is able to perfectly reconstruct a signal in all situations. Some filtering techniques are non-adaptive systems, where the characteristics of the filter are unchanging. Others are adaptive filters that adjust their weights based on the incoming signal. Both of these techniques have their own advantages and disadvantages. Several of these techniques are discussed next.

2.1.1 Non-adaptive filters

Non-adaptive filters do not change as the incoming signal changes. This is a simpler design than an adaptive filter, and does not suffer from instability, which can occur in adaptive systems. However, since non-adaptive filters do not change with time, the noise reduction ability of the filter may vary greatly as the external environment changes.

Automatic gain control (AGC) is a fairly simple noise reduction technique. AGC adjusts the gains of the frequency bands by comparing the statistics of the signal in each band to the statistics of pure noise in that band [12]. Unfortunately, although AGC is able to increase the overall SNR of the signal, it does not actually improve intelligibility as it also removes important speech cues from the signal [9].

Spectral subtraction is a technique that is similar to AGC. The statistics of noise are estimated and then the noise spectrum is subtracted from the output signal. Unfortunately, spectral subtraction also removes speech cues, again improving the SNR but not the intelligibility [13].

An alternate to filtering based on the frequency characteristics is to use modulation filtering. Since speech usually has much more amplitude modulation than noise, the gain in a band can be adjusted based on the amount of signal modulation. However, this type of a filter tends to leave artifacts, called "musical noise", which is not desirable in a hearing aid. [14].

A Wiener filter is an optimal filter and can be used to separate noise and speech [15]. When tested for hearing aids, Wiener filters were found to only improve intelligibility for half the hearing impaired patients, and actually decreased intelligibility for all non-hearing impaired subjects [13]. Clearly, this filter is not an ideal solution since half the users would receive no benefit, and the signal may actually become less intelligible for users with little hearing loss.

Whitmal and Rutledge propose a noise reduction system based on wavelets. The method improves intelligibility for flat-loss subjects, but does not seem to benefit patients with sloped loss [16]. This technique does not appear to be appropriate for hearing aids, as most flat-loss subjects suffer from conductive hearing loss, which is normally caused by some type of a physical blockage in the ear. This type of a loss can be corrected through surgery or by simple volume amplification [5]. Most hearing aid users have sloped hearing loss, therefore this technique would not benefit the majority of users.

2.1.2 Adaptive filters

Adaptive filters can theoretically give a better performance in more situations, as their weights can be adjusted to better match the external noise environment.

However, they do have several disadvantages. Firstly, it is possible for an adaptive filter to become unstable [17]. Secondly, adaptive filters do not tend to work well in reverberant areas (areas with echo), where the desired signal can be reflected and form part of the background noise. Accordingly, adaptive filters are prone to signal cancellation in reverberant areas [18, 13].

The Widrow-Tsypkin self-adaptive filter has been used in hearing aids [9], but requires prior knowledge of the noise signal. Its performance also degrades considerably when the noise and signal come from the same direction [9]. A single-microphone extension of the Widrow-Tsypkin approach is the Sambur filter [9]. This filter, however, makes the assumption that speech is highly correlated and noise is uncorrelated. This assumption does not hold for unvoiced consonants, which are important speech cues. It also does not hold for correlated noise, which occurs in reverberant areas. This filter is therefore not an appropriate choice for a hearing aid [9].

Graupe, Basseas and Grosspietsch [9] propose a different type of self-adaptive filter, called the Zeta filter, which is based on time rather than frequency. The system seemed to improve intelligibility in laboratory tests [9].

Magotra et. al [19] also developed an adaptive noise reduction system called a "real-time adaptive correlation enhancer (RACE)". This filter was found to improve speech discrimination ability in patients [19].

Although these noise reduction systems seem promising, neither study specifically mentioned the performance in reverberant areas. Unfortunately, adaptive filters do not tend to perform well in reverberant areas [18, 13].

2.1.2.1 Summary

Table 2.1 lists the advantages and disadvantages of the examined noise reduction systems. It is clear that noise reduction is a complex problem. Currently no noise reduction technique exists that is robust enough to handle all noise situations required in a hearing aid, although some techniques are promising in certain situations. Current research is looking less to a single filter solution and more towards multiple microphone or multiple environment solutions. These types of solutions are discussed further in sections 2.1.3 and Chapter 3 respectively.

2.1.3 Microphone Arrays

Many common noise reduction techniques are able to increase the SNR of a signal but do not improve intelligibility because the filters also remove speech cues. Microphone array processing is one of the more successful techniques thus far because they are able to preserve speech cues. The standard noise reduction techniques mostly rely on frequency domain processing and make assumptions about the frequency spectrum of noise. Conversely, microphone arrays filter sounds spatially, making the assumption that the signal is coming from a specific direction. Noise can be modeled and subtracted from the signal by filtering spatially based on phase [21]. This increases the SNR of signals from the look direction, usually assumed to be the front. Most importantly, microphone arrays do not seem to have the same intelligibility problems as frequency based noise reduction techniques [18].

A study by Hamacher [22] looked at the delay-subtract beamformer and compared it to monaural and binaural spectral subtraction filtering techniques. The

Technique	Adaptive?	Advantages	Disadvantages
High pass [20]	no	•simple	•not all noise is low frequency •can remove speech components
AGC [9]	no	•common technique for gain	•does not improve intelligibility
Modulation [14]	no	•improves SNR	•leaves 'musical noise' artifacts
Spectral subtraction [13]	no	•improves SNR	•does not improve intelligibility
Wiener filter [13]	no	•very common filter	•does not work for all users •decreases intelligibility for normal hearing patients
Wavelets [16]	no	•improves SNR	•only improves intelligibility for flat loss patients
Widrow-Tsypkin [9]	yes	•well tested	•requires prior knowledge of the noise signal •performance degrades when signal and noise are from the same direction
Sambur [9]	yes	•improves SNR	•assumptions do not hold for correlated noise or unvoiced consonants
Zeta [9]	yes	•improved SNR and intelligibility	•no tests with reverberation
RACE [19]	yes	•improves speech discrimination	•no tests with reverberation

Table 2.1: Advantages and disadvantages of different noise reduction techniques

study found that the binaural spectral subtraction was more effective than simple monaural spectral subtraction, but that a delay-subtract approach was more effective than the simple filtering techniques [22].

Kates has authored or co-authored a number of papers studying the superdirective beamformer [23, 18, 24]. In [23], Kates compares the superdirective beamformer to the delay-and-sum beamformer, and found that the superdirective array was able to give an additional 5 dB of array gain over the delay-and-sum. The superdirective beamformer is recommended in all three publications.

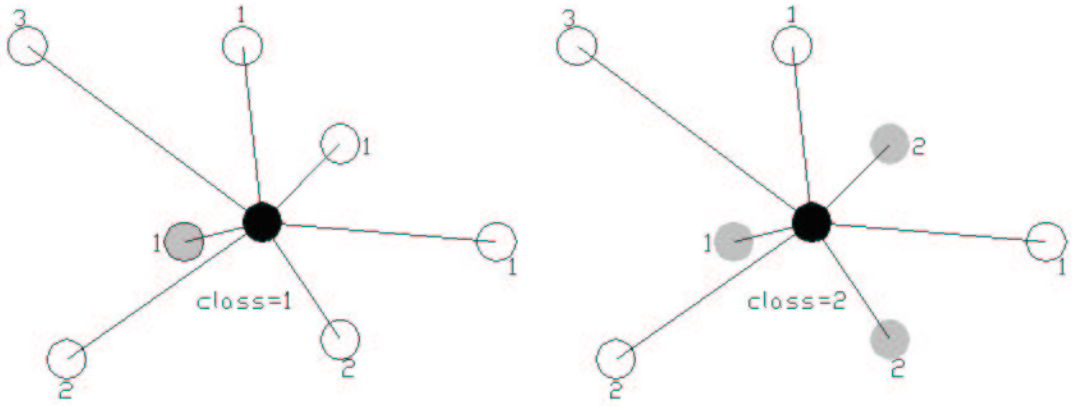
Wittkop and Hohmann [25] found that beamformers had difficulty dealing with diffuse noise. To combat this problem, they developed a simple environment classification system to determine if there was diffuse noise in the signal and adjust the beamformer accordingly [25]. This is an interesting application of environment classification. Although Wittkop and Hohmann suggest using the classifier to adjust a pre-processor, this idea of combining a classifier with a beamformer could easily be applied to the beamformer itself, using an environment classifier to change the weights in a beamformer, rather than using a fully adaptive system. This would reduce the instances of signal cancellation found in adaptive system, but would still allow the beamformers to use the environment specific weights required in some beamformers. A number of weights could be pre-programming for the different environments and a set of weights could be selected based on the classifier output.

2.2 Classification Techniques

Audio environment classification aims to determine the type or class of a certain background noise, from several different possible background noise types. As discussed in [25], it is possible to combine microphone array processing with an environment classification system to improve the performance. Using an environment classifier also allows the use of more than one static set of weights in a beamformer. This allows some differentiation between different sound environments, without using a fully adaptive system.

Many of the more sophisticated classification techniques are intelligent systems that formulate rules or patterns for the classification through machine learning. Supervised learning techniques use a set of training vectors consisting of the input vector and a class designation to train the classifier. Alternately, unsupervised techniques do not require that the input vectors be pre-labelled, but instead create classes from the patterns in the data itself. Supervised techniques are more common for classifiers, which require that input vectors be designated as part of a pre-set class, whereas unsupervised techniques are more common for clustering algorithms, which aim to determine the possible classes in a set.

In this thesis, several classification techniques are examined: K-nearest neighbours (K-NN), multi-layer perceptrons (MLP), hidden Markov models (HMM) and MLPs using windowed input (WMLP).



a) 1-NN b) 3-NN

Figure 2.1: 1-NN and 3-NN implementations of the K-NN algorithm. In the 1-NN the class assigned is the class of the closest training vector. In the 3-NN the class assigned is that of the majority of the three closest training vectors.

2.2.1 K-Nearest Neighbours

The K-nearest neighbours classifier uses a relatively simple technique to determine the class of a new input. The training vectors are simply stored in the classifier, instead of being used to adjust the model. For every new input, the distance between the input vector and each of the training vectors is calculated using a suitable distance metric, usually Euclidean distance. The class of the new input is determined as the majority class of the K closest training vectors. For example, for a 1-NN system, the class of the input vector is the class of the closest training vector. For a 3-NN system, the class of the input vector is the class of the majority of the three closest training vectors. This is illustrated in Figure 2.1. In the case of a tie, the class is recursively calculated using $k-1$ points, until a class is found. This is illustrated in Figure 2.2.

The K-NN classifier is very simple, but is computationally expensive. Please see

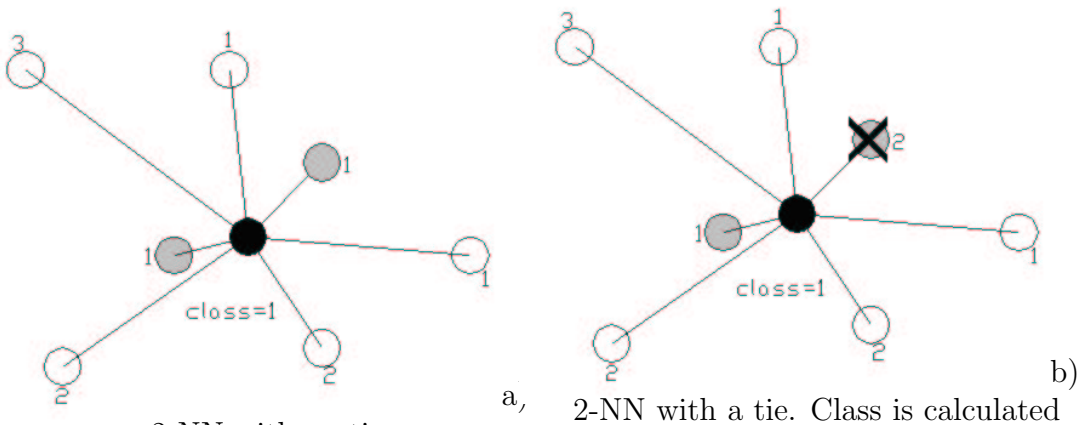


Figure 2.2: 2-NN classifier with and without a tie situation. When there is no tie (a), so the class assigned is the class of the two closest vectors. When there is a tie (b), the furthest vector is eliminated and the class is calculated from the closest $K-1$ vectors.

Section 7.2.1.3 for further discussion. If too few points are used, the classes are not properly represented by the training set. However, as the number of training vectors increases, both the computation time and the memory requirements increase. This type of classifier is more suited to simple classification problems with relatively few training vectors.

2.2.2 Multi-layer perceptrons

Multi-layer perceptrons are computational models that mimic the behaviour of biological nervous systems, which are based on the firing of neurons. A neuron receives excitory and inhibitory input in the form of neurotransmitters. If the signal is above a certain threshold, the neuron sends a signal to other neurons via its axon. The neurons are highly interconnected, and the overall signal is determined by the output of more than one neuron at a time.

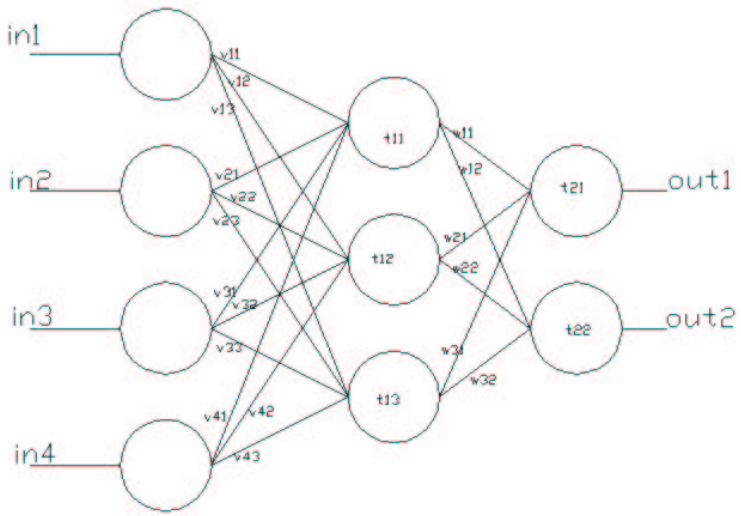


Figure 2.3: Fully-connected three-layer feedforward neural network. The parameters V_{ij} are the weights from the input layer (i) to the hidden layer (j), W_{jk} are the weights from the hidden layer (j) to the output layer (k), T are the threshold values.

In a computer system, the neuron is represented by a mathematical function. The excitory and inhibitory systems are represented by positive and negative inputs from the other nodes in the network. The inputs are weighted and summed to represent the incoming neurotransmitter action. In a biological cell, if this value passes a certain threshold value, then the neuron would transmit, otherwise it would not. In a computer system, the threshold value is simply added to the input and the value is passed through a sigmoid output function. A feedforward network assembles the nodes in layers. The input layer is connected to the input vector, with one node for each of the N input terms. The output layer is used to indicate the class of the input vector. Hidden layers are between the input and output layers and the nodes each layer are connected through a weight matrix. The architecture used in this thesis is a fully-connected model, as illustrated in Figure 2.3.

In this thesis, the MLP is trained using backpropagation. This is a supervised learning technique that uses the error at the output to adjust the weight and threshold values of the nodes. The error term is propagated backwards through the network to adjust the thresholds and weights of the hidden layer. First the input is propagated forward through the network to determine the output with the current weights. During the testing phase, only the forward phase is completed. The value at the input node is simply the value of the input vector. The values of the hidden and output nodes are calculated as [15]:

$$b_j = f_\beta \left(\sum_{i=1}^{N} a_i w_{ij} + t_{bj} \right) \tag{2.1}$$

where b_j is the j^{th} node in the current layer, a_i is the i^{th} node in the previous layer, w_{ij} is the weight from a_i to b_j, N is the number of nodes in the previous layer, t_{bj} is the threshold for b_j and f_β is the sigmoid activation function. This formula is used to calculate the values of each node in each layer. The values of the nodes in the output layer give the calculated class.

The error at the output node is calculated from the difference between the calculated and desired outputs as [15]:

$$e_k = b_j(1 - b_k)(b_k^d - b_k) \tag{2.2}$$

where e_k is the error at node k in the output layer, b_k is the calculated value at node k, and b_k^d is the desired value of node b_j.

The error at the hidden layers is calculated from the output layer error as [15]:

$$e_i = b_i(1 - b_i)\left(\sum_{j=1}^{N} w_{ij}e_j\right) \tag{2.3}$$

where e_j is the error at node j in the hidden layer, b_j is the calculated value at node j, w_{jk} is the weight from the current hidden node to node k in the output layer, and e_k is the error at node k in the output layer.

Once all the error terms are calculated, the weights and thresholds can be adjusted based on the error as [15]:

$$\Delta w_{ij}(n) = \alpha b_i e_j + \eta \Delta w_{ij}(n-1) \tag{2.4}$$

where Δw_{ij} gives the change in the weight going from node i in one layer to node j in the next, b_i is the value at node i, e_j is the error at node j and α is the learning rate constant, which is a positive constant between zero and one that is used to control the rate of learning. A smaller constant will take longer to train but give more accurate results; a larger constant trains more quickly, but is less accurate. The parameter η is a positive rate constant in the range of zero to one that is used to determine the impact of the previous change on the current change.

The threshold values can be adjusted as [15]:

$$\Delta t(n) = \alpha e_j + \eta \Delta t(n-1) \tag{2.5}$$

where Δt gives the change in the thresholds, t is the threshold value at the current

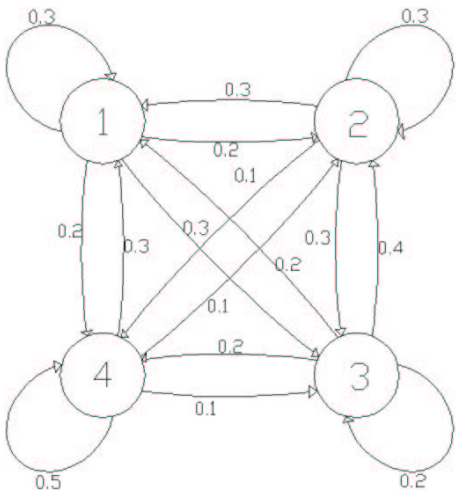


Figure 2.4: four-state, fully connected Markov model represented as a graph

node, α is the learning rate constant, and e_j is the error at the current node.

The weights are adjusted after each vector, and the training is stopped after a certain number of epochs of training. Once the training is terminated, the performance can be assessed on a test set using the final weights and thresholds.

2.2.3 Hidden Markov Models

Hidden Markov models are stochastic signal models that are based on Markov chains. A Markov chain is a discrete, stochastic process that has the Markov property, which dictates that the current state of the system is dependent only on the current inputs and the previous state [26].

Markov models can be represented as graph representing the states as nodes, and giving the transition probabilities between each state as the directed edges on the graph. An example of a Markov model graph is shown in Figure 2.4.

In the regular Markov model, the states are visible, and the probabilities can

be determined exactly using the state transition matrix. A hidden Markov model is an extension of the regular Markov model, but the actual states are hidden and cannot be accessed directly. Instead, the model generates observable sequences that can be used to estimate the state of the system. The model is extended to include observation probabilities within each of the states. The probability of seeing a particular observation is the product of the probability of being in a certain state and the probability of that state generating that particular output, summed across all the possible states. It is not possible to generate a definite probability of being in a certain state, as multiple states can generate the same observations, and an observation sequence can be generated using many different state chains.

Hidden Markov models are characterized by a number of different parameters. These parameters are listed and explained below:

1. N is the number of states in the model. These are hidden, but remain a part of the model.

2. Q is the number of possible observations. Because each observation has a probability associated with it, the number of possible different observations is finite. This is also called the codebook size.

3. $A = \{a_{ij}\}$ is the state transition matrix, where a_{ij} is the probability of a transition from state i to state j

4. $B = \{b_j(O_t)\}$ is the probability observation matrix, where $b_j(O_t)$ is the probability of having observation O_t in state j.

5. $\pi = \{\pi_i\}$ is the initial state probability matrix, where π_i is the probability of being in state i at time $t = 1$.

The entire model can be represented by these properties. As a short form, the model can be represented as:

$$\lambda = (A, B, \pi) \tag{2.6}$$

where λ represents the entire model [26].

The input to the HMM is a set of input vectors from time $t = 1$ to $t = T$. The initial state probability is π. At each t the model changes from state to state based on the probabilities described in matrix A. As the states change, the model generates observations based on the probabilities described in B. The probability of the model generating a certain observation sequence can be calculated from these values. An input is classified by training one HMM for each possible class. The class of an input is the class of the HMM with the highest probability.

There are three phases associated with training an HMM [26].

1) Phase 1 - Determining the probability of a certain set of observations being generated by a certain model$\lambda = (A, B, \pi)$ $(P(O|\lambda))$

2) Phase 2 - Finding the most probable state sequence, given a certain model $\lambda = (A, B, \pi)$ and an observation sequence

3) Phase 3 - Training the model to better match the desired output sequence, attempting to maximize $P(O|\lambda)$

Each of these three phases has a well-known solution that is used in this thesis. The first phase is solved using the forward-backward algorithm, the second phase is

solved using the Viterbi algorithm, and the third phase is solved using the Baum-Welch algorithm.

2.2.3.1 Forward-Backwards Algorithm

The first problem is fairly simple to solve, as it just requires the multiplication of probabilities and no readjustment. This problem is solved using the 'forward-backwards' algorithm. This algorithm estimates the forward probability, starting at time $t = 1$ with the initial probabilities π. The forward probability is a joint probability of generating the observation sequence $O = \{O_1, O_2, \ldots, O_t\}$ up to the current time t and being in a certain state at time t [26].

The forward probabilities are calculated in two stages. The α_1 values are first initialized as [26]:

$$\alpha_1 = \pi_i b_i(O_1) \tag{2.7}$$

The remaining α values are calculated recursively as [26]:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) \tag{2.8}$$

where N is the number of states in the model, a_{ij} is the transition probability from state i to state j and $b_j(O_{t+1})$ is the probability of state j generating the observation O_{t+1}.

The probability of the model generating the entire sequence of observations can be calculated by summing the α values at time $t = T$. Each $\alpha_T(i)$ is the probability of generating the entire observation sequence and being in a certain state. The sum

of the α values across i, gives the probability of generating the entire sequence and being in any final state [26].

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.9}$$

The backwards probability is the probability of generating the observation sequence from $t = t + 1$ to the end time $t = T$. The backwards probability is the joint probability of being in a certain states at a certain time and having a certain observation sequence from time $t = t$ to $t = T$.

The backwards calculation starts from time $t = T$ and progresses backwards. The β_T values are initialized to 1, and the remaining β values are calculated recursively as [26]:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \tag{2.10}$$

2.2.3.2 Viterbi Algorithm

No exact solution exists to the second problem, since the states are hidden. The Viterbi algorithm attempts to find the single best state sequence, by maximizing $P(Q, O|\lambda)$, where Q is the state sequence.

The Viterbi algorithm makes use of a new variable that can be used to explain the optimization criteria. The probability of being in state i at time t given the observation sequence and the model is given as $\gamma_t(i)$ and is determined as [26]:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \tag{2.11}$$

since the product of $\alpha_t(i)$ and $\beta_t(i)$ gives the probability of being in state i at time t and having the entire observation sequence and $P(O|\lambda)$ is the probability of the observation sequence given the model.

A new variable, δ, is defined to track the highest probability paths. The δ value for a certain time t and certain state i gives the highest possible probability along a single path that ends at state i at time t. Mathematically [26]:

$$\delta_t(i) = \max_{q_1,q_2,\ldots,q_t} P(q_1 q_2 \ldots q_t = i, O_1 O_2 \ldots O_t|\lambda) \tag{2.12}$$

Another variable, ψ is used to track the actual state that had the maximum value tracked by δ. The δ and ψ values are calculated recursively, starting at $t = 1$. Once all the values are calculated, the best state sequence is backtracked from the final value, using the states in ψ.

2.2.3.3 Baum-Welch Algorithm

Adjusting the model is actually the most difficult of the three phases, and even the widely used Baum-Welch method does not guarantee finding a globally optimal solution, as it only performs local optimization. Another variable, ξ is defined for the Baum-Welch algorithm. This variable describes the probability of being in state i at time t and being in state j at time $t + 1$ as [26]:

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \tag{2.13}$$

This variable relates to the γ value as [26]:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{2.14}$$

since γ describes the probability of being in state i at time t.

The π values can be re-estimated directly from γ_1 since both describe the expected number of times the system is in state S_i at time $t = 1$ [26]:

$$\overline{\pi}_i = \gamma_1(i) \tag{2.15}$$

Summing γ over t gives the expected number of transitions from state i over all t, and summing ξ over t gives the expected number of transitions from state i to state j. These values can be used to re-estimate the A terms, since A is the state transition matrix, and is simply the probability of transitioning to another state from a known state. The new A values are calculated as [26]:

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{2.16}$$

The B probabilities give the probability of observing a certain value when in a certain state. Hence, the B probabilities are re-estimated as the expected number of times the system is in a certain state and observing a certain value, divided by the total number of times it is in that state. The new B values are calculated as [26]:

$$\overline{b}_j(k) = \frac{\sum_{t=1}^{T-1} \gamma_t(j), \text{ S.T. } O_t{=}v_k}{\sum_{t=1}^{T-1} \gamma_t(j)} \tag{2.17}$$

2.3 Clustering Techniques

Clustering is the processes of organizing items into groups whose members are related in some way. Clustering is related to classification in that both divide unknown inputs into classes. However, whereas classification aims to put input vectors into pre-defined classes, clustering aims to determine appropriate classes based only on the input data. Because of this, clustering techniques often use unsupervised training techniques.

K-means clustering is a relatively simple algorithm that aims to create K clusters from the data set by grouping data vectors that are close together and separating vectors that are far apart. Unfortunately, the user is required to determine how many clusters should be used. The self-organizing map creates a more continuous map, but does not distinguish individual clusters.

2.3.1 K-means Clustering

K-means clustering works by dividing samples into the closest of a number of randomly placed clusters, and then adjusting the placement of the clusters based on the samples in the cluster until they converge.

The system is initialized using K random centroid values that represent the centres of the clusters. For each input vector, the distance is calculated between the vector and each of the centroids. In this thesis, Euclidean distance is used (see Section 2.4.2). The vector being classified is assigned to the cluster with the closest centroid. Once each of the vectors is assigned, the centroid of each cluster is updated to be the mean of all the vectors in the cluster. The algorithm is stopped

```
Initialize N centroid values randomly
while(change>threshold){
  for each input vector i{
    min = MAX_FLOAT
    argmin = 0
    for each centroid j{
      distance = Euclidean distance (i,j)
      if(distance < min){
        min = distance
        argmin = j
      }
    }
    assign vector i to cluster argmin
  }
  change = 0
  for each centroid j{
    oldval = j
    j=average of all inputs assigned to j
    change = change + abs(oldval-j)
  }
}
write out all input vectors in each cluster
```

Figure 2.5: Pseudocode for K-mean clustering algorithm

once a certain threshold is reached. Pseudocode is presented in Figure 2.5.

This algorithm is logically very simple, but can produce different results in different runs because of the random initialization. Additionally, because the K value must be determined before the program is run, the K-mean process does not give any information about how many clusters are actually appropriate for the system.

2.3.2 Self-organizing Maps

A self-organizing map (SOM) is a type of neural network that uses unsupervised learning. The SOM gives a spatial representation of the input data in two dimensions. This shows the general relationship between different samples, and does not require that the number of clusters be predetermined. A SOM contains many nodes organized in either a square or a honeycomb pattern. The value of each node is a vector that is the same size as the input vector. These values are initialized randomly, and the training of the map is based on the distance between the node and the input.

The learning algorithm is a competitive algorithm. First the distance is calculated between the input vector and each node. Any distance measure can be used; however, the Euclidean distance is very commonly used for the SOM, and it is the distance measure used in this thesis (see Section 2.4.2). The node with the smallest distance to the input vector is then used as the base of the update. The closest node is called the best matching unit (m_c). The best matching unit and the nodes in a defined neighbourhood (N_c) around the best matching unit are all updated to bring them closer to the input vector, and all other nodes are left unchanged. The update is performed as follows [27]:

$$m_i(t+1) = \begin{cases} m_i(t) + \alpha(t)[x(t) - m_i(t)] & \text{for } i \in N_c \\ m_i(t) & \text{for } i \notin N_c \end{cases} \tag{2.18}$$

where m_i is the value of node i, $x(t)$ is the input vector, N_c is the neighbourhood and α is the adaptation gain, where $0 < \alpha < 1$ and alpha decreases with time.

The neighbourhood variable can also be changed with time. The training of the SOM is performed in two separate stages. The first stage is the ordering stage. In this stage, the neighbourhood variable is large, so that a large number of nodes are changed. These global scale changes affect the placement of the different regions in the map and define how different inputs relate to each other on the SOM. The second stage of training is the convergence stage. In this stage, both the neighbourhood variable and the α value are kept low. This has the effect of making small changes to a local area, changing how similar nodes are related, without making large changes to an area's placement on the map [27].

Once the iterations are complete, each node is labelled with the class of the closest training vector, which produces a continuous map showing the relative closeness of each of the classes. The algorithm does denote discrete groups, but groups can be selected logically from the map.

2.4 Feature Selection Techniques

Feature selection is used to determine which features can best be used to separate the data, given a set of candidate features. The technique used to perform the feature selection is the sequential forward floating search (SFFS). This is a more general version of the sequential forward search (SFS) and plus-l minus-r selection techniques [28]. In [28], Zongker and Jain compare the SFFS algorithm to SFS, generalized SFS, plus-l minus-r and Min-max algorithms, as well as a branch-and-bound. They found that the SFFS algorithm gives results that are better than the SFS, generalized SFS and plus-l-minus-r and comparable to branch-and-bound but

much faster. The backwards versions of the search are also tested and the results are comparable. Hence SFFS is used for this thesis.

SFFS requires a measure of the significance of each of the features, which is done using Fisher's interclass separability criterion and also using Euclidean distance.

2.4.1 Fisher's Interclass Separability Criterion

The ideal subset of features maximizes the inter-class variation (variation between different classes), but minimizes the intra-class variation (variation of the samples within a single class). Fisher's interclass separability criterion is a measure of both the inter and intra-class variation. It is measured as [29]:

$$J = trace[(Q_b + Q_w)^{-1} Q_b] \tag{2.19}$$

where J is Fisher's criterion, Q_w is the within-class scatter, and Q_b is the between class scatter. The parameter Q_w is defined as [29]:

$$Q_w = \frac{1}{CN} \sum_{j=1}^{C} \sum_{i=1}^{N} (w_{ij} - M_j)(w_{ij} - M_j)^T) \tag{2.20}$$

where C is the number of classes, N is the number of samples in each class, w_{ij} is the vector containing the feature values for the i^{th} sample in class j, and M_j is the vector containing the mean value for each feature within class j.

Q_b is defined as [29]:

$$Q_b = \frac{1}{C} \sum_{j=1}^{C} (M_j - M)(M_j - M)^T \tag{2.21}$$

where M is the vector containing the mean value for each feature between all classes.

2.4.2 Euclidean Distance

The Euclidean distance between two vectors A and B in Euclidean Space of size N is calculated as:

$$\sqrt{\sum_{i=1}^{N}(a_i - b_i)^2} \tag{2.22}$$

2.4.3 Sequential Forward Floating Search Procedure

The sequential forward floating search algorithm is a bottom-up search method that extends the sequential forward search (SFS) algorithm. The SFS algorithm is a sub-optimal search algorithm that was introduced as a way to perform feature selection without using a full search. However, the SFS procedure suffers from nesting problems, where features selected in one stage cannot be later unselected. The SFS algorithm was later generalized to the Plus-l-Minus-r search methods, which corrected the nesting problem of the SFS algorithm. However, there is no theoretical way to determine the l and r values. Instead, the SFFS algorithm allows these values to change or "float" throughout the training [30].

Two separate papers by Pudil et. al [30] and Zongker and Jain [28] have looked at the SFFS algorithm and compared it to other search methods. Both papers found that the SFFS algorithm and its backwards counterpart the sequential backward floating algorithm (SBFS) give approximately the same results. Additionally, both papers found that the SFFS gave results that were very close to optimal and took less time to train than other tested methods.

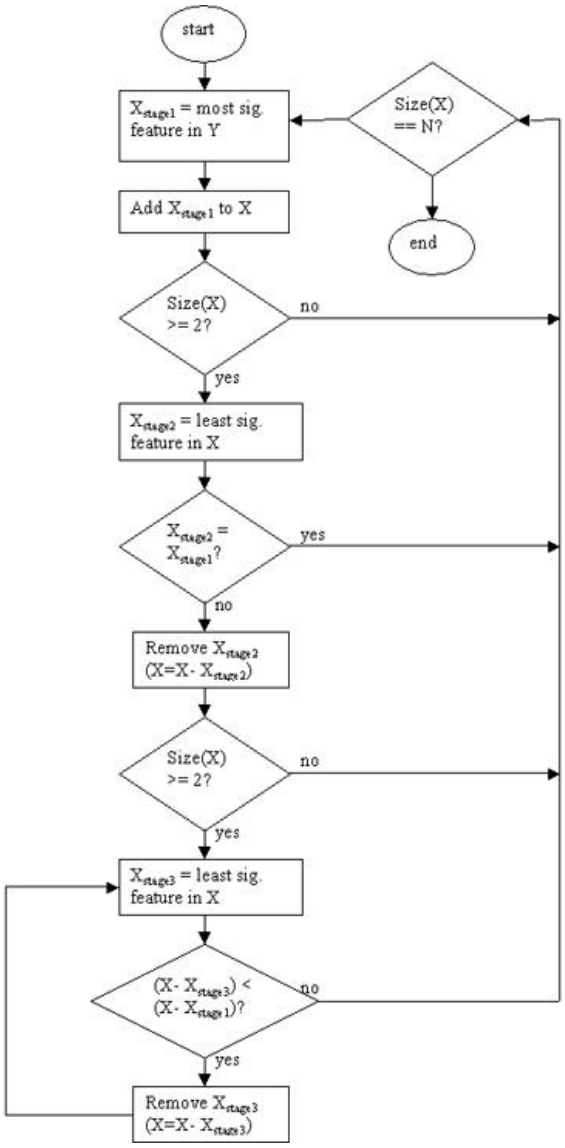


Figure 2.6: Flow chart illustrating the sequential forward floating search algorithm

The SFFS algorithm is illustrated in Figure 2.6. The full set of features is denoted as Y, and X_k is used to denote the set of k selected features. The algorithm proceeds in three stages. The algorithm starts with an empty feature set $X_0 = \emptyset$. In the first stage, the most significant feature with respect to X_k in the set $Y - K_k$ is added to form the set $X_{k+1} = X_k + x_{k+1}$. This stage is repeated until $k = 2$. The significance of a feature f_j in the set $Y - X_k$ with respect to the set X_k[30]:

$$S_{k+1}(f_j) = J(X_k + f_j) - J(X_k) \tag{2.23}$$

In the second stage, the least significant feature in X_{k+1} is found. From the criterion, the significance of a feature x_k in the set of features X_k is defined as [30]:

$$S_{k-1}(x_j) = J(X_k) - J(X_k - x_j) \tag{2.24}$$

If this is the feature that was added in stage one, it is kept and the algorithm returns to the first stage. If it is not, the feature is removed from the set to form set X'_k. If $k = 2$ then the algorithm returns to the first stage. If not, it proceeds to stage three.

In stage three, features continue to be excluded from the set. Again, the least significant feature is found. The algorithm first calculates $J(X'_k - x_s)$ and $J(X'_k - x_{stage1})$, where x_{stage1} is the feature added in stage one, and x_s is the least significant feature in the set X'_k. If $J(X'_k - x_s) < J(X'_k - x_{stage1})$ then x_s is eliminated otherwise the algorithm goes back to stage one. After the elimination, if $k = 2$ then the algorithm returns to stage one. If not, it continues with stage three.

The algorithm stops when the algorithm enters stage one and k is equal to a pre-determined number of features.

2.5 Summary

This chapter gave an overview of the current techniques used for noise reduction. It also presented the basic algorithms for the techniques used in this thesis.

Microphone arrays appear to be the most promising technique for noise reduction. Many microphone arrays, and some other noise reduction systems use the signal autocorrelation in their calculations. Hence, the classes selected in this work are selected based on the signal autocorrelation.

The classifiers used in this thesis are the K-nearest neighbours, the multi-layer perceptron and the hidden Markov model. Clustering techniques are used to pick a candidate set of classes and the algorithms used for this are the self-organizing map and K-means clustering. Feature selection is used to pick a good set of features for the classifier to use, and the algorithms used for this is the sequential forward floating seach. This is tested using two different distance metrics: Euclidean distance and Fisher's interclass separability criterion.

Chapter 3

Literature Review

3.1 Introduction

Many of the noise reduction techniques presented in Chapter 2 rely on knowledge of the spectral shape of noise, or similar characteristics. Similarly, many microphone arrays use information about the correlation of the noise to set the weights for the beamformer. Unfortunately, different noise situations have different frequency spectra and correlations, and a single noise reduction technique may not work well in all situations. A common solution to this problem is to use adaptive noise reduction techniques; however, adaptive systems are subject to signal cancellation in reverberant areas [31]. Another possible solution is to use pre-set, non-adaptive weights and switch between these weights by using an environment classification system.

It has been shown in [32] that different hearing aid users have different preferences for their hearing aid settings, and these preferences change as the listening

environment changes. There is a measurable increase in intelligibility when multiple programs are used and, given the opportunity, hearing aid users will use several different programs [33, 10]. Clearly, there is a precedent and a need for automatic switching hearing aids.

In a 1993 study [34], Ringdahl shows that patients with hearing aids use a average of 3.4 ± 1.1 different programs for at least an hour each day, and 4.3 ± 1.2 programs for at least one half hour each day. This is actually a relatively small number of programs, considering the wide range of environmental sounds encountered each day. Kates [10] suggests that six clusters would be a reasonable number for a hearing aid classification unit. This will require that different audio environments be clustered together to form a relatively small number of classes.

This chapter will examine some of the current literature on audio environment classification, looking at the types of classifiers used, the classes that are used as output and the types of features used for input.

3.2 Classifiers

The classification of audio environments has been studied by a number of different researchers looking at many different classifiers. These techniques differ not only in the type of classifier used but also in the classifier configuration, the features used as input and the audio environments tested. The classifiers used range from relatively simple rule-based classifiers to more complex ANNs and HMMs.

3.2.1 K-nearest Neighbours

One of the simplest classifiers tested is the K-nearest neighbours (K-NN) classifier. The distance measure used and the numbers of neighbours both contribute to the final accuracy of the system. This is a logically simple classification procedure, but it is actually quite computationally expensive since it requires the calculation of the distance to every point in the training set. Often, K-NN classifiers are used as a basis for comparison for more sophisticated classification techniques.

A 1-NN classifier was tested by Peltonen et al. using six separate classes of noise and was found to have a recognition rate of only 68.6% [35]. The performance of the more sophisticated classifiers was much better, indicating that 1-NN classifier is likely too simple for environment classification. The performance of a K-NN classifier may be improved by using a different K value, or a different feature vector. However, it is unlikely that a simple K-NN classifier will be able to deliver the performance required in an environment classification system for a hearing aid.

This classifier remains a good basis for comparison, due to the relative simplicity of its implementation, and its well-understood functionality. The K-NN classifier forms a good baseline of performance for more sophisticated classifiers.

3.2.2 Bayes Classifiers

A naive Bayes classifier is another simple classifier that has been used for environment classification. A naive Bayes classifier is based on probabilities. A Bayesian network is used to determine the probability of an event given a certain known set of inputs [36].

In [37], Buchler et. al. tested a Bayes classifier against a rule-based classifier, an ANN and an HMM for classification of speech, noise, speech-in-noise and music. Both the ANN and the HMM performed better than the Bayes. However, the authors note that the Bayes classifier is faster than either model, and hence may be a good choice if the computation time is limited.

3.2.3 Gaussian Mixture Models

A slightly more complex type of classifier is the Gaussian mixture model (GMM) classifier. This classifier models the different environment classes as mixtures of different Gaussian probability density functions, with varying properties. An initial estimate of the system is iteratively corrected, most often using an expectation-maximization algorithm. The mixtures are trained on known sets of data, and the unknown samples are classified in the class with the highest probability [38].

Gaussian mixture models are a fairly common type of classifier, but they are relatively simple, so they tend to be used only for comparison to other types of classifiers. Malkin and Waibel compared a GMM classifier to a neural network [39], Ravindran and Anderson compared GMMs to an AdaBoost classifier [40] and Peltonen et. al. compared GMMs to K-NN classifiers [35]. The results of all of these studies indicate that the GMM is the weaker of the tested classifiers. It does not appear that the GMM is the best classifier for this task.

3.2.4 Hidden Markov Models

In two separate papers, Nordqvist and Leijon examine the use of HMMs to classify different audio environments. In [41], an HMM is used to classify a telephone environment. The study uses a two-stage HMM. The first stage is used to classify a telephone and face-to-face speech in quiet and noise environments. The classification is based mostly on signal modulation, since telephones are band-limited. The second classifier is used to determine the individual parts of the signal, speech or noise. The classifier was able to distinguish between the two environments, and the switching takes 10-15 seconds. However, the classification rate is very dependent on the conversation style.

This study shows potential uses for HMMs in both environment classification and in noise reduction. The environment classification part of the study is very limited, however, as only two different environments are compared. The time required to change is also quite large, as the user would need to be on the telephone for 15-20 seconds before having the hearing aid switch. This might make it difficult to hear the first part of the telephone conversation. This is an important part of the conversation if the user is attempting to verify the identity of the person on the line. Additionally, although the HMM uses the signal modulations in the model, there is no comparison to a classifier based solely on the signal modulation.

A second study by Nordqvist and Leijon [33] uses HMMs to classify three different audio environments: speech in multi-talker babble, speech in traffic and clean speech. The paper uses the same two-stage classifier as the first study, where one classifier is used to determine the environment and a second classifier is used to

pick out the individual parts of the signal. An individual HMM is used for each environment, and the output is the HMM having the highest probability. The system was able to give hit rates from 96.7% to 99.5%, with false alarm rates from 0.2% to 1.7%. The classifier produced a change in output within five to ten seconds after an abrupt change in the audio environment. The system may have some difficulty in reverberant environments, where clean speech is classified as speech in babble. The number of environments tested is fairly limited. However, the results are very encouraging.

In both of their studies [33, 41], Nordqvist and Leijon use a two-stage classifier, with one stage used to separate the speech and noise parts of the signal. This type of an application is well suited to noise reduction techniques. In a similar application, Sheikhzadeh et. al. [42] use HMMs for speech enhancement and compare the performance of the HMM to spectral subtraction and Weiner filtering. The HMM is a continuous, autoregressive HMM. The HMM performed better than the standard methods in both objective and subjective tests. However, it was a more complex system to implement, which can be costly in a hearing aid since it runs on very low power.

A similar classification problem is the classification of music samples. Pollastri and Simoncelli applied HMM classification to the problem of music classification in their 2001 study, attempting to use HMMs to determine the composer of an unknown music piece. The study attempted to classify the music of four classical composers (Dvorak, Mozart, Beethoven and Stravinsky) and one pop composer (the Beatles). HMMs were chosen because music is time dependent and HMMs are

able to incorporate time-dependent features. The HMM used was a linear left-right HMM and testing showed good performance using 12 classes. Increasing the size of the codebook used did increase the accuracy, although this increase flattened after a certain point, indicating that there is a near-optimal number of codebook values for a certain network. The HMM consistently outperformed the regular Markov model. The best performance attained was 42% correct, which seems very low. However, human music experts were only able to correctly identify 48.16% of unknown samples. Overall, the performance of the HMM is approaching the performance of a human expert, which is good for such a complex problem.

3.2.5 Artificial Neural Networks

An alternate to the HMM classifiers is to use an artificial neural network. Neural networks are well suited to classification tasks, and would therefore be a good match for this application.

Buchler et. al. [37] studied both HMMs and neural networks for sound environment classification, and compared them to the performance of a minimum distance and a Bayes classifier. The study found that the HMM was the most successful model for environment classification with an 88% accuracy rate, followed closely by the neural network with an 87% accuracy rate. These two models are very close in performance, therefore it is difficult to definitively claim that the HMM is a better classifier. The performance of both the minimum distance and Bayes classifiers was much worse. Interestingly, the HMM and the neural network functioned best using different sets of signal features. This is discussed further in Section 3.3.

Shau, Xu and Kankanhalli [43] also studied an HMM implementation and compared it to a support vector machine (SVM), which is another type of neural network. Their work looked at determining the genre of unknown music samples. The framing of the samples was done based on the intrinsic rhythm of the samples, which better captures the natural structure of the genre, and can be used to track the movement of the piece in time. Classification was done using one five-state HMM for each genre. The classifier was able to distinguish between pop, country, jazz and classical genres with 89% accuracy. The performance of the HMM was actually slightly worse than the performance of the support vector machine, but the HMM offers the advantage of being more easily able to add new genres, since it does not require retraining the entire system. This is actually an interesting point that holds true for environment classification as well. A neural network would require an entirely new set of weights if one new class was added, whereas an HMM network is fairly easily expanded by adding a new model to the system.

Temko and Nadeu [44] studied the classification of 16 different audio events found in meeting rooms. The study compared a GMM and an SVM. They also experimented using different clustering schemes for the SVM, using a standard clustering algorithm, and a clustering algorithm similar to a binary tree implementation. The best results were obtained using the SVM with the new proposed clustering scheme, with a hit rate of 88.29%.

Both Dagli [45] and Felton and Wang [46] use neural networks for music genre classification. Dagli uses a radial basis function neural network to identify seven different genres and gives an overall accuracy of 90%. Felton and Wang use an

SVM to identify seven different genres. The results from this study were a bit more disappointing, with hit rates from 40% to 95% depending on the genre.

Neural networks have also been used for relatively smaller classification tasks. Khan, Al-Khatib and Moinuddin [47] looked at classifying speech and music from unknown audio samples. They used a multi-layer perceptron feed-forward network, trained using back-propagation, and were able to attain a 96.6% accuracy using ten hidden nodes. This is an impressive accuracy rate. However, there are a small number of possible classes used in this implementation, so it is likely unreasonable to expect this high an accuracy rate when classifying a larger number of environments.

A similarly impressive result was achieved by Bugatti, Flammini and Migliorati [48] using neural networks for the same problem. The study compares a neural network to a Bayesian classifier for the task of classifying speech and music data from unknown samples. A pre-processor is used to remove silent segments and create the feature vectors.

The neural network is a multi-layer perceptron, trained using the Levenberg-Marquardt method. The total error rate of the neural network is only 6%, compared to the error rate of 17.7% for the Bayesian filter. Music misclassifications made up the vast majority of the error rate for the neural networks. Most of the misclassifications were for rap music, which has strong speech components. Unfortunately, the two classifiers used vastly different feature vectors for the Bayesian and neural network classifiers, so it is possible that some of the difference in their performance came also from the features used to describe the audio sample. The feature vector for the Bayesian classifier is very short and derives mostly from a single feature,

whereas the feature vector used for the neural network is very extensive.

3.2.6 Summary

Both the HMM and the neural network classifiers appear to have potential for audio environment classification. The multi-layer perceptron neural network is the most common and appears to perform well. The GMM does not appear to perform as well as the other potential classifiers. The K-NN also does not tend to perform well. However, since it is one of the simplest classifiers, it provides a good baseline against which to compare other classifiers.

3.3 Feature Vectors

Each of the models discussed require an input representing the signal. It is possible to use frames of signal itself, or the frequency spectrum. However, this is not always the best way to represent the signal for classification, and it often quite a large representation. Normally, a signal can be classified using a smaller number of inputs by characterizing the signal using different features.

The representation of the signal as a vector of features has been the topic of a number of different papers. The set of features selected need to represent the signal and distinguish it from signals in other classes, hence the feature vector selected will be dependent on the signals being represented, and the classes used. For example, identifying white noise from speech babble can often be accomplished using just the mean frequency. Classifying single-talker speech from multi-talker babble can often be accomplished using a feature including the signal modulation. The best

set of features is also dependent on the type of classifier used [37].

Some papers use a relatively simple set of features. Dagli [45] uses one of the simplest feature sets, consisting of the FFT data. Felton and Wang [46] also uses the FFT, but also includes a spectrogram and the Mel-frequency cepstral coefficients (MFCCs). Most often, however, the FFT is not included directly and instead the signal is represented by a number of features that are used to characterize the spectrum.

In [10] Kates presents and analysis of variance (ANOVA) on a set of four features for audio environment classification. The paper suggests using three frequency domain features, mean frequency and slope of the high and low frequency components, as well as the envelope modulation of the signal. The mean frequency is defined as the first moment of the signal, and the slopes of the high and low frequencies are the log spectrum curve fits of the frequencies above and below the mean frequency. The envelope modulation was originally defined as the ratio of the mean to the standard deviation of the mean magnitude within a certain frequency bin. It was found, however, that the envelope modulation was relatively steady, and therefore could be simplified as the ratio of the mean to the standard deviation of the mean magnitude of the entire signal.

In ANOVA tests, Kates found that each of the features described was significant, and their interactions were not significant, indicating they each represented a distinct part of the signal. When testing the feature vector with a real classifier, Kates was able to achieve better than 90% accuracy when using less than seven clusters [10]. The features used in Kates' paper are often used as at least part of

the feature vector in other papers on classification.

Another very common feature for classification are the Mel Frequency Cepstral Coefficients (MFCCs). A cepstrum is the DCT or FFT of the dB representation of the signal. The Mel scale is a frequency representation of a signal that corresponds to the human auditory system, emphasising the lower frequencies. Hence the MFCCs of a signal are the discrete cosine (DCT) or Fourier Transform (FFT) of logarithm of the signal (dB representation) transformed into the Mel scale. Linear cepstral coefficients can be used by not transforming the signal first into the Mel scale, and simply taking the DCT or FFT of the dB representation [49].

A paper by Malkin and Waibel [39] used 64 MFCCs and the mean frequency. The feature vector was tested using both a GMM and a neural network classifier, and the study found that the best error rate was 22.21% using the neural network with 16 hidden nodes, and the best rate for the GMM was 22.43%. A combination of the two classifiers was able to give an error rate of 19.95%. The use of MFCCs in the feature vector is actually quite common when using a GMM classifier. The hit rate for this study is quite low in comparison to some of the other studies, indicating that MFCCs or GMMs may not be a good representation or classifier for environment classification.

Shao Xu and Kankanhalli [43] used MFCCs and LPC derived cepstrum coefficients as well as their delta and acceleration values as a feature vector for an HMM classifier used to classify music by genre. Their system was able to give an accuracy of 89%. However, both the problem and classifier were different, so these results cannot be directly compared with the study of Malkin and Waibel.

Ravindran and Anderson [40] also did some work using MFCCs and GMMs. The study used a model of the human auditory system to generate features, and compared the new feature vector to a feature vector comprised of MFCCs, using GMM and AdaBoost classifiers. For the human auditory model, the basilar membrane (BM) is modelled as a filterbank with 128 bands from 180 Hz to 7246 Hz. A spatial derivative and half-wave rectifier models the cochlear nucleus (CN), and a temporal integration over 8 ms represents the central auditory neurons. The features used are the discrete cosine transform of the log of the model. The human auditory feature vector performed better than the MFCC feature vector, with the best performance from the AdaBoost classifier. This is an interesting approach to the feature vector selection, since it is based on a model of the human auditory system. Unfortunately, the feature vector consists of 128 features, since the BM is modelled as a 128-band filterbank. The frame size on the system is only 8 ms, which would require a both a large classifier and very fast processing of the input to generate the feature vector. Because a hearing aid is limited in its power usage, such a large feature vector requiring fast processing is not possible. Because it uses a filterbank, however, it may be possible to eliminate some of the additional processing by combining it with the frequency response shaping already required by the hearing aid.

A paper by Peltonen et. al. [35] also examined the use of different feature vectors for classification, using a K-NN and a GMM classifier. The study examined a large number of potential features in three different categories, and attempted to determine a good combination of these features. The features tested were grouped

into three different categories: time domain features, frequency domain features and linear prediction and cepstral coefficients. The time domain features consisted of the zero crossing rate (number of zero voltage crossings in a frame) and the short time average energy within a frame. The frequency domain features consisted of the band energy ratio (ratio of the energy in a certain band to the total energy), the spectral centroid (the first moment of the spectrum, similar to the mean frequency in the Kates paper [10]), the bandwidth (width of the range of frequencies), the spectral roll-off point (the frequency at which a certain amount of the power spectrum is contained in the lower frequencies), and the spectral flux (change of the shape of the power spectrum between frames). The linear prediction and cepstral features consist of the linear prediction coefficients (LPCs) found using autocorrelation, and the cepstral coefficients and MFCCs, both derived from the LPCs. The best classification rate was 68.4%, which was found with a 1-NN classifier, using the band-energy, the flux, the roll-off and the centroid of the signal. Interestingly, MFCCs did not make up part of the best feature vector, which may support the conclusion drawn by Ravindran and Anderson in their paper [40], which suggested that there were more relevant features than just the MFCCs of the signal.

A classification rate of just 68.4% is actually quite a low classification rate compared to some of the other studies. This may be because a 1-NN classifier is an extremely simple classifier. A 1-NN classifier is very dependent on the data set.

In [44], Temko discusses classification of meeting room events with an SVM. The features suggested in the paper are from three categories. In the perceptual-spectral category, the features used are the zero-crossing rate, the short time energy, the

subband energies in four subbands, pitch and the spectral flux between two adjacent frames, measured in four subbands. In the cepstral-based category, 12 MFCCs from 20 bands are used, not including the 0^{th} coefficient. A last category used is the FFT-based spectral parameters, which are based on the log of the filterbank energies.

Some of the features that have been suggested by both Peltonen et. al.[35] and Temko [44] are actually quite common features for audio classification. In [47], Khan, Al-Khatib and Moinuddin provide a short survey of previous literature, looking at possible features. The list of possible features is fairly long and includes low energy frames, roll-off point of the spectrum, spectral flux, zero crossing rate, spectral centroid, four Hz modulation energy, cepstral residual, pulse metric, cepstral coefficients, amplitude, pitch, harmonic coefficients, MFCCs, log energy, line spectral frequencies (LSF) and differential LSF (DLSF). The variance and delta values of many of these features are also included in many feature vectors. The authors decided to use a feature vector consisting of some previously used features, and some new features. The feature vector used consisted of the percentage of low energy frames, the RMS of a low-pass response, the spectral flux, the mean and variance of the discrete wavelet transform, the difference of maximum and minimum ZCR and the LPCs. Using this feature vector, they were able to attain an accuracy of 96.6% on a neural network classifier used to identify speech and music.

Several of these features are also used in a study by Bugatti, Flammini and Migliorati [48]. The authors also used classifiers to identify speech and music. For a Bayesian classifier, the authors used the variance of the ZCR, the third order

moment of the ZCR and the difference between the number of ZCR samples above and below the mean. For the neural network classifier, they used the spectral flux, the standard deviation of the short-term energy, the minimum short-term energy, the product of the mean and standard deviation of 86 centroid values, the mean and standard deviation of the cepstrum coefficients, the ratio of high-frequency power to the whole-spectrum power and syllabic frequency. They were able to attain an error rate of 6.0% on the neural network, but the Bayesian classifier gave a much higher error rate of 17.7%.

Buchler et. al. [37] looked at a different set of features in four different categories: amplitude modulation, spectral profile, harmonicity and amplitude onsets. Amplitude modulation can be described in terms of the width of the amplitude histogram, the modulation spectrum of the frequency envelope in three ranges, or as in Kates [10], as the logarithmic ratio of the mean to the standard deviation of the magnitude across an observation interval. The spectral frequency is described here by the spectral centre of gravity and the fluctuations of the spectral centre of gravity. Pitch is modeled by tonality and pitch variance, but the authors do not use the pitch value itself. A pitch is detected if there is a peak above 20% of the signal energy within 50-500 Hz. When a pitch exists, the signal is harmonic, when there is no pitch it is inharmonic. Tonality is the ratio of the harmonic to inharmonic periods over time, and pitch variance is only a measure of the harmonic parts. Amplitude onsets are calculated using the envelope of the signal in 20 Bark bands, using a time constant of 10 ms. An amplitude onset is identified as either a 7 dB or a 10 dB difference between 5.8 ms frames. Four features are used to

describe the amplitude onsets: onset mean and variance of the onset strength in an observation interval of one second, number of common onsets across bands, and the relation of high to low frequency onsets. Beat can also be determined in this way by determining the steady onsets over a longer period of time.

A number of different classifiers were tested, and the best feature vector depended on the classifier used. The best performance was an HMM using a feature vector consisting of tonality, amplitude width, centre of gravity, fluctuation of centre of gravity, the onset mean and the number of common onsets between bands. A two-layer perceptron with two hidden nodes was only slightly worse, using tonality, amplitude width, centre of gravity, fluctuation of the centre of gravity, number of common onsets between bands, and the beat. Tonality was considered important, as were at least one of the amplitude modulation features. The spectral features of centre of gravity and fluctuation of the centre of gravity and at least one onset feature can improve the score. The beat feature was not considered to be important [37]. This work demonstrates and describes a large number of possible features for use in a classifier, and shows how the feature vector is changed as the classifier characteristics are changed.

As demonstrated by Buchler et. al. [37], the feature vector is dependent on the type and characteristics of the classifier used. To cope with this problem, Feldbusch [50] describes a heuristic for feature selection that incorporates the classifier as the features are tested. In this case, the classifier tested is a neural network. Finding a good feature vector is important since the signal must be fully described to achieve a good performance. Research has also shown, however, that the performance

degrades when too many features are used, so the problem is more difficult than simply identifying all the possible ways to describe a signal. Unfortunately, this makes finding an optimal feature subset an NP-hard problem, which is why a heuristic method is described in this paper. There are generally two ways to find a feature vector for a system. A fitter measures the performance of the feature vector based only on the data itself, whereas a wrapper incorporates the classifier and measures the performance of the feature vector for each particular classifier. Because neural networks must be trained, it is not efficient to build a wrapper around a neural network. Instead, a wrapper is built around a fast statistical classifier whose classification rate corresponds to the classification rate found from a neural network. The list of features tested and the final features selected are not discussed. However the paper indicates a means of performing feature selection.

The selection of a feature vector is an NP-hard problem [50]. Most papers suggest different possible features and different considerations; however, it is likely that a good feature vector will need to be determined through a heuristic approach.

A summary of the features vectors is presented in Table **??**

3.4 Audio Environment Classes

Although there has been a fair amount of work done on determining a good feature vector for the classifier, there has actually been very little work dedicated to determining a good set of classes to actually use in a hearing aid. Most of the papers referenced in this section either do not discuss the classes used at all, or mention the classes used, but do not discuss the reasons for using those classes. This is

actually a very significant decision since it will directly affect how well a user can hear in certain situations. It is important to determine how many classes should be used, which classes are important, which classes are similar enough that they can be clustered together. Additionally, it is important to determine how environments will be classified if they span multiple classes, for example, a crowded bar with both speech babble and music.

In his 1995 paper [10], Kates suggests that six separate environment classes should be able to cover the requirements for the average hearing aid user. Based on research with manual multi-programmable hearing aids, it was found that patients use 3.4 ± 1.1 programs for at least one hour every day, with 4.3 ± 1.2 programs in use for at least one half hour every day. Users also identified three important classes of noise that should be included in a classification program: outdoors (nature), automobile travel (in-car) and shopping. In his study, Kates used 11 different noise environments that were then clustered: apartment, babble, dinner, dishes, Gaussian, printer, traffic, typing, sentence, siren and vent. Kates found that the apartment and traffic noise were very similar, likely because the majority of the noise in the apartment comes from traffic. Ventilation noise was also found to be quite close to both the apartment and traffic noise. Typing, printer and dishes were also found to be quite similar, likely because they are all percussive noises.

Other studies have looked less closely at the actual clustering, and just assumed several classes of noise. In [33] Nordqvist and Leijon look at classifying speech in traffic noise, speech in multi-talker babble and speech in quiet. In [41], they look at speech in quiet and in traffic noise in a face-to-face environment and over the

telephone. Ravindran and Anderson [40] examine four classes of noise in five environments. The classes used were speech, music, noise and speech in noise, and the five environments were social (babble), in-car, office, industrial and traffic. Buchler et. al. [37] use the same classes and environments. Peltonen et. al. also use five environments, but their selection is slightly different: outdoors, vehicles (traffic), public/social (babble), office/meeting room/quiet room, home and reverberant spaces.

There is some consensus on some of the clusters to be used. Traffic appears in many of the studies, and the outdoors environment also likely includes some traffic components for cityscapes. The in-car environment also appears in two studies [40, 37], and likely includes some aspects of the traffic noise, but in a more enclosed environment. Office noise similarly appears in a number of the studies [37, 40, 35], and likely includes a component of typing noise as well. The industrial environment used by Buchler et. al.[37] and Ravindran and Anderson [40] would also likely include some percussive noise. Speech babble or social settings are also very common, most likely because it is a background that is often identified as troublesome for hearing aid users. Music is also likely an important environment to identify as listening to music would likely require a flatter response from the hearing aid, and less noise reduction to avoid cancelling part of the sound.

Overall, however, there seems to be little work on identifying and clustering audio environments for this application. The classes used in each study are summarized in Table **??**.

3.5 Summary

Table **??** presents a summary of the classifiers, feature vectors and classes discussed in this literature review, along with the final test results. Overall, the HMM and the ANN both appear to be good candidates for audio classification. Much of the work has focused on the selection of a good classifier and although many different classes of audio samples have been tested, most authors have selected the classes based on prior knowledge of the problem, rather than through a formal technique. Similarly, although many features have been suggested, few authors have used a formal feature selection technique to choose their input features. Hence, this work will focus on the use of a more structured approach to the selection of the classes and features, as well as the selection and testing of an appropriate classifier.

Chapter 4

Initial Classifier Tests

[1]

This research focuses on the development of an environment classifier that can be used in a hearing aid. This thesis examines answers several questions related to classifier systems:

1. Which type of classifier would work well for this application?

2. What is a reasonable set of output classes for the classifier?

3. What features should be used as input to the classifier?

The project is divided into four different stages. Each stage of the research is presented in a separate chapter, as the results of each stage influence the methods of the proceeding stage. Each chapter presents the methods, results and discussion for its respective stage. The stages of the research are:

[1]Portions of this chapter have been published in [51] and have been accepted for publication in the International Journal of Information Technology and Intelligent Computing

1. Assessing the different classifiers for functionality (this chapter)
2. Determining the desired output classes (Chapter 5)
3. Performing feature selection using the selected output classes (Chapter 6)
4. Testing the candidate classifiers using the selected classes and features (Chapter 7)

4.1 Methods

Initial tests are performed using MLP, HMM and an MLP with windowed input in order to test the effectiveness of using the windowed input with the MLP. This section presents the features and data set and classifier configurations used for these tests. These algorithms are developed in C++ using the Microsoft .NET compiler.

4.1.1 Features

The feature vector for the initial tests is based on those suggested by Kates in [10]. The feature vector consists of the mean frequency, the high and low frequency slopes and the envelope modulation of the sample. The features are calculated on a 200 ms frame. These features are described in detail in Chapter 6.

The regular MLP uses a single frame of the signal to determine the class, and the HMM uses a sequence of five frames, consisting of one second of audio input. The windowed MLP uses a variable window size of two to five frames.

4.1.2 Data Set

The data set for the initial tests is taken from the Freesounds database at the Universitat Pompeu Fabra [52], and white noise generated in MATLAB. The files are separated into one-second segments. Each class is taken from 37 one-second segments. For the MLP, which only requires one frame of data per input, each one-second segment is further separated into four 250 ms segments. From each 250 ms segment, one 200 ms frame of features was extracted. The windowed MLP and the HMM each require more than one frame per input, hence each one-second segment is used to produce a single set of two to five input vectors.

4.1.3 K-nearest Neighbours

The K-NN is tested with 100 different K values from one to 100. Two thirds of the data set is used as training vectors and one third is used for testing. This gives a sufficient amount of data to train the algorithms, while still retaining some data for testing.

4.1.4 Non-windowed MLP

The network model used is a three-layer (one hidden layer) feed-forward perceptron, trained by back-propagation. Previous research has shown that a four-layer network (two hidden layers) works well for classification as it can match an arbitrary classification boundary [53]. A more recent study by Guang-Bin et. al. [54] shows that a three-layer perceptron net (one hidden layer) can also match an arbitrary boundary. In this work, a three-layer MLP is used because it is smaller and

therefore requires less power. Additionally, early tests performed as part of this work show that the four-layer network is actually more difficult to train as it tends to get stuck in local minima in the error space.

The input to the network is four nodes, with one node per feature. The output of the network is four nodes, with each node corresponding to a single class. This output coding is selected over a binary-coded output because in a binary output, two different classes can have one or more output node that is the same. This implies a connection between these classes that may not actually exist. Additionally, using a one output per class configuration allows the system to indicate an unknown class.

The number of hidden nodes is variable. Since the initialization for the weights and thresholds is random, the results of 50 separate training runs are averaged to produce the results. The run with the best performance on the test set is also saved. The network is tested using different numbers of hidden nodes (four to ten), and different numbers of training epochs (10,000 to 30,000).

4.1.5 Hidden Markov Model

The hidden Markov model implementation is slightly different than the MLP implementation, since a separate HMM is required for each audio class. The set of four HMMs makes up a model set, and the class of the input vector is determined as the HMM with the highest probability in the set.

The model used is a fully-connected model. In [33], Nordqvist and Leijon use a fully connected HMM for audio environment classification, which assumes that any state may transfer to any other state, and any state may be the initial state.

However, Rabiner assumes a left-right configuration for a similar application, in speech recognition. A left-right model only has state transitions to higher states, and the initial state is always the first state [26, 55]. In speech recognition, however, the order of the sounds matters, whereas in an audio environment the ordering is mostly random. Therefore, the fully-connected model is a logical choice for this application.

The state transition matrix (A) and the initial state probabilities (π) are initialized randomly. However, it has been shown that observation probability matrix (B) benefits from a better initial estimate [26], hence the observations are first clustered to give initial estimates for B, using K-means clustering.

A discrete model is used, hence the values used in the B matrix are determined using a vector consisting of codebook values. The actual input values are quantized to their codebook values before they are input into the B matrix. The B matrix contains the probabilities for each different vector, as opposed to simply holding the probabilities for each of the individual inputs. This allows the model to capture the relationship between the different inputs, but also greatly increases the size of the B matrix.

The HMMs are much more consistent since the initialization for the B matrix is deterministic. Therefore, the results are generated by averaging five HMMs. The model with the best accuracy on the test set is also saved. The HMMs are trained using different number of classes (three or four) and different numbers of codebook values (four to six). The number of classes is restricted to four since beyond this number the initialization of the B matrix becomes difficult. The HMMs are also

tested with different number of training iterations, from 10 to 30.

4.1.6 Windowed MLP

Although both MLPs and HMMs can match input patterns to output classes, HMMs have a theoretical advantage over MLPs as they can also track time-based changes. The MLP does not naturally contain a time-based component. The MLP can be adjusted to have a time-based component by using a windowed input. In a windowed MLP, the input to the MLP is modified to include past inputs by adding more nodes to the input layer, as seen in Figure 4.1.

When the input is windowed, the size of the input window must be considered as an additional parameter. In this case, the position of the sample in the window is not important since, in the vast majority of cases, the background is stationary. Therefore, the network is used to classify the background from the full sequence of inputs. In cases where the background is non-stationary, the change will propagate through the system, and should only cause a disturbance in the windows containing samples from more than one environment. For this system, the window is limited to five samples, which is only one second of data. Any disturbances could theoretically be smoothed using an averaging filter at the output.

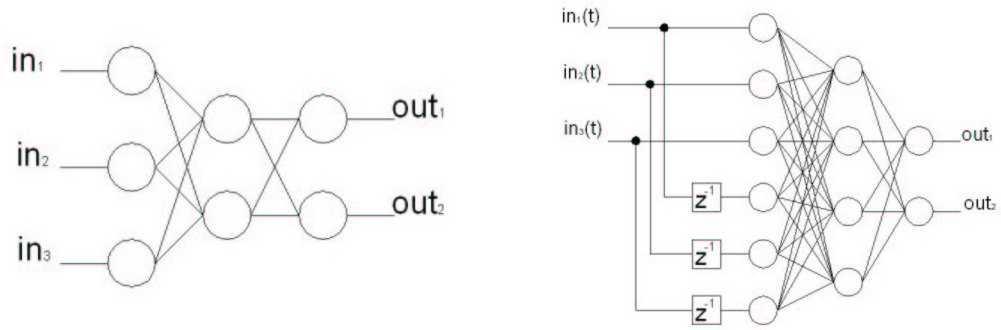


Figure 4.1: MLPs using normal and windowed input vectors

4.2 Results and Discussion

4.2.1 K-nearest Neighbours

The training and testing set accuracies are presented in Figure 4.2. The training set has the highest accuracy (100%) with a 1-NN classifier and the accuracy tends to decline as the K value increases. This occurs because the value being tested is actually a part of the set it is being compared against. For the training set the 1-NN value is the input itself and the class will therefore always be correct.

For the testing set, the set with the highest accuracy is the 8-NN. The 8-NN gives an accuracy of 81.8% for the testing set and an accuracy of 93.2% for the training set. However, all classifier values between 6-NN and 14-NN are above 80.7% for the testing set, which is only a 1.1% difference in accuracy. The 2-NN classifier also has an accuracy of 80.7%. However, the accuracy tends to decrease for K values larger than 14. Because the results are so dependent on the training set, it is difficult to say that 8-NN is generally the best K value for this application. Realistically, any small K value would likely produce similar results. For this thesis, however, the 8-NN classifier is used for comparison.

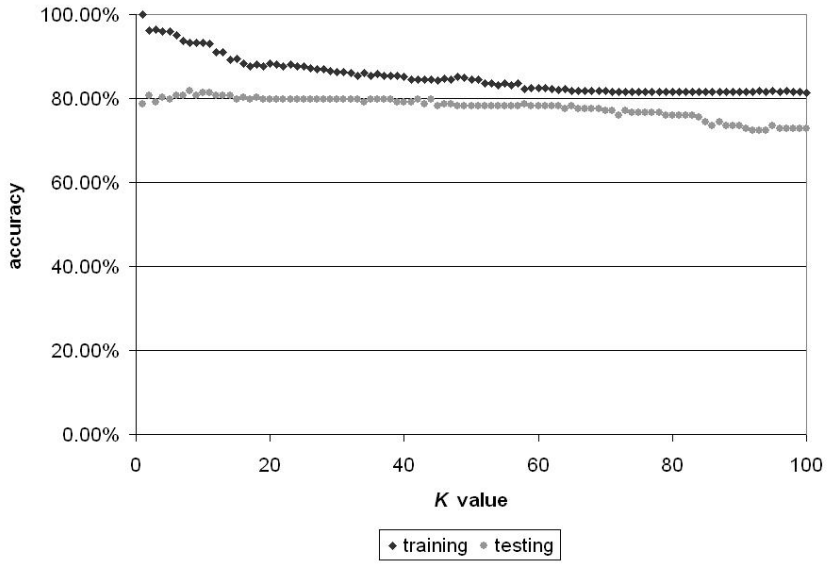


Figure 4.2: Accuracy of the K-NN classifier with different K values for the initial tests

	selected			
actual	babble	traffic	typing	white
babble	37.5%	62.5%	0%	0%
traffic	8.3%	89.6%	2.1%	0%
typing	0%	0%	100%	0%
white	0%	0%	0%	100%

Table 4.1: Confusion matrix for the 8-NN classifier for the initial tests

The confusion matrix for the 8-NN is presented in Table 4.1. It is clear from this table that the 8-NN has difficulty separating the babble and traffic classes. The accuracy of this classifier for the babble class is only 37.5%, which is clearly not suitable for this application.

4.2.2 Non-windowed MLP

The average accuracy of the MLP is actually quite low, ranging from 75.5% to 78.8% accuracy on the testing set, depending on the number of hidden nodes and the number of training epochs. The best-run values are much higher than the average values, ranging from 84.9% to 92.7% accuracy on the testing set. This large range indicates that the MLP is not a very reliable technique. Neural networks are able to find local minima in the error space, but not a global minimum. Accordingly, it is possible to generate models that do not fully converge to an acceptable solution for the training set. This may be part of the reason for the relatively low average accuracy.

There is no definite trend with respect to the number of training epochs. A 20,000 epoch training period is selected for comparison, as it is the midrange of the tested values.

From the graph in Figure 4.3, it can be seen that there are also very few definite trends with respect to the number of hidden nodes. There is a very slight decrease in both best-run and average accuracy for networks with more than seven hidden nodes. The network with five hidden nodes (trained with 20,000 epochs) is selected for comparison. For this model, the best-run accuracy for the testing set is 89.1%. Using these weights, the accuracy on the training set is 90.5%. The average accuracy achieved is 78.6% for the testing set and 95.4% for the training set. This model is selected because it has fewer than seven nodes, and the smallest difference between the average and best-run accuracy, making the model slightly more reliable than the others. It also has the smallest difference between the best-run accuracy for the

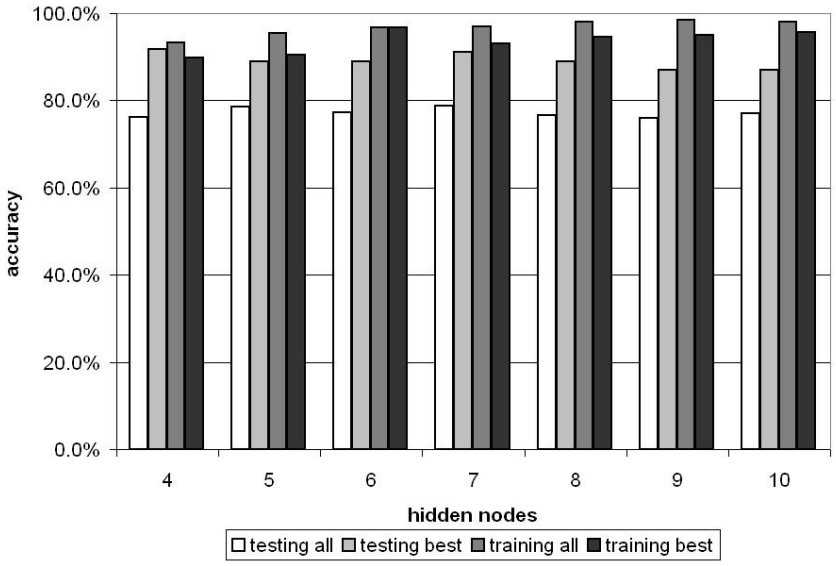


Figure 4.3: Accuracy of MLP using different number of hidden nodes for the initial tests

training and testing set, indicating that this model is more robust than the other models.

Practically, however, the number of hidden nodes does not greatly affect the final accuracy, and any relatively small number of hidden nodes would be an appropriate selection.

The accuracy of the 8-NN is actually better than the average accuracy of the non-windowed MLP. However, the best-run results of the MLP are better than the K-NN, which indicates that with proper training the MLP is likely the better choice.

The confusion matrix for the best run of the five-hidden node matrix is presented in Table 4.2. The matrix shows that the majority of the confusion is between the babble and the traffic classes. This observation holds true for all network

	selected				
actual	babble	traffic	typing	white	unknown
babble	87.5%	6.2%	0%	0%	6.2%
traffic	10.4%	75.0%	4.2%	0%	10.4%
typing	0%	4.2%	93.8%	0%	2.1%
white	0%	0%	0%	100%	0%

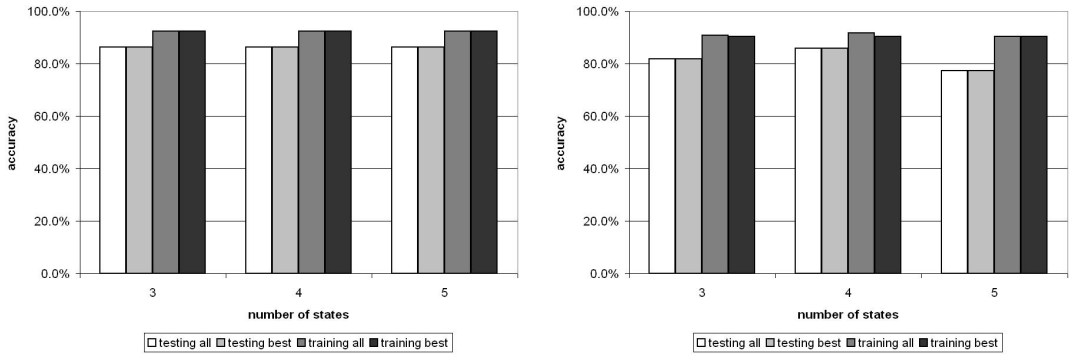
Table 4.2: Confusion matrix for the best-run of the five-hidden node MLP for the initial tests

configurations for both the testing and the training sets. This is similar to the problem encountered with the K-NN classifier and may be an indication that the features used in these tests are not sufficient to properly separate these two classes.

4.2.3 Hidden Markov model

The average accuracy of the HMM is higher than the MLP, ranging from 72.7% to 86.8% on the testing set. The best runs of the HMMs also range from 72.7% to 86.8% accuracy. The majority of the HMMs have an average accuracy that is the same as the best-run accuracy, and those that are not the same differ only by the classification of one sample. This indicates that the models converged to a similar result every time they are trained. The training of the HMMs is clearly more reliable than the MLPs. This is likely because the initialization for the HMMs is not completely random, but is based on a K-means clustering process.

The results from the HMM tests show that the number of training iterations has little effect on the final accuracy of the model. The largest difference between models trained for different iterations is only 2.3%, and the majority of the models see no change in accuracy as the number of training iterations is changed. The



a) Codebook size = 4 b) Codebook size = 6

Figure 4.4: Accuracy of the HMM using different numbers of classes for two HMMs with codebook sizes of four and six for the initial tests

number of training iterations required for the HMM is much smaller than that required by the MLP. However, the training time is still relatively long as the matrices are much larger than the matrices found in the MLP.

There also appears to be little difference in accuracy between models with different numbers of states, particularly when using a smaller number of codebook values, as seen in Figure 4.4. As the number of codebook values increases, the accuracy of both the two-state and four-state models decrease slightly. The four-state models are more difficult to initialize with the K-means clustering, which may be an indication that there are not actually four distinct states in all the environments being classified. Overall, the three-state model appears to be the best model.

The parameter that has the greatest effect on the performance is the number of codebook values. The results are relatively similar for codebooks with three, four and six quantized values. However, the performance of the codebook with five values is significantly lower, as seen in Figure 4.5. It is possible that the codebook with five values separates or combines a cluster of input vectors that would otherwise be

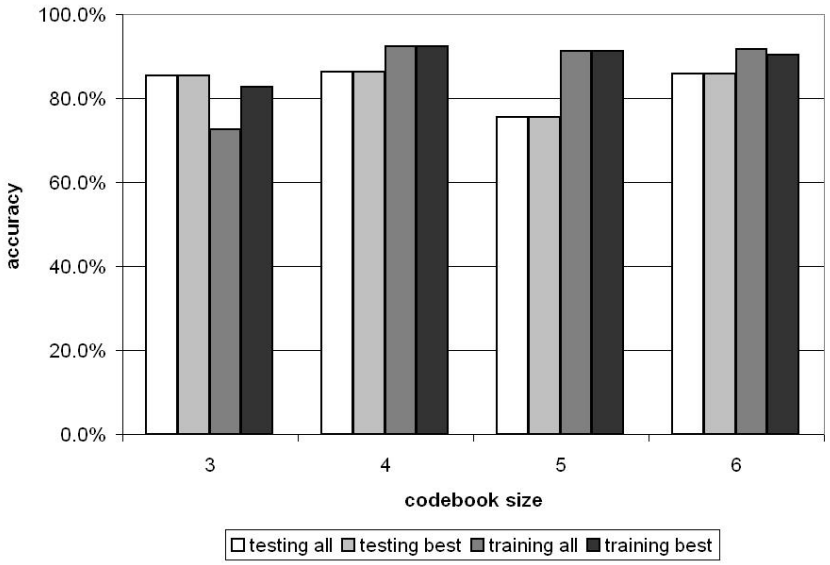


Figure 4.5: Accuracy of the HMM using different codebook sizes for a three-class HMM for the initial tests

classified differently. The four-value codebook is selected for comparison because it is the most accurate for the three-class model. It is also a relatively small codebook, making it faster to train and easier to fit in the relatively small memory found on DSP-based hearing aids.

The confusion matrix for the HMM implementation is shown in Table 4.3. It is clear from Table 4.3 that the HMM also has difficulty classifying the babble sound class. However, the HMM most often confuses babble with typing. This is a further indication that different features may be needed to properly classify the babble class.

	selected			
actual	babble	traffic	typing	white
babble	54.5%	0%	45.4%	0%
traffic	9.1%	90.9%	0%	0%
typing	0%	0%	100%	0%
white	0%	0%	0%	100%

Table 4.3: Confusion matrix for the best-run set of the three-class four-codebook value HMM for the initial tests

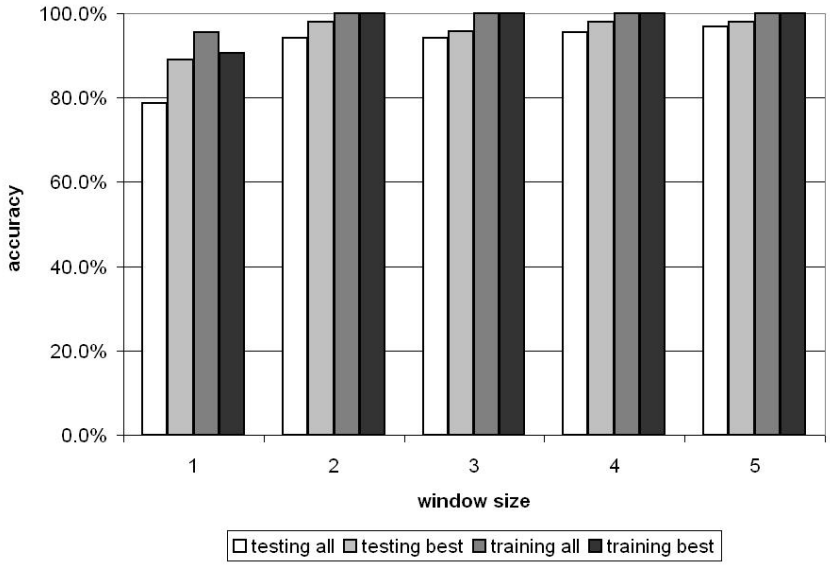


Figure 4.6: Accuracy of the windowed MLP vs. window size

4.2.4 Windowed MLP

Testing with the non-windowed MLP shows that for the four-input network, a network with five hidden nodes is a good model. Using the results from the non-windowed MLP tests, the windowed MLPs are set so that the ratio of hidden nodes to input nodes is 5:4. Window sizes from two to five samples were tested, and the results are presented in Figure 4.6.

Results indicate that even a small window provides a significant increase in accuracy over the non-windowed MLP. The testing set best-run accuracies are between 95.8% and 97.9% depending on the window size, compared to 89.1% for the non-windowed MLP, and 86.4% for the HMM. Additionally, the average accuracy is increased significantly to between 94.2% and 96.9%. The windowed MLP is clearly more accurate than both the non-windowed MLP and the HMM.

The difference between the average and best-run accuracy for the non-windowed MLP is 10.5%. The windowed MLPs show a difference of only 1.0% to 3.7%. This indicates that the windowed MLP will produce more consistent results.

One of the major benefits of the windowed MLP is that it is robust. Although the non-windowed MLP is able to generalize in certain cases, the average accuracies for the training and testing sets are quite different. For the non-windowed MLP, the difference between the average accuracy for the training and testing sets is 16.9%. For the windowed MLP, this difference is reduced to 5.8% for the two-sample window, and just 3.1% for the five-sample window.

The average accuracy increases slightly as the window size is increased. However, the best-run accuracy is relatively flat, with the two, four and five-sample windows having a best-run accuracies of 97.9%, and the three-sample window having a slightly lower best-run accuracy at 95.8%. The reliability also increases slightly as the window size increases. The larger window size is therefore likely easier to train, but the smaller window sizes are also capable of achieving excellent accuracy.

Although the larger windows are slightly more accurate on average, the two-sample window was selected for comparison since it is smaller and therefore has

	selected				
actual	babble	traffic	typing	white	unknown
babble	100%	0%	0%	0%	0%
traffic	8.3%	91.7%	0%	0%	0%
typing	0%	0%	100%	0%	0%
white	0%	0%	0%	100%	0%

Table 4.4: Confusion matrix for the best run of the two-sample windowed MLP for the initial tests

a smaller computational load, and easier to store on the relatively small memory found in a hearing aid.

The confusion matrix for the MLP using window size of two samples is presented in Table 4.4. There is only one misclassified sample, which is a traffic sample that is misidentified as babble. This is similar to the problems seen in the non-windowed MLP.

4.2.5 Comparison of Classifiers

The final accuracy of all three systems for the initial tests is presented in Table 4.5. The windowed MLP has the best accuracy for both the average and the best-run models. It is also the most general model. For a single set of training data the K-NN is the most reliable model because it is not trained. Of the trained models, the HMM is the most reliable, since the difference in accuracy between different training runs of the same model is very small. The windowed MLP is more reliable than the non-windowed MLP. Overall, the initial tests indicate that the windowed MLP is a good choice for audio environment classification.

	Testing set		Training set	
Classifier	average	best-run*	average	best-run*
KNN	81.8%	81.8%	93.2%	93.2%
MLP	78.6%	89.1%	95.4%	90.5%
HMM	86.4%	86.4%	92.3%	92.3%
WMLP	94.2%	97.9%	100.0%	100.0%

Table 4.5: Accuracy of three classifiers for the initial tests. *Note that best-run indicates the run giving the highest accuracy on the testing set.

Chapter 5

Selection of Audio Classes

It is necessary to determine what type of classes would be useful for a hearing aid user. Because microphone arrays have shown so much promise as a way to remove noise (please see Section 2.1.3), this environment classification system is intended for use with a noise reduction system based on a superdirective microphone array.

The intent of the class selection is to pick a set of logical, useful classes that reduce the overlap between different sets of weights, but still cover a wide variety of sounds. This chapter presents the methods used to perform this class selection and present and discusses the results of this procedure.

5.1 Methods

In order to select candidate classes, a clustering algorithm is used on a large database of sounds. These clustering algorithms are used to determine which classes can be grouped and which should be separated.

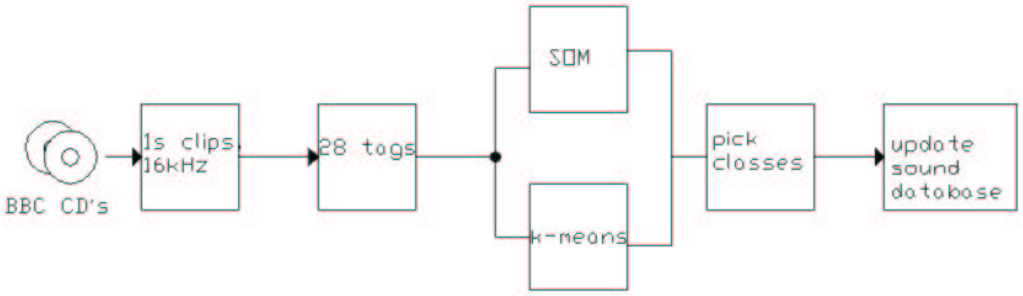


Figure 5.1: Procedure for the selection of output classes

The sounds being clustered are 1 s clips, tagged with one or more of 28 different possible tags. The clustering is performed using a SOM and a K-means clustering algorithm. The classes are logically selected using the output of both of these clustering algorithms. This process is illustrated in Figure 5.1.

K-means clustering separates the data into K distinct clusters, where K is chosen before the algorithm is started. This algorithm is simple and quite common, but the choice of K can affect the results. The SOM does not give distinct clusters, but can show how closely samples are related to each other. Using a combination of a clustering algorithm that gives distinct clusters and an algorithm that gives a more fuzzy set of clusters gives a better overall representation of the samples.

5.1.1 Data set

It is necessary that the data set for the selection of the classes represent a large variety of classes. It also needs to contain samples with more than one type of sound. For example, classes with both traffic and footsteps, or restaurant noise and babble. The data set is taken from the British Broadcasting Corporation environmental sound CDs. The sounds from the CDs are transferred to wav format

at 16 kHz. This sampling rate is chosen because sampling in a hearing aid is often performed at 16 kHz in order to conserve power. This is important because the autocorrelation of the signal is based on the signal lags and would therefore change as the sampling rate of the signal changed.

The audio files are divided into 1 s samples. One second is selected because it allows a sufficient number of samples to calculate the required features, but is still fast enough to properly track a person who is moving through different sound environments.

These samples are then identified as containing one or more of 28 different possible sounds. A summary of the samples used is presented in Table 5.1. The table denotes the number of samples used where the tagged sound is the only sound in the sample, the number where the tagged sound appears in combination with one or more other sounds, and the number of recordings those samples were drawn from. An effort is made to adjust the number of samples from each tag to generate a sample set that is more evenly balanced across the tags. However, some tags occur much more frequently and therefore appear in many more sounds in combination, and some tags appear only in a very small number of samples. The babble and footsteps samples appear in a large number of samples in combination because these sounds are so prevalent. Similarly, the nature tag is also quite prevalent because this tag was used to denote any rural, outdoor sound. Many of the subcategories of nature (eg. birds, water sounds) appear more in combination because they are also tagged as being nature sounds.

An effort was also made to ensure that samples were taken from as wide an

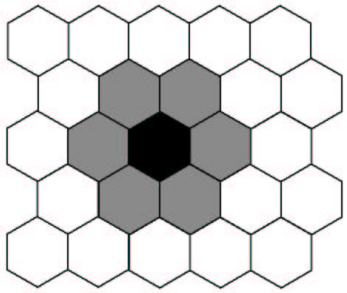


Figure 5.2: SOM honeycomb pattern. The black node is the best matching node, the grey nodes are the immediate neighbourhood ($N_c = 1$)

assortment of recording as possible. There are, however, still many sounds that are taken from only a single or a small number of recordings, simply because there are no other recordings available in the set.

Because the weights in a beamformer are usually based on the autocorrelation of the signal, the clustering is also based on the autocorrelation of the signal. MATLAB is used to take the signal autocorrelations and create the data files for the clustering algorithms.

5.1.2 Self-Organizing Map

A SOM is used to find clusters of similar sounds that could be put together to form a class of sounds. The SOMs are trained on the autocorrelation of the signal, using 5, 10 and 20 lags. The map used is a honeycomb pattern, which gives an immediate neighbourhood ($N_c = 1$) of six cells. This is illustrated in Figure 5.2.

The SOM training proceeds in two stages, and the α and N_c values change at different rates in each of these stages. The first stage is the ordering stage and it is trained for 3000 iterations. The neighbourhood size is reduced linearly from $\frac{1}{4}$ of

Table 5.1: Summary of the samples used for clustering

tag	alone	combo	recordings
in-car	40	40	7
in-bus	20	30	3
in-train	20	9	1
in-subway	20	13	1
car signal	0	10	2
nature	20	117	9
birds	10	57	10
geese	0	10	1
farm animals	0	10	1
dog	0	33	1
water splashing	2	28	1
water washing	0	27	1
water running	0	25	1
traffic	30	42	14
train/tram	30	1	2
subway station	25	0	1
footsteps	40	166	21
babble	40	352	29
children	20	71	11
laughter	8	40	14
applause	25	2	2
office noise	9	30	1
restaurant noise	7	83	5
electronic sounds	20	7	6
music	30	75	16
shopping noise	15	60	4
phone ring	0	10	1
industrial	60	0	15

the size of the map to three. The α value is reduced linearly from 0.9 to 0.1. The intent of the ordering stage is to make large, coarse changes to map, which changes the locations of groups of samples. By starting the neighbourhood at $\frac{1}{4}$ of the map size, large portions of the map are changed at one time. By starting the alpha value at 0.9, these changes are fairly large, which allows the samples to change the shape of the map.

The second stage is the convergence stage and it is trained for 6000 iterations. The neighbourhood size is reduced linearly from three to one, and the α value is reduced linearly from 0.1 to zero. This stage is intended to make small-scale changes within the groups, so the distance between adjacent samples is reduced. Hence, the neighbourhood and size of the changes is small. This stage is trained for longer than the ordering stage because the changes themselves are smaller.

The final output maps each cell on the map to the closest input vector. The map itself is drawn with an outer hexagon indicating the average distance to its neighbouring cells as a greyscale value, and an inner hexagon with a colour indicating the input vector tags. The average and maximum distance between cells is also recorded. Because there are a large number of different tags and combinations of tags, it is not possible to show each of the different combinations as a different colour since it would require using colours that are very similar. Instead, a map is drawn for each different tag, with different colours indicating different combinations that include that tag. This also allows an easy comparison of how well each tag is clustered, and whether or not the combination of tags affects the clustering for each tag. These are discussed at length in Appendix A.

5.1.3 K-means clustering

K-means clustering is also used to cluster the sounds. The algorithm is run until the total change in the centroid distances is less than 0.001. Within each cluster, the number of each different tag combination is counted, and this is sorted by tag and written to a file.

5.2 Results and Discussion

The SOM and K-means clustering give a general indication of how the samples should be divided. However, the divisions are imperfect as the sample set is quite complex. The division of the samples changes as configuration of the classifier and the input changes.

In general, the number of groups in which a tag appears in the K-means clustering increases as the number of autocorrelation lags increases, and as the total number of groups increases. This makes sense intuitively since the group spread would logically increase as the total number of groups increases, since there are more divisions between the groups. The increase in the group spread as the number of autocorrelation lags increases likely occurs because none of the sounds tested are excessively reverberant. Hence, it is likely that in the larger autocorrelation matrices, the correlation of the later values is quite low for the majority of the samples. The later lags of all the samples would be similar, making grouping more difficult.

A similar trend is seen in the SOM tests, where the spread of each tag increases

as the number of autocorrelation lags increases. Both the 50×50 and the 100×100 maps are large enough to capture the clustering of the tags. Even the smaller 50×50 map has more nodes than input vectors, and the 100×100 map tends to have large areas that are essentially unused and covered by the same input vector.

Using the information from both the SOM and the K-means clustering gives a good indication of which tags can be combined and which form their own clusters. A detailed discussion of each tag can be found in Appendix A.

The results show that although in-car, in-bus and in-train sounds are similar, traffic is actually not very similar to in-car sounds and is therefore grouped separately. Car turn signal do not affect the car noise, and a combination of car noise and music tends to be dominated by the car noise. Hence, car noise and music combination samples are grouped with car noise.

The "nature" category is actually made of a number of subcategories that should be grouped separately: birds, water washing and water running. Similarly, sounds containing babble do not form a coherent cluster. Hence, babble sounds are clustered into three sub-categories: office noise, restaurant noise and shopping noise. These categories are sufficiently changed by secondary sounds, therefore samples that have a combination of the babble sounds and a secondary noise are clustered with the primary babble sound.

The applause category is well clustered, but is not a sufficiently common sound to warrant its own group. Electronic sounds are not well clustered, and are not considered sufficiently important to group separately. These are grouped with the primary sounds where appropriate. Industrial sounds are also not well clustered.

These sounds are not used for training as it is unlikely that a hearing-impaired person would wear a hearing aid in such a location.

The following nine classes are therefore used in this thesis:

1. in-car, in-bus, in-train, car signal
2. traffic, in-subway, subway station
3. birds, dog, farm animals
4. water washing
5. water running
6. office noise, phone ring
7. restaurant noise
8. shopping noise
9. music

Chapter 6

Feature Selection

6.1 Methods

This section discusses the features tested and the algorithms used for feature selection. Section 6.1.2 describes each of the features tested. These features are all taken from literature and have been tested by other authors for audio classification problems. Section 6.1.3 presents the implementation of the feature selection algorithm.

The selection of input features is based on the output classes. A number of different features are selected for testing from literature. Each feature is extracted from each audio file in the updated database. These features are then passed to a feature selection algorithm that selects some appropriate features. This is then used to generate the final file for testing. This process is illustrated in Figure 6.1

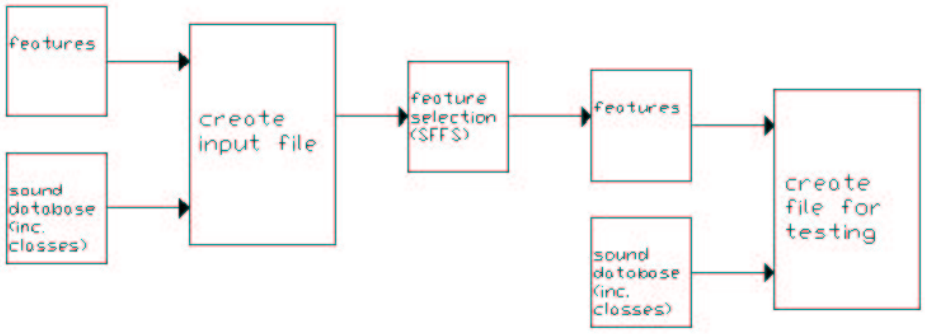


Figure 6.1: Procedure for the selection of input features

Table 6.1: Summary of the samples used for feature selection

tag	samples	recordings
in-car	50	7
traffic	50	11
birds	50	6
water washing	50	1
water running	50	1
shopping	50	4
restaurant	50	5
office noise	50	1
music	50	10

6.1.1 Data Set

The data set for the feature selection is also taken from the British Broadcasting Corporation CDs. The sound samples are manually re-tagged as being part of one of the classes defined in the class selection process and a data set is chosen. Please see Section 5.2 for more information on the classes used. The data set consists of 50 samples from each class. A summary of the samples used is presented in Table 6.1

6.1.2 Features

Each feature is extracted from a 1 s sample of background noise. Because some systems track time-based changes, some require that more than one set of features be extracted from the sample. In this case, the sample is subdivided into smaller time segments and a feature is extracted from each time segment. Since the largest set of features tested is five, the smallest time segment used to calculated a feature is 200 ms. A study by Kates [10] uses 200 ms as a sample time for calculation of features because it is approximately the minimum time required for a human to register a change in loudness. The time required for a person to detect signal fluctuation is much smaller. Hence, a feature calculation time of 200 ms or greater should be sufficient for a feature to capture signal changes at least as well as a human.

The features tested in this work are described below.

6.1.2.1 Mean Frequency, Centre of Gravity, Spectral Centroid

Mean frequency, centre of gravity and spectral centroid are common terms for the same feature. This feature is used a number of different studies [10, 37, 35]. Buchler also uses the fluctuations of the spectral centre of gravity [37].

The mean frequency is the first moment of the log frequency spectrum. It gives a general description of the frequencies in which the majority of the signal is contained. The mean frequency [10] is calculated as:

$$F_{mean} = \frac{\sum_{k=1}^{N/2} |F_k|}{\sum_{k=1}^{N/2} \frac{|F_k|}{f_k}} \tag{6.1}$$

where F_{mean} is the mean frequency as an FFT bin index, f_k is the frequency at index k, $|F_k|$ is the magnitude of the frequency response at index k, and N is the number of samples in the frame.

6.1.2.2 High and Low Frequency Slopes

This feature vector is described by Kates [10]. The high and low frequency components are separated by the mean frequency. The slopes of the high and low frequency components give a general description of the shape of the spectrum about the mean. The slope of both the low and high frequencies is determined by least-squares fit to the log frequency response [10]. Both are given in the form:

$$y(k) = a_0 + a_1 log_2(k) \tag{6.2}$$

where a_0 is a constant, and a_1 gives the slope in dB/octave.

For the low frequency [10], parameters a_0 and a_1 are calculated as:

$$\begin{bmatrix} a_0 \\ \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{L} \frac{1}{k} |F_k|_{db} \\ \\ \sum_{k=1}^{L} \frac{1}{k} |F_k|_{db} log_2 k \end{bmatrix} \tag{6.3}$$

where k is the FFT bin index, $|F_k|_{db}$ is the magnitude in dB of the FFT bin index, and L is the FFT bin index just below the mean frequency.

The high frequency slope is calculated in a similar manner, but using the frequencies above the mean.

6.1.2.3 Envelope Modulation, Spectral Modulation, Modulation Depth

Envelope modulation, spectral modulation and modulation depth describe very similar features in a number of studies. In [10], Kates uses the term "envelope modulation", in [41] Nordqvist and Leijon use the term "spectral modulation", and in [37], Buchler et. al. describe the "modulation depth" in three bands, from 0-4 Hz, from 4-16 Hz and from 16-64 Hz. All of these features describe how the amplitude of the signal changes over time.

The most thorough description of this feature comes from Kates [10]. The envelope modulation describes the change in the magnitude of the signal over time. This feature was originally intended to describe the changes in each frequency band. However, Kates determined in his study that this was not significantly better than the average modulation across all bands [10]. The envelope modulation therefore describes how the mean magnitude changes over time.

A short (200 ms) background noise segment is required to determine the envelope modulation. This 200 ms segment is further divided into 6.4 ms segments, starting 3.2 ms apart, which gives a 50% overlap in the smaller segments. Both of these numbers come directly from the Kates study [10], and are based on human auditory perception. An average human takes 3.5 to 10 ms to detect a gap in a noise segment, depending on the person, the frequency and whether the noise is wideband or narrowband. A time segment of 6.4 ms falls directly in this range. Similarly, 200 ms is used because it is approximately the time required for a human to properly detect the loudness of a signal.

The average magnitude across all frequencies is calculated for each 6.4 ms seg-

ment. The mean magnitude and the standard deviation are calculated across the time segments, and the envelope modulation is given as the mean magnitude over the standard deviation measured across the segments.

This is calculated as:

$$m_{env} = \frac{\mu}{\sigma} \tag{6.4}$$

where m_{env} is the envelope modulation, μ is the mean of the segment means, σ is the standard deviation of the segment means, and where μ is calculated as

$$\mu = \frac{1}{M} \sum_{i=1}^{M} \mu_{6.4i} \tag{6.5}$$

and

$$\sigma = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (\mu_{6.4i} - \mu)} \tag{6.6}$$

where M is the number of 6.4 ms segments and $\mu_{6.4i}$ is the mean magnitude of the 6.4 ms segment i.

6.1.2.4 Linear Prediction Coefficients

Several studies use the linear prediction coefficients (LPCs) directly as a feature [47, 35]; however, many studies choose instead to use the cepstral coefficients derived from the LPCs [33, 43, 48]. Please see Section 6.1.2.5 for more information on LPC derived cepstral coefficients. Peltonen et. al. use both the LPCs and the cepstral coefficients [35].

The LPCs are the coefficients that would be used in a linear predictor filter. A linear predictor attempts to predict the next sample from M previous samples.

The LPCs are derived using the Wiener-Hopf [56] equations as:

$$R\overrightarrow{W_f} = \overrightarrow{r}$$

where R is the MxM autocorrelation matrix of the previous M samples, $\overrightarrow{r}$ is the 1xM cross-correlation of the previous sample with the past M samples, and $\overrightarrow{W_f}$ are the M LPCs. The LPCs can be easily calculated in MATLAB using the built-in function [57].

This feature was tested using LPCs of orders two to six.

6.1.2.5 Cepstral Coefficients

The word cepstrum is taken from the word 'spectrum' by reversing the first four letters. Cepstral coefficients are very common for speech processing and therefore also likely have some application in environment classification. The cepstral coefficients, however, are closely related to both the LPCs and the Mel-Frequency Cepstral Coefficients (MFCCs). MFCCs are discussed further in Section 6.1.2.6.

A cepstrum is found by taking the inverse Fourier transform of the log of the absolute value of the Fourier Transform of the system [58]:

$$c(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} log|S(e^{j\omega})|e^{j\omega n}d\omega$$

where $c(n)$ are the cepstral coefficients, and $S(e^{j\omega})$ is the fourier transform of the signal.

This feature is tested using 16 and 32 bands.

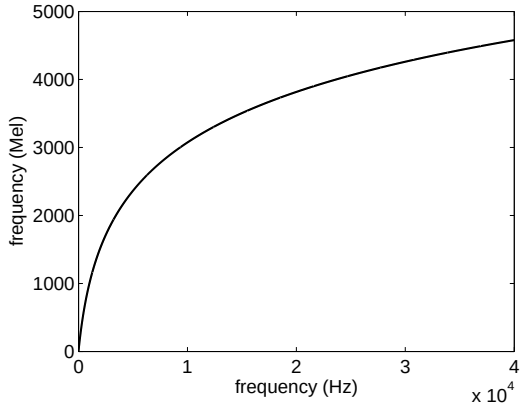


Figure 6.2: Frequency in Mel vs. frequency in Hz

6.1.2.6 Mel-Frequency Cepstral Coefficients (MFCCs)

The Mel-Frequency Cepstral Coefficients (MFCCs) are similar to the linear cepstral coefficients except that the log of the frequency response is first transformed into the Mel scale, using a triangular Mel-scale filterbank.

The Mel scale is a frequency representation of a signal that corresponds to the human auditory system, emphasizing the lower frequencies. Hence the MFCCs of a signal are the discrete cosine (DCT) or Fourier transform (FFT) of logarithm of the signal (dB representation) transformed into the Mel scale. Conversion to the Mel scale can be accomplished using the approximation [59]:

$$m = 2595 \log_{10} \left(1 + \frac{f}{700} \right) \tag{6.7}$$

where m is the frequency in Mels and f is the frequency in Hz. This produces the curve shown in Figure 6.2.

The MFCCs are the discrete cosine transform of the Mel-scaled log amplitude

of the frequency response. This is calculated in a number of steps.

1. Use an FFT to obtain the frequency response.
2. Take the log amplitude of the frequency response.
3. Transform to the Mel scale using the triangular Mel filterbank.
4. Take the DCT of the resulting signal.

The Mel filterbank is a bank of triangular filters used to transform the signal into N Mel components, where N is the number of filters in the filterbank. The filterbank is defined by the staring and ending frequencies, and the number of filters N. For this project, the starting frequency is 0 Hz, the ending frequency is the Nyquist frequency.

To create the filterbank, the starting and ending frequencies are transformed into Mel frequencies, using equation 6.7. N equally spaced centres are then placed between the start and end Mel frequencies. These are transformed back into Hz, and the centres are used to create overlapping triangular filters, as shown in Figure 6.3. The N Mel coefficients are obtained from the frequency response as [59]:

$$X_{mel}(i) = \sum_{k=1} M \log_{10} |X_{hz}(k)| f(i,k) \tag{6.8}$$

where $X_{mel}(i)$ is the i^{th} Mel coefficient and $1 \leq i \leq N$, k is the frequency bin index and M is the number of frequency components (order of the FFT), $|X_{hz}(k)|$ is the magnitude of the frequency response at bin index k, and $f(i,k)$ is the i^{th} triangular Mel filterbank filter at frequency bin k.

This feature is tested using 16 and 32 bands.

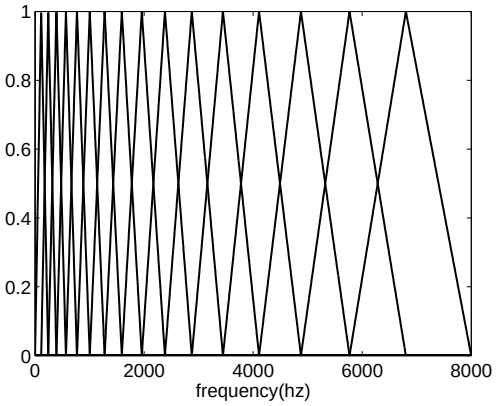


Figure 6.3: Mel filterbank with 16 filters and sampling frequency of 16 kHz

6.1.2.7 Onset features

Buchler et. al. [37] describe a number of onset features used in their paper and note that having at least one onset feature improves the accuracy of the classifier.

An onset occurs when the magnitude of a band increases by more than a set threshold value between two frames. These are counted in discrete frequency bins. The conversion to dB is calculated as follows:

$$|X_{dB}(i,t)| = 20\log_{10}\left(\frac{|X(i,t)|}{|X_0|}\right) \tag{6.9}$$

where $|X_{dB}(i,t)|$ is the magnitude of the frequency component in frequency bin i at time t in dB, $|X(i,t)|$ is the magnitude of the frequency component in frequency bin i at time t, and X_0 is a reference level. In this case X_0 is set to one for simplicity, because the onset is the difference between two time frames, and the X_0 terms will cancel.

Three different types of onset features are tested in this thesis. The number

of onsets feature gives the number of onsets counted within each bin. This gives one feature per bin. The onset strength is simply the magnitude of the onset in dB. The mean of the onset strengths is the mean across all frames and all bins, which gives a single feature for the whole time segment. Similarly, the variance of the onset strengths is also calculated across all bins and times, also giving a single feature.

All three onset features are assessed using onset strengths of 5 dB, 7 dB and 10 dB, and using 16 and 32 frequency bands.

6.1.2.8 Pitch and Tonality

The pitch is the measure of the major frequency components of the sound. Pitch is used by Pollastri and Simoncelli [60] in their study of composer classification, but no environmental classification studies have been found that directly use pitch as a feature. Buchler et. al. [37] use the tonality feature, but do not actually use the pitch value as part of the feature.

There are a number of different techniques that can be used to detect a pitch, including techniques based on the zero crossings, autocorrelation, cepstral coefficients or using maximum likelihood techniques. However, one of the most common techniques is the harmonic product spectrum (HPS) technique, which offers the benefit of being relatively simple and computationally inexpensive. A single note contains both the fundamental frequency and harmonic components that occur at integer multiples of the fundamental frequency. The idea behind HPS is that the fundamental frequency can be enhanced and more easily located by shifting the harmonic components back and multiplying the signals together. Because they oc-

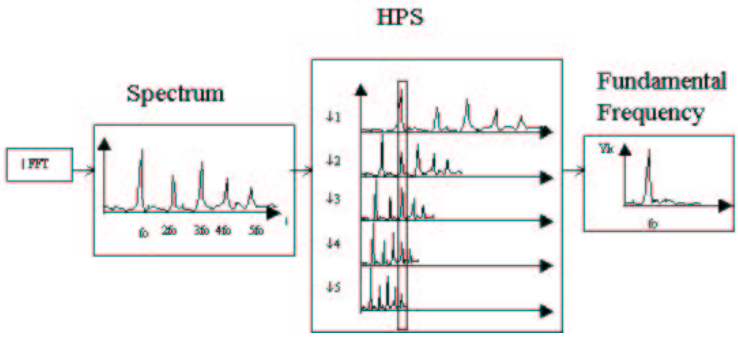


Figure 6.4: The harmonic product spectrum algorithm. Taken from [61]

cur at integer multiples of the signal, they can be shifted back to the fundamental frequency by downsampling.

The signal is first windowed and transformed into the frequency domain using an FFT. Then, multiple copies of the signal are down sampled from 1 to N, where N is the number of harmonics to be considered. These are multiplied together, and the fundamental frequency is found as the maximum frequency component. This is illustrated in Figure 6.4

In some cases, there is not a strong peak value, which indicates that there is no strong pitch in the signal. Buchler et. al. use a slightly different pitch detection method, based on the signal autocorrelation, but define a pitch to be a peak at 20% between 50 Hz and 500 Hz. A similar approach is taken in this study; however, the range is changed to be from 100 Hz to 1000 Hz. This change is made since 50 Hz is still quite a low frequency; especially since 60 Hz noise is common. The top of the frequency range is changed to 1000 Hz since 500Hz is actually quite low. For example, a concert-tuned A note is 440 Hz.

The threshold is set to be 20% of the signal energy. The signal energy is calcu-

lated from the FFT as:

$$E = \sum_{i=1}^{M} |X(i)|^2 \tag{6.10}$$

where M is the order of the FFT, and $|X(i)|$ is the magnitude of the frequency component at index i. Hence, there is a pitch if the magnitude of the frequency component at the pitch frequency is at least 20% of the signal energy.

Tonality is a feature based on the ratio of harmonic to inharmonic frames. A harmonic frame is a frame where there is a pitch, and an inharmonic frame is a frame with no pitch. In this work, this feature is changed to be the ratio of harmonic frames to the total number of frames in the signal. This is because it is possible to have a segment with no inharmonic frames, which would result in a division by zero.

In this study, pitch is also tested as a potential feature. If pitch is, in fact, not a useful feature for this type of application, then the feature selection algorithm will not select it.

These features are both assessed using three and five harmonics in the pitch calculation.

6.1.2.9 Zero Crossing Rate Features

The zero crossing rate (ZCR) is the number of times in a frame that the energy of a signal changes signs (positive to negative or negative to positive). The zero crossing rate itself is found using simple counting techniques.

The zero crossing rate was used directly as a feature by Peltonen et. al. for environmental classification [35] and Temko and Nadeu in the classification of meet-

ing room sounds [44]. Khan et. al. also used a ZCR feature (difference between the minimum and maximum ZCR) in their study of speech and music classification [47].

6.1.2.10 Spectral Rolloff

Peltonen et. al. [35] use the spectral rolloff point as a features. The spectral rolloff point is defined as the frequency below which a certain percentage of power spectrum is contained. This is calculated simply by finding the power spectrum and summing until the spectral rolloff point is reached.

This feature was tested using percentages of 80%, 85% and 90%.

6.1.2.11 Bandwidth or Spectral Width

Peltonen et. al. [35] included the bandwidth as a feature in their neural network and GMM classifier study. Buchler et. al. [37] also studied the spectral width as a feature.

The bandwidth is the frequency range that a certain percentage of the signal occupies. Starting from the low frequencies, the power values are summed. The lower limit of the bandwidth is found as the frequency index when the cumulative sum is:

$$B_{low} = \frac{1}{2}(1 - B_{\%}) \tag{6.11}$$

where $B_{\%}$ is the percentage of the signal used to determine the bandwidth. The

upper limit is found when the cumulative sum is:

$$B_{high} = B_{\%} + B_{low} \tag{6.12}$$

The bandwidth is found as the difference between B_{low} and B_{high}, normalized by the size of the FFT.

This feature was tested using percentages of 80%, 85% and 90%.

6.1.2.12 Line Spectral Frequencies

The line spectral frequencies (LSF) are discussed as a possible feature vector in the paper by Khan, Al-Khatib and Moinuddin [47]. The line spectral frequencies represent the same information as the LPC, but in a different form. The LSF are often used when the information in an LPC needs to be quantized, for example for transmission [62].

A polynomial of size M is called palindromic if

$$a_m = a_{M-m} \tag{6.13}$$

where a are the weights of the polynomial. It is called antipalindromic if

$$a_m = -a_{M-m} \tag{6.14}$$

If a polynomial has all of its roots on the unit circle, then it is either palindromic or antipalindromic. However, not all palindromic or antipalindromic polynomials have all their roots on the unit circle [62].

Additionally, any real polynomial of order M can be represented as a sum of a palindromic polynomial and an antipalindromic polynomial of order $M + 1$ [62]. The process of finding the LSF essentially consists of finding these two polynomials. If all the zeros of the polynomial are inside the unit circle, then the zeros of the palindromic and antipalindromic polynomials are on the unit circle and are interleaved with one root at 0 and one at π. Because the zeros are on the unit circle, the roots can be defined by their angles [62]. These occur in pairs at the positive and negative angles, with one pair at 0 and π, hence the LSF can be defined by a set of coefficients of size M. In MATLAB, these can be found simply using the "poly2lsf" function.

This feature was tested using LSFs of orders two to six.

6.1.2.13 Wavelets

A wavelet is a different type of basis function that is used to represent a signal. Whereas an FFT has bands that are uniformly spaced across all frequencies, wavelet coefficients are not uniformly spaced. Instead, the frequency space is subdivided so that there is a higher resolution in the higher frequencies [56]. This gives a good representation overall since there is actually more information in the higher frequencies, and also gives a good representation for hearing aids since human hearing is roughly logarithmic and also has higher resolution in the high frequencies.

A simple wavelet decomposition can be attained using a series of two-channel filterbanks (downsampled high and low-pass filters). The first stage of the wavelet comes from this filterbank. A second filter then further subdivides the high-pass portion of the signal, and the low-pass portion becomes the next stage of the wavelet

function. This is continued for the rest of the signal [56].

A study by Khan et. al. [47] uses the mean and variance of the wavelet transform as a feature in the classification of speech and music.

MATLAB provides functions to perform wavelet transforms. In this work, a simple Haar wavelet is used. The mean and variance of the wavelet transform are tested as features.

6.1.2.14 Percentage of Low Energy Frames

This feature is used by Khan et. al. [47] for the classification of speech and music. The energy of the signal can be found by taking the FFT of a frame, and then squaring the magnitude. A frame is considered to be low-energy if its value is below a certain percentage of the average. For this work, this feature was tested using a threshold of 10%, 15% and 20% of the average.

6.1.2.15 Short-Time Energy

The short-time energy of the spectrum is the total energy from the whole spectrum within one frame. Peltonen et. al. [35] use the short-time energy averaged across a number of frames, whereas Temko and Nadeu [44] use the short-time energy from every frame. These are, however, essentially the same feature. Bugatti et. al. use the minimum short term energy from a number of frames as a feature.

6.1.2.16 Band Energy Ratio, Subband energies

The feature set used by Peltonen et. al. [35] includes the band energy ratio, which is the ratio of the energy in one band to the energy of the entire spectrum. Temko and

Nadeu [44] use the subband energies directly (not as a ratio), but this essentially gives the same feature. For this feature, the signal is first passed though an FFT, and then the signal is divided into a certain number of bands by averaging the frequencies in the band. This feature was tested using 16 and 32 bands.

6.1.2.17 Autocorrelation

The autocorrelation of a signal is a measure of the similarity of past samples to the current sample. Because the clustering is based on the autocorrelation of the signal, it would make sense that the autocorrelation would be able to properly separate the samples. The autocorrelation is calculated as:

$$R = E[\overrightarrow{u}(n-1)\overrightarrow{u}^{H}(n-1)] \tag{6.15}$$

where R is the autocorrelation, $\overrightarrow{u}(n-1)$ is the sample at time $n-1$, E denotes the expectation operator, and H denotes the Hermitian transpose. The expectation is approximated as the average across the segments.

This feature is tested using 5, 10, 15 and 20 lags.

6.1.3 Feature Selection

In this thesis, the features are selected using the SFFS algorithm (please see section 2.4.3).

Fisher's interclass separability criterion and the Euclidean distance are both tested as the distance measure for the SFFS.

Because Fisher's interclass separability criterion requires inverting a matrix,

there is a possibility that a certain set of features will not result in an invertible matrix and it will not be possible to find the significance of the feature. In this case, the function returns a very low J value to discourage the use of this feature, since a non-invertible matrix is normally caused by having features that are linearly dependent. Since this means that the features are redundant, it is not desirable include both features in the final set.

When using the Euclidean distance, the distance measure is calculated as the average distance between the class means.

For features that have more than one value (eg. a 16-bin MFCC), each part of the feature is assessed individually. This is because it is not necessarily the case that all parts of the feature are equally important. In this way, it is possible to have a feature that is only one part of a larger feature. For example, it may be possible to have a feature that is one bin of a 16-bin MFCC.

The SFFS is first run to find feature sets of three to five features. This will give a reasonably sized model that can easily be implemented in a hearing aid. These features are tested in all the candidate classifiers. SFFS is then used to find sets of six to nine features, which are tested with the K-NN and the best classifier from the three to five sets.

6.2 Results and Discussion

6.2.1 Fisher's Interclass Separability Criterion

There are several difficulties with using SFFS with Fisher's interclass separability criterion. There are several features with a very small amount of scatter, and little separation between the classes. These features tend to be selected by the feature selection algorithm because the within-class scatter is very small.

There also appears to be some problem with this criterion when dealing with redundancy. Because the large majority of these features are related to each other in some way, there are a large number of feature sets that result in non-invertible matrices. It was originally thought that there would be a relatively small number of these matrices, hence these cases are simply trapped and given a very low score. However, even with only three or four features, the majority of the features sets become non-invertible. In two of the five-feature sets the final set is actually non-invertible. The feature set contains the first two features in the set, which are likely part of the set simply because of their placement.

It is clear from these results that using Fisher's interclass separability criterion with SFFS does not result in a workable set of features. Hence, the features selected using this distance metric are not tested with the classifiers.

6.2.2 Euclidean Distance

The features selected by SFFS using Euclidean Distance are presented in Table 6.2. These features are more reasonable than the features selected with Fisher's

interclass separability criterion. The features selected using Euclidean distance are mostly nested, with the exception of the single-feature frame going from three to four features. The features are almost all autocorrelation features, and one onset feature that appears in every set. Unfortunately, the set includes four of the same autocorrelation features with different numbers of lags (5, 10, 15 and 20). These are actually the same feature. Because this distance measure includes no measure of redundancy, the same autocorrelation feature (autocorrelation at five lags) is picked out up to four times in the same set.

The original sets are tested with the classifiers; however, these features are clearly not the ideal set of features. Hence, the redundant features are removed from the set manually and the SFFS is re-run without the redundant features included. The features selected from these new sets are presented in Table 6.3.

The actual distances between the class means are presented in Figures 6.5 and 6.6. In both cases, the average distance between the classes decreases as the number of features increases. This is likely because the best features are added first, and adding more features actually decreases the distance between the classes. The distance also decreases slightly as the number of samples in the window decreases.

The distances for the feature sets that include the redundant features are larger than the distances for the feature sets without repeated features, simply because including a feature with a large distance twice will obviously result in a larger distance than a smaller distance feature.

Overall, the distances between the class means for the features selected with SFFS are much larger than the average distance for the Kates' vector. The average

Table 6.2: Features selected using SFFS and Euclidean distance

	number of features		
window	3 features	4 features	5 features
1	no. onsets 10 dB 16 bands 6 autocorr 5 lags 4 autocorr 5 lags 5	no. onsets 10 dB 16 bands 6 autocorr 5 lags 5 autocorr 10 lags 5 autocorr 15 lags 5	no. onsets 10 dB 16 bands 6 autocorr 5 lags 5 autocorr 10 lags 5 autocorr 15 lags 5 autocorr 20 lags 5
2 2	no. onsets dB 16 bands 6 autocorr 5 lags 5 autocorr 10 lags 5	no. onsets 10 dB 16 bands 6 autocorr 5 lags 5 autocorr 10 lags 5 autocorr 15 lags 5	no. onsets 10 dB 16 bands 6 autocorr 5 lags 5 autocorr 10 lags 5 autocorr 15 lags 5 autocorr 20 lags 5
3	no. onsets 10 dB 16 bands 6 autocorr 5 lags 5 autocorr 10 lags 5	no. onsets 10 dB 16 bands 6 autocorr 5 lags 5 autocorr 10 lags 5 autocorr 15 lags 5	no. onsets 10 dB 16 bands 6 autocorr 5 lags 5 autocorr 10 lags 5 autocorr 15 lags 5 autocorr 20 lags 5
4	autocorr 5 lags 4 autocorr 5 lags 5 autocorr 10 lags 5	autocorr 5 lags 4 autocorr 5 lags 5 autocorr 10 lags 5 no. onsets 10 dB 16 bands 6	autocorr 5 lags 4 autocorr 5 lags 5 autocorr 10 lags 5 no. onsets 10 dB 16 bands 6 autocorr 20 lags 5
5	autocorr 5 lags 5 autocorr 10 lags 5 autocorr 15 lags 5	autocorr 5 lags 5 autocorr 10 lags 5 autocorr 15 lags 5 no. onsets 10 dB 16 bands 6	autocorr 5 lags 5 autocorr 10 lags 5 autocorr 15 lags 5 no. onsets 10 dB 16 bands 6 autocorr 20 lags 5

Table 6.3: Features selected using SFFS and Euclidean distance with redundant features removed

	number of features		
window	3 features	4 features	5 features
1	no. onsets 10 dB 16 bands 6 autocorr 20 lags 4 autocorr 20 lags 5	no. onsets 10 dB 16 bands 6 autocorr 20 lags 4 autocorr 20 lags 5 autocorr 20 lags 6	no. onsets 10 dB 16 bands 6 autocorr 20 lags 4 autocorr 20 lags 5 autocorr 20 lags 6 autocorr 20 lags 7
2	no. onsets 10 dB 16 bands 6 autocorr 20 lags 4 autocorr 20 lags 5	no. onsets 10 dB 16 bands 6 autocorr 20 lags 4 autocorr 20 lags 5 autocorr 20 lags 6	no. onsets 10 dB 16 bands 6 autocorr 20 lags 4 autocorr 20 lags 5 autocorr 20 lags 6 autocorr 20 lags 7
3	no. onsets 10 dB 16 bands 6 autocorr 20 lags 4 autocorr 20 lags 5	no. onsets 10 dB 16 bands 6 autocorr 20 lags 4 autocorr 20 lags 5 autocorr 20 lags 6	no. onsets 10 dB 16 bands 6 autocorr 20 lags 4 autocorr 20 lags 5 autocorr 20 lags 6 autocorr 20 lags 7
4	autocorr 20 lags 4 autocorr 20 lags 5 autocorr 20 lags 6	autocorr 20 lags 4 autocorr 20 lags 5 autocorr 20 lags 6 no. onsets 10 dB 16 bands 6	autocorr 20 lags 4 autocorr 20 lags 5 autocorr 20 lags 6 no. onsets 10 dB 16 bands 6 autocorr 20 lags 7
5	autocorr 20 lags 4 autocorr 20 lags 5 autocorr 20 lags 6	autocorr 20 lags 4 autocorr 20 lags 5 autocorr 20 lags 6 no. onsets 10 dB 16 bands 6	autocorr 20 lags 4 autocorr 20 lags 5 autocorr 20 lags 6 no. onsets 10 dB 16 bands 6 autocorr 20 lags 7

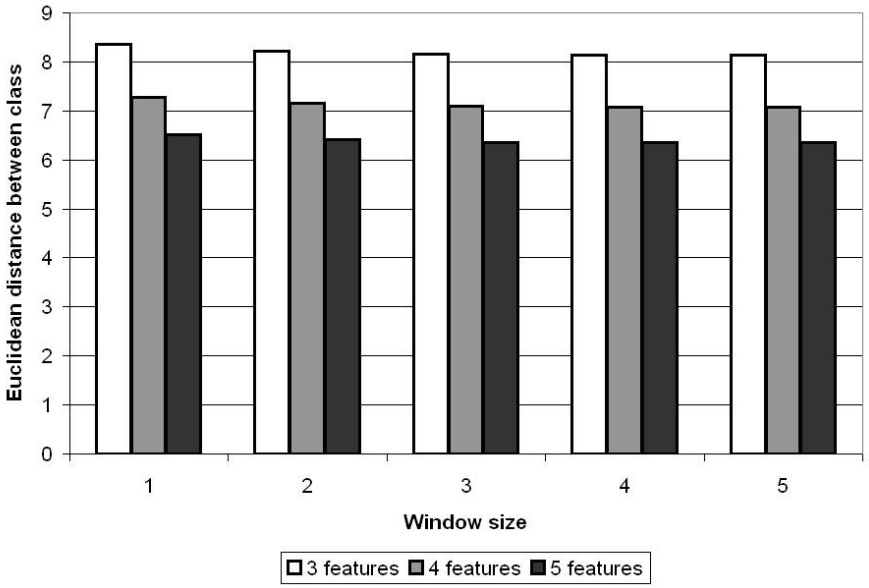


Figure 6.5: Euclidean distance between class means for features selected with SFFS using Euclidean distance with redundant features not removed

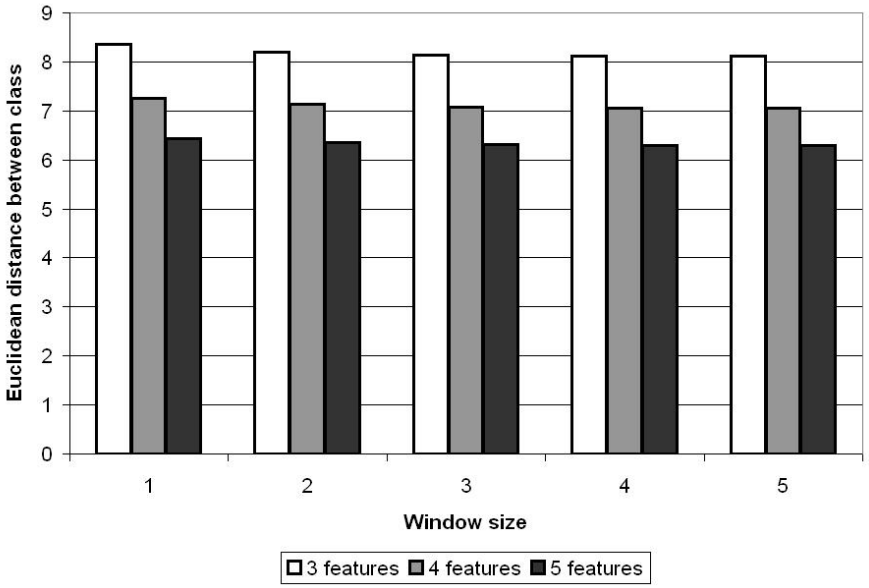


Figure 6.6: Euclidean distance between class means for features selected with SFFS using Euclidean distance with redundant features removed

Table 6.4: Additional features selected to create six to nine feature sets using SFFS and Euclidean distance with redundant features removed

	number of features			
window	6 features	7 features	8 features	9 features
1	no. onsets 10 dB 16 - 8	no. onsets 7 dB 16 - 12	autocorr 2	no. onsets 7 dB 16 - 13
2	no. onsets 10 dB 16 - 8	no. onsets 7 dB 16 - 12	no. onsets 7 dB 16 - 13	autocorr 2
3	no. onsets 10 dB 16 - 8	no. onsets 7 dB 16 - 12	autocorr 3	no. onsets 5 dB 16 - 13
4	no. onsets 10 dB 16 - 8	no. onsets 7 dB 16 - 12	no. onsets 10 dB 16 - 7	autocorr 8
5	no. onsets 10 dB 16 - 8	no. onsets 7 dB 16 - 13	autocorr 3	autocorr 8

Table 6.5: Euclidean distance between class means for six to nine feature sets selected with SFFS using Euclidean distance with redundant features removed

	number of features			
window	6	7	8	9
1	5.8521	5.3742	5.0023	4.6825
2	5.7745	5.2989	4.9228	4.6039
3	5.7284	5.2565	4.8716	4.5582
4	5.7091	5.2292	4.8042	4.5261
5	5.7011	5.2111	4.8223	4.5079

distance between class means for the Kates' vector is only 1.7584.

Sets of six to nine vectors are also selected for testing with the best classifier from the three to five feature sets. These sets are all nested, so the additional features added to the set are presented in Table 6.4.

The distances for the six to nine features sets are presented in Table 6.5. Similar to the three to five feature sets, the distance between the class means decreases as the number of features increases and as the number of sample windows increases.

The within-class scatter for Kates' feature vector and the SFFS vectors with the

Table 6.6: Scatter of different features sets, calculated as average distance of samples to class mean

	Kates	SFFS (redundant removed)						
t	f=4	f=3	f=4	f=5	f=6	f=7	f=8	f=9
1	0.35903	0.75623	0.93707	1.0611	1.2504	1.3919	1.5597	1.6515
2	0.35713	0.72381	0.89759	1.0314	1.2141	1.3631	1.5253	1.6241
3	0.33402	0.70762	0.87864	1.0244	1.2004	1.3562	1.5024	1.6351
4	0.36767	0.61453	0.88635	1.0021	1.1822	1.3631	1.5203	1.5916
5	0.36194	0.59827	0.86761	0.9798	1.1651	1.3152	1.4493	1.5169

redundant features removed. These are calculated by taking the average distance of each sample to the class mean. The results are presented in Table 6.6. From these results, it can be seen that the scatter for Kates' feature vector is smaller than the scatter for the SFFS vectors. The effect of the scatter on the models is discussed further in Section 7.2.4.

6.3 Summary

Although Fisher's interclass separability criterion is theoretically sound, it runs into practical problems with matrix inversion when used on a real set with more than one feature being evaluated at a time. Hence, the system is tested using the features selected with Euclidean Distance. However, Euclidean distance has problems with redundancy of the features, which are removed manually to produce the final feature set. However, this is clearly not an ideal solution to the problem.

Additionally, neither measure is able to account for the computational complexity of the features. Because the feature extraction will need to be implemented in hardware, the ideal selection criterion would account for the computational com-

plexity of the features. For example, selecting features that use a smaller number of bands, or features that require a smaller amount of time to evaluate.

The feature selected with Euclidean distance tend to be autocorrelation features, which is logical since these features are used in the class selection. This is also an indication that the class selection is logical, since the distance between the class means using autocorrelation features is large.

Chapter 7

Overall System Testing

This chapter discusses the final tests of the system, using the classes and features selected in Chapters 5 and 6 and the classifiers tested in Chapter 4.

7.1 Methods

The features selected in the feature selection are tested against the features used in Kates' study[10]. This provides a baseline to assess these features, and also allows for comparison against the initial tests with different classes. The algorithms are developed in C++ using the Microsoft .NET compiler.

7.1.1 Data set

The data set used for the final tests is the same data set used for the feature selection. Please see Section 6.1.1 for more information.

7.1.2 Features

The features are first clipped to within 1.5 standard deviations of the mean to avoid skewing the normalization with outlying samples. Next, the samples are normalized between zero and one.

Feature selection is performed separately for each sample window size. Hence, for the features selected with SFFS, each sample window size is tested with the features selected for that size.

7.1.3 K-folds Validation

The testing is performed using K-folds validation. In this type of testing, the entire data set is divided into K roughly equal sized parts. The tests are run K different times, each time using $K-1$ parts for training and one part for testing. This gives a better indication of the real performance because the testing and training sets are more varied.

For this thesis, a K of five is selected for the K-folds testing. For the MLP and the WMLP, the initialization is random, so there is more variation in the tests. Hence, each of the K-folds tests is run five times, for a total of 25 tests per model. The HMM initialization is based on a K-means clustering and is therefore less random. Hence, each K-folds test is run two times, for a total of ten tests per model.

7.1.4 Artificial Neural Network

The class selection process gives nine different classes (please see Chapter 5).

For the tests with the features selected by Kates [10], there are four inputs. The non-windowed MLP is tested with SFFS features selected from the single-sample window. It is tested with sets of three to five features selected using Euclidean distance.

The models are tested with four to twelve hidden nodes. Each model is trained for 10,000 epochs, as the initial tests show that this amount of training is sufficient (please see Section 4.2.2).

7.1.5 Hidden Markov Model

For the tests with the features selected by Kates [10], there are four inputs. The non-windowed MLP is tested with SFFS features selected from the five-sample window as this is the window size used for the HMM. It is tested with sets of three to five features selected using Euclidean distance.

Each model is tested with two to four states and four to seven codebook values. They are trained for ten iterations each, which is shown to be sufficient training in the initial tests (please see Section 4.2.3).

7.1.6 Windowed MLP

For the tests with the features selected by Kates [10], there are four inputs for each time delay. Hence, each MLP has $4W$ inputs, where W is the window size. Each network is tested with window sizes of two to five.

In the initial tests, the best number of hidden nodes in the non-windowed MLP is used to set the number of hidden nodes in the windowed MLP by maintaining

the same ratio of input to hidden nodes. However, it is not necessary to maintain the same ratio for the windowed tests, as this can become a large number of hidden nodes. Instead, each windowed MLP is tested with the same number of hidden nodes as the best non-windowed (10), the same ratio ($10W$) and a number midway between these points ($5(W+1)$). Although not every possible number of nodes is tested, this gives an indication of a good range for the hidden layer.

The non-windowed MLP is tested with SFFS features selected from the window size corresponding to the window size used in the WMLP. It is tested with sets of three to five features selected using Euclidean distance. The number of hidden nodes is varied similarly to the tests with Kates' feature vector, by using the same number of hidden nodes as the best run of the non-windowed, the same ratio and the midway point.

7.2 Results and Discussion

7.2.1 K-nearest Neighbours

7.2.1.1 Kates' feature vector

The results from the K-nearest neighbours classifier indicate that this test set is more difficult to properly classify than the initial test set. There are a few major reasons for this decrease in accuracy. Firstly, the number of groups has increased. This makes the task of classification more difficult because there are more ways that a sample could be misclassified. The classes used in the initial study were also quite distinct from each other, particularly the white noise class, which has a

very different profile than naturally occurring sounds. The new grouping of sounds contains classes that are relatively similar to each other. Additionally, the samples themselves are less clean, and contain combinations of different sounds. This can also make the classification more difficult as several samples may contain similar sounds.

The results of the test set are presented in Figure 7.1. The average accuracy denotes the average for that KNN across all sets tested using K-folds validation. The best-run is the best of the K-folds sets. The average accuracies on the testing set range from 57.1% to 67.1%. The best-run accuracies range from 62.2% to 73.3%. Similar to the initial tests, the performance on the training set decreases as the K value increases, for the reasons discussed in Section 4.2.1.

The highest average accuracy (67.1%) and the highest best-run accuracy (73.3%) both occur at a K value of three. However, all the K values up to eight have an average accuracy that is within 2.9%. Any relatively small K value would likely be a good choice for this application. The accuracy gradually decreases as the K value increases.

The best-run set of the 3-NN set is presented in Table 7.1. The sound of water washing is clearly one of the easiest to classify. All of the samples in the category are correctly classified, and only one sample is incorrectly classified as being in this category. There is some confusion with the traffic category being misclassified as in-car noise, which is likely because these two categories both contain a large amount of engine noise. There is also a fair amount of confusion between the shopping, office and restaurant noise, likely because all three of these categories contain a

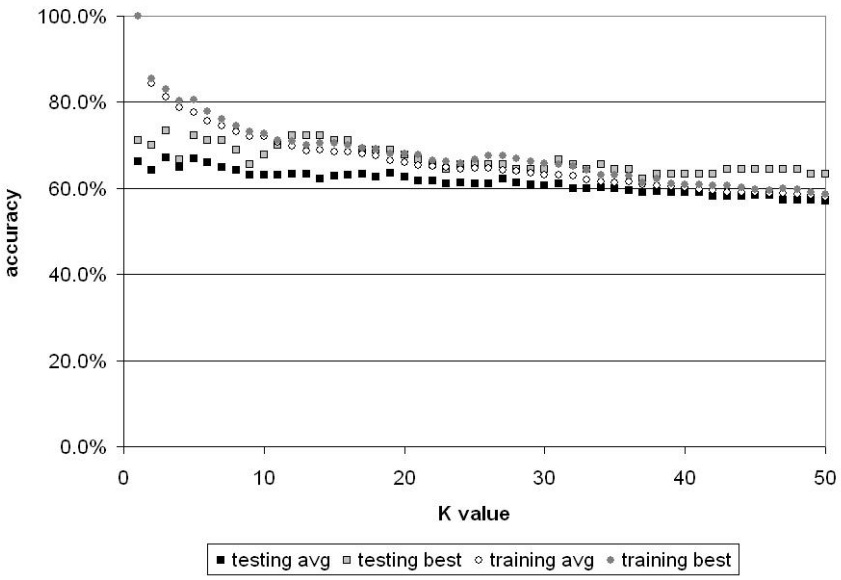


Figure 7.1: Accuracy of KNN with different K values using Kates' feature vector

	selected								
actual	in-car	traffic	birds	washing	running	office	restaurant	shopping	music
in-car	70.0%	0%	0%	0%	0%	10.0%	0%	20.0%	0%
traffic	30.0%	70.0%	0%	0%	0%	0%	0%	0%	0%
birds	0%	0%	80.0%	10.0%	0%	0%	0%	0%	0%
washing	0%	0%	10.0%	90.0%	0%	0%	0%	0%	0%
running	0%	0%	0%	0%	100%	0%	0%	0%	0%
office	10.0%	10.0%	0%	0%	0%	50.0%	20.0%	10.0%	0%
restuarant	0%	10.0%	20.0%	0%	0%	0%	60.0%	10.0%	0%
shopping	0%	10.0%	10.0%	0%	0%	0%	10.0%	70.0%	0%
music	10.0%	0%	10.0%	0%	10.0%	0%	0%	0%	70.0%

Table 7.1: Confusion matrix for the best-run 3-NN using Kates' feature vector

significant amount of babble noise.

7.2.1.2 Inputs selected with SFFS

The KNN results for the inputs selected with SFFS and without the redundant features removed are presented in Figure 7.2. The results for the four and five feature vectors are very similar because the feature added to create the five-feature vector is actually the same as one of the features in the four-feature vector (please

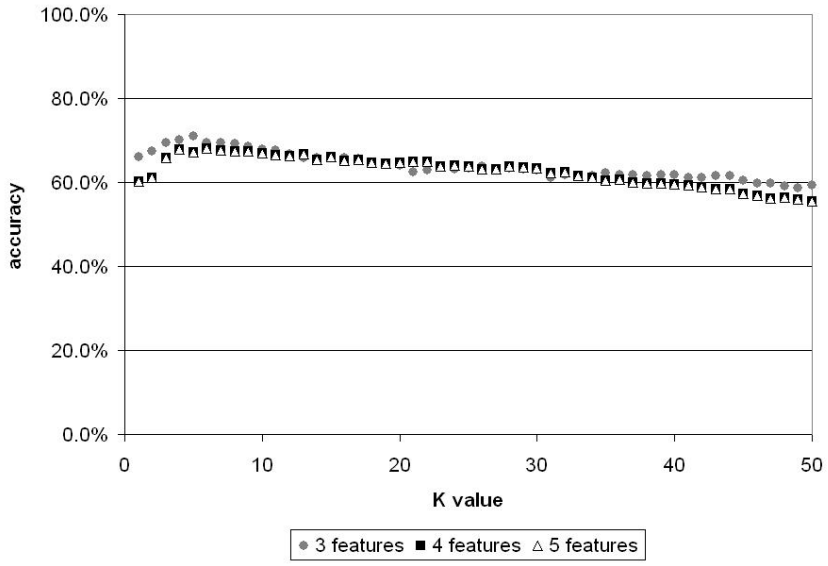


Figure 7.2: Average accuracy of KNN on the testing set with different K values using the input vector selected with SFFS using Euclidean distance with redundant features included

see Section 6.2.2 for further discussion). The results for the three-feature vector are actually higher than for the four and five feature vectors, which may be because the average distance between the classes is actually larger for the three-feature vector (please see Section 6.2.2).

The highest average accuracy for the three-feature vector occurs when the K value is five (71.1%), and the highest best-run accuracy occurs when the K value is four (75.6%). For the four and five-feature vectors, the highest average (68.0%) and best-run (77.8%) accuracies both occur when the K value is eight. Overall, however, any relatively small K value will likely give acceptable results. Because the results for the three-feature vector are better on average, the three-feature, 5-NN is used for comparison.

	selected								
actual	in-car	traffic	birds	washing	running	office	restaurant	shopping	music
in-car	90.0%	0%	0%	0%	0%	10.0%	0%	0%	0%
traffic	20.0%	30.0%	10.0%	0%	0%	20.0%	10.0%	10.0%	0%
birds	0%	10.0%	50.0%	10.0%	0%	10.0%	20.0%	0%	0%
washing	0%	0%	0%	100%	0%	0%	0%	0%	0%
running	0%	0%	0%	0%	100%	0%	0%	0%	0%
office	0%	0%	0%	0%	0%	90.0%	10.0%	0%	0%
restuarant	0%	0%	0%	0%	10.0%	10.0%	80.0%	0%	0%
shopping	10.0%	0%	10.0%	0%	0%	0%	0%	80.0%	0%
music	0%	10.0%	20.0%	0%	10.0%	0%	0%	10.0%	50.0%

Table 7.2: Confusion matrix for the best-run of the three-feature 5-NN using the feature vector selected with SFFS with redundant features

The confusion matrix for three-feature 5-NN is presented in Table 7.2. It is clear from this matrix that the KNN has some difficulty separating the traffic class. This is not unexpected as most of the features used are autocorrelation features, and the SOM has already shown that the autocorrelation for the traffic class is not always tightly clustered. Similarly, this model also has difficulty separating the bird class. The two water classes are very well separated, as is the office noise class. There is, however, some difficulty separating the music class. Overall, however, the KNN performs much better with this feature vector than with the Kates vector.

The results when the repeated features are manually removed are slightly better. In this set of tests, the best average accuracy for the three-feature set occurs using the 5-NN (71.1%) and the best average accuracy for the four and five-feature sets both occur using the 4-NN (72.0% and 72.4% respectively). The overall best-run accuracy for the three-feature set is the 4-NN (75.6%), the four-feature set is the 3-NN (77.8%) and the five-feature set is the 6-NN (77.8%). Overall, this set follows a similar pattern to the previous feature sets in that all of the relatively small K values have similar accuracies, and any relatively small K would be a suitable choice for this algorithm. The accuracy tends to decrease slightly as the K values

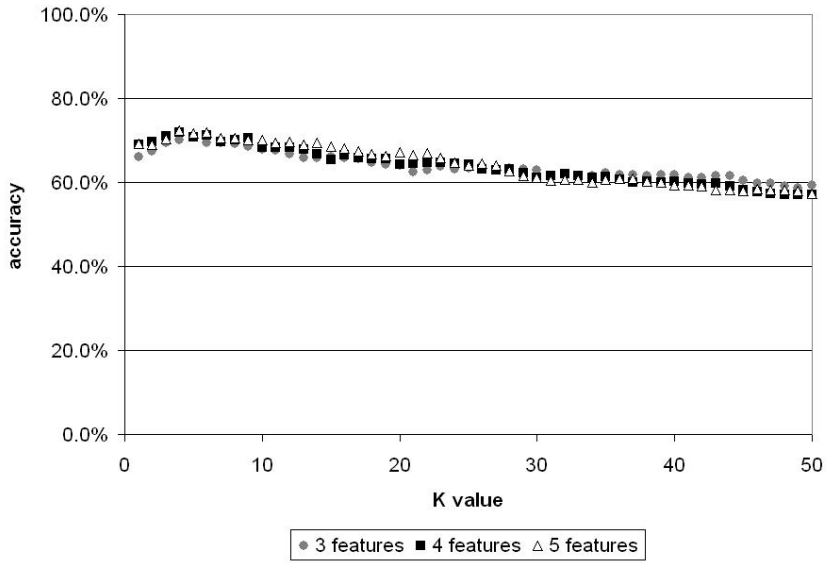


Figure 7.3: Average accuracy of KNN on the testing set with different K values using the input vector selected with SFFS using Euclidean distance with redundant features removed

are increased, as demonstrated by see Figure 7.3. Because the five-feature 4-NN gives the best average results, it will be used for comparison.

The confusion matrix for the best run of the five-feature 4-NN using the SFFS feature vector with the redundant features removed is presented in Table 7.3. Clearly the in-car, water washing and water running categories are well classified. These are also the most tightly clustered samples in the SOM and in the K-means clustering. Since similar features are being used, this is expected. The traffic category is not well classified, with spread across a number of different categories. Two of the samples are misclassified as in-car noise, which is likely because these samples both contain engine noise. Similarly, there is some confusion between the office, shopping and restaurant noises, likely because these all contain babble. The

	selected								
actual	in-car	traffic	birds	washing	running	office	restaurant	shopping	music
in-car	100%	0%	0%	0%	0%	0%	0%	0%	0%
traffic	20.0%	40.0%	10.0%	0%	0%	10.0%	10.0%	10.0%	0%
birds	0%	30.0%	60.0%	0%	0%	10.0%	0%	0%	0%
washing	0%	0%	0%	100%	0%	0%	0%	0%	0%
running	0%	0%	0%	0%	100%	0%	0%	0%	0%
office	0%	0%	20.0%	0%	0%	80.0%	0%	0%	0%
restuarant	0%	0%	0%	0%	0%	20.0%	80.0%	0%	0%
shopping	0%	0%	10.0%	0%	0%	10.0%	0%	80.0%	0%
music	0%	0%	0%	20.0%	10.0%	10.0%	10.0%	10.0%	40.0%

Table 7.3: Confusion matrix for the best-run of the five-feature 4-NN using the feature vector selected with SFFS with redundant features removed

music category is also difficult to properly identify, with samples being incorrectly identified as being in a number of different categories.

The KNN is also tested using larger feature vectors, also selected using SFFS with the redundant features removed. These results are presented in Figure 7.4. It can be seen in this figure that as the number of inputs is increased, the accuracy of the system also increases. This is not a large increase, but there is a pattern that indicates that increasing the number of inputs may also help other classifiers increase their accuracy.

The six-feature vector has the highest average and best-run accuracies with the 3-NN. Both the seven and eight feature vectors have the highest average accuracy with the 1-NN and the highest best-run accuracy with the 5-NN. The nine-feature vector has the highest average and best-run accuracy with the 5-NN. Similar to the other tests, all the relatively small K values give similar results, and any relatively small K would be appropriate.

The confusion matrix for the best run of the 5-NN using the nine feature vector is presented in Table 7.4. The in-car category and both of the water categories are very well classified. There is some difficulty classifying the traffic class, likely

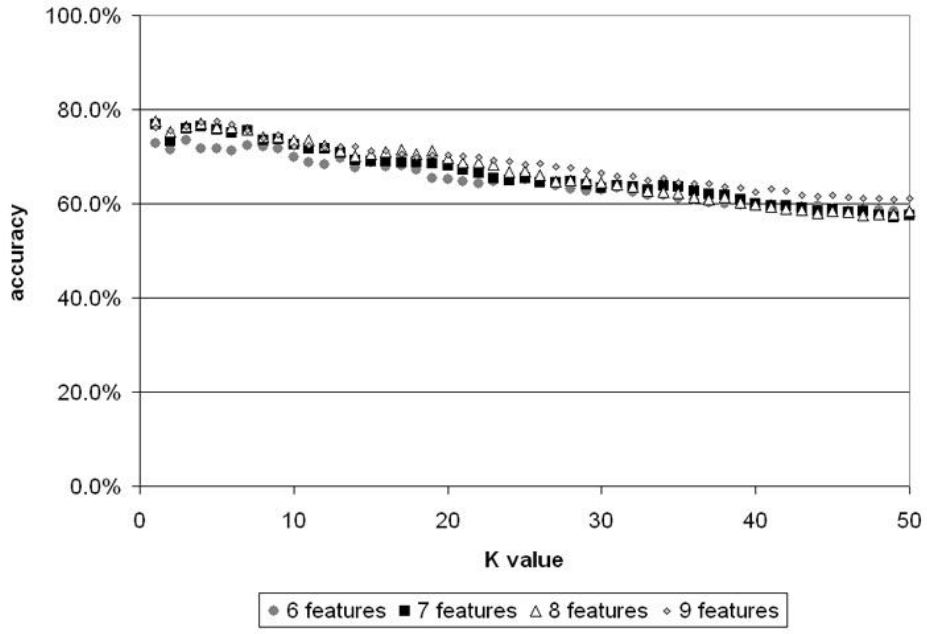


Figure 7.4: Average accuracy of KNN on the testing set with different K values using the higher order input vector selected with SFFS using Euclidean distance with redundant features removed

	selected								
actual	in-car	traffic	birds	washing	running	office	restaurant	shopping	music
in-car	100%	0%	0%	0%	0%	0%	0%	0%	0%
traffic	0%	70.0%	0%	0%	0%	10.0%	10.0%	0%	10.0%
birds	0%	0%	90.0%	0%	0%	0%	10.0%	0%	0%
washing	0%	0%	0%	100%	0%	0%	0%	0%	0%
running	0%	0%	0%	0%	100%	0%	0%	0%	0%
office	0%	20.0%	0%	0%	0%	80.0%	0%	0%	0%
restaurant	10.0%	0%	0%	0%	0%	10.0%	60.0%	20.0%	0%
shopping	0%	10.0%	0%	0%	0%	0%	10.0%	80.0%	0%
music	0%	0%	10.0%	0%	0%	0%	0%	10.0%	80.0%

Table 7.4: Confusion matrix for the best-run of the nine-feature 5-NN using the feature vector selected with SFFS with redundant features removed

because this category contains a fairly large number of recordings and combination sounds. It is also not as tightly clustered as other sounds in the SOM. There is also some difficulty classifying the restaurant noise class. These sounds tend to be misclassified as either office noise or shopping noise. This is likely because all of these sounds contain babble noise and the restaurant noise was actually clustered quite closely with these sounds.

7.2.1.3 Memory and Processing Requirements

Although the K-NN algorithm is logically simple, as the size of the training set increases, the algorithm becomes more computationally expensive. Because the input is compared to each training vector, the time required for computation is directly related to the number of training vectors. The comparison is performed on the basis of a distance measure; hence the computational complexity is also dependant on the size of the input vector. If each input has N features, and there are V training vectors, the K-NN classification will require NV multiplications, additions and subtractions to calculate the distance between the input and every training vector. It also requires V evaluations of a square-root function. Hence, the

complexity of the K-NN is $O(NV)$. Since each training vector needs to be stored, the algorithm also requires NV memory locations.

For these tests, 360 training vectors are used. When using Kates' feature vector, the number of inputs is four. Therefore, the total memory required is 1440 memory locations. Each input requires 1440 multiplications, additions and subtractions, and 360 evaluations of the square root function.

For the vector selected with SFFS with the redundant features not removed, the best feature vector has three inputs. This model therefore requires 1080 memory locations, 1080 multiplications, additions and subtractions and 360 evaluations of the square root function. When the redundant features are removed, the five-feature vector is the best model. This requires 1800 memory locations, 1800 multiplications, additions and subtractions and 360 evaluations of a square root function. For tests with larger feature vectors, the nine-feature vector is the best. This requires 3240 memory locations, 3240 multiplications, additions and subtractions and 360 evaluations of the square root function.

It is important to note, however, that these numbers are highly dependent on the number of training vectors.

7.2.2 Artificial Neural Network

7.2.2.1 Kates' feature vector

The new classes are first tested with the features suggested by Kates [10], which consists of the mean frequency, the slopes of the high and low frequency components and the envelope modulation. The accuracies for the models vary based on the

Table 7.5: Ranges of accuracies for different classifiers

model	feature vector	repeats?	num. features	testing avg.	testing best	training avg.	training best
MLP	Kates	N/A	4	52.2% - 63.4%	62.2% - 76.7%	57.3% - 78.0%	59.2% - 82.2%
MLP	SFFS	yes	3	51.5% - 64.2%	58.9% - 68.9%	54.3% - 73.6%	54.3% - 73.6%
MLP	SFFS	yes	4	52.0% - 59.4%	60.0% - 68.6%	53.2% - 65.2%	54.4% - 66.9%
MLP	SFFS	yes	5	51.9% - 58.9%	57.8% - 65.6%	53.1% - 65.1%	51.9% - 67.5%
MLP	SFFS	no	3	54.3% - 64.0%	58.9% - 68.9%	55.8% - 73.7%	59.4% - 74.7%
MLP	SFFS	no	4	52.5% - 63.7%	60.0% - 70.0%	55.6% - 75.3%	59.7% - 77.5%
MLP	SFFS	no	5	52.6% - 64.0%	58.9% - 70.0%	55.0% - 75.8%	57.8% - 78.9%
HMM	Kates	N/A	4	53.6% - 59.7%	57.2% - 68.3%	53.6% - 59.7%	57.2% - 68.3%
HMM	SFFS	yes	3	40.1% - 46.6%	41.4% - 51.7%	39.9% - 45.4%	40.6% - 47.2%
HMM	SFFS	yes	4	49.6% - 56.6%	52.8% - 62.8%	48.9% - 63.1%	52.2% - 63.1%
HMM	SFFS	yes	5	49.6% - 56.6%	52.8% - 62.8%	52.7% - 63.3%	56.7% - 64.4%
HMM	SFFS	no	3	40.1% - 46.5%	41.1% - 51.1%	61.2% - 76.5%	63.9% - 77.2%
HMM	SFFS	no	4	49.3% - 58.2%	51.7% - 63.3%	61.2% - 76.5%	63.9% - 77.2%
HMM	SFFS	no	5	47.9% - 56.0%	51.7% - 62.2%	53.0% - 63.2%	56.7% - 64.4%
WMLP	Kates	N/A	4	54.5% - 65.5%	71.1% - 82.2%	86.1% - 92.7%	80.5% - 95.8%
WMLP	SFFS	yes	3	31.8% - 64.1%	44.4% - 68.9%	47.8% - 80.8%	34.7% - 75.0%
WMLP	SFFS	yes	4	39.1% - 58.9%	55.5% - 68.9%	64.5% - 87.5%	64.5% - 87.5%
WMLP	SFFS	yes	5	36.4% - 58.5%	51.1% - 66.7%	64.8% - 85.6%	63.3% - 82.2%
WMLP	SFFS	no	3	43.8% - 57.9%	55.6% - 70.0%	73.8% - 86.8%	60.0% - 88.9%
WMLP	SFFS	no	4	45.6% - 59.6%	61.1% - 75.6%	82.8% - 91.6%	81.1% - 90.3%
WMLP	SFFS	no	5	45.6% - 58.0%	61.1% - 71.1%	82.9% - 93.3%	75.3% - 93.1%

number of hidden nodes used. The ranges of accuracies for the MLP using Kates' feature vector are presented in Table 7.5 on page 121.

The number of nodes used in the hidden layer changes the accuracy of the system. The accuracy with a small number of hidden nodes is quite low, likely because there are not enough weights to properly distinguish between all the categories. As the number of hidden nodes increases, the accuracy increases. The accuracy peaks at ten hidden nodes and then starts to decrease. Please see Figure 7.5. The average accuracy on the testing set for a model with ten hidden nodes is 63.3% and the best-run accuracy is 76.7%. The average accuracy on the training set is 75.5% and the best-run accuracy is 77.5%.

The trend of the number of hidden nodes is similar to the initial tests. In the initial tests a five node hidden layer is good for a four class MLP. Here, the best

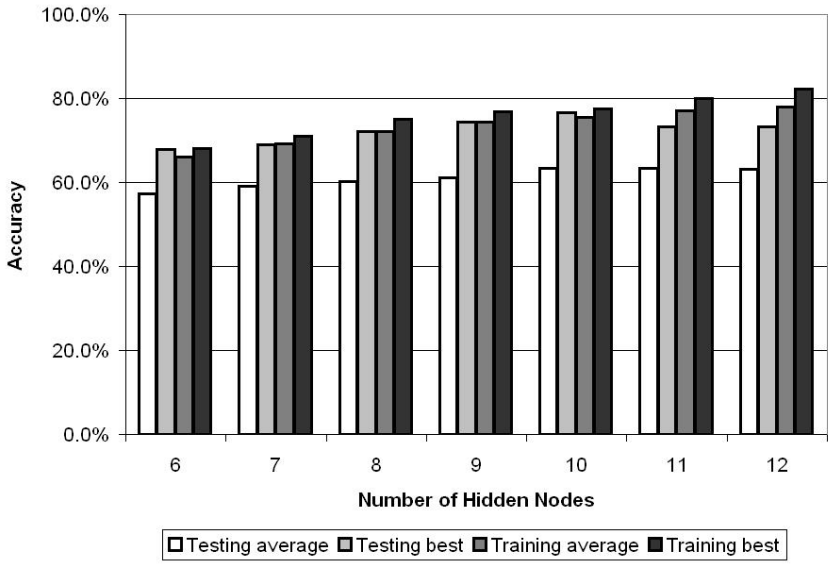


Figure 7.5: Accuracy of MLP with different number of hidden nodes using Kates' feature vector

number of hidden nodes tested is ten, which is also one higher than the number of output nodes.

The confusion matrix for the best run of the ten hidden node model is presented in Table 7.6.

7.2.2.2 Inputs selected with SFFS

The MLPs are trained on three to five feature sets selected with SFFS and Euclidean distance. In the first test, the MLP is trained on the sets that include redundant features, in the second test, the MLP is trained on sets that have the redundant features removed.

The ranges of accuracies for the MLP using the selected feature vectors are presented in Table 7.5 on page 121.

	selected									
actual	in-car	traffic	birds	washing	running	office	restaurant	shopping	music	unknown
in-car	70.0%	0%	0%	0%	0%	0%	0%	0%	0%	30.0%
traffic	0%	40.0%	0%	0%	0%	30.0%	0%	10.0%	0%	20.0%
birds	0%	0%	80.0%	0%	0%	0%	0%	0%	0%	20.0%
washing	0%	0%	10.0%	90.0%	0%	0%	0%	0%	0%	0%
running	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%
office	0%	0%	0%	0%	0%	90.0%	0%	0%	0%	10.0%
restuarant	0%	0%	0%	10.0%	0%	0%	70.0%	0%	0%	20.0%
shopping	0%	0%	30.0%	0%	0%	0%	0%	60.0%	0%	10.0%
music	0%	0%	0%	0%	0%	10.0%	0%	0%	90.0%	0%

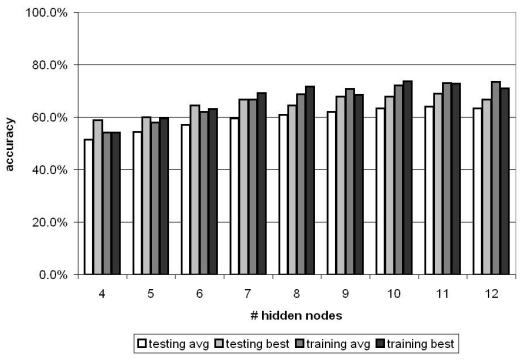
Table 7.6: Confusion matrix for the best-run of the ten node MLP using Kates' feature vector

For the three-feature set, the model using 11 hidden nodes gives the best average and best-run accuracy on the testing set and 12 nodes gives the best average accuracy on the training set. Please see Figure 7.6.

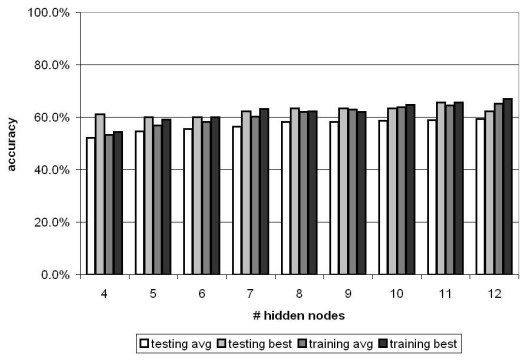
For both the four and five-feature sets, the using 12 hidden nodes gives the best average accuracy for the testing set. For the four-feature set, using 11 nodes gives the highest best-run accuracy. For the five-feature set, both the 11 and 12 node best-run accuracies are the same and 12 nodes gives the best average and best-run accuracy on the training set. Please see Figure 7.6.

The average accuracy of the 11 and 12 node models are very similar in all cases. The 11-node model is used for comparison as this is the best three-feature model and the three-feature model gives the best results out of the models tested. For the set that includes the redundant features, this is likely because the addition of the new features increases the complexity of the model without adding much additional information. Please see Figure 7.7. This also allows for a direct comparison with the models trained on Kates features [10].

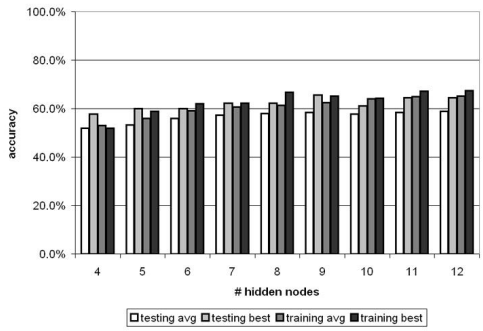
The confusion matrix for the best run of the three feature set MLP with 11 hidden nodes is presented in Table 7.7. The classifier is able to easily classify both



a) three-feature vector



b) four-feature vector



c) five-feature vector

Figure 7.6: Accuracy of the MLP for different SFFS feature vectors with repeated features using different numbers of hidden nodes

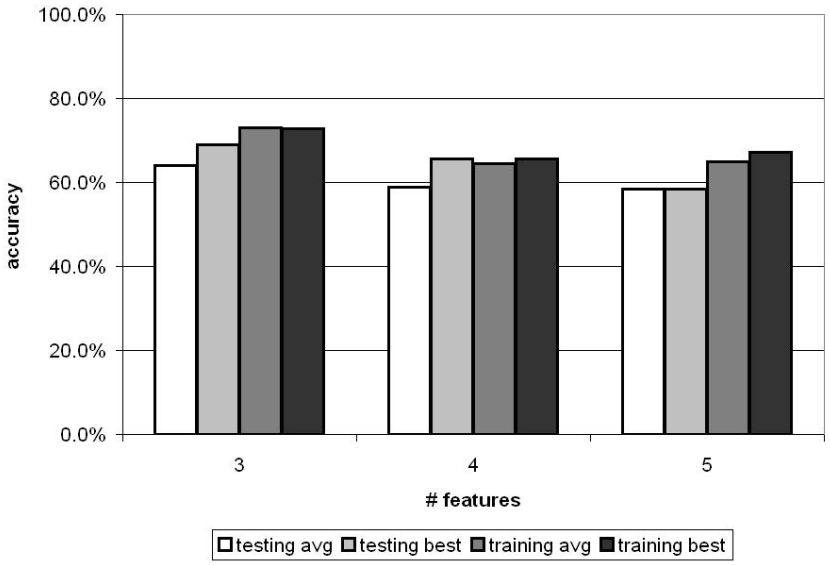


Figure 7.7: Accuracy of MLP with different sizes of SFFS feature vectors with the redundant features included

of the water categories. The in-car category is also quite well classified. Restaurant noise is often misclassified as being either office or shopping noise, likely because these all contain babble. The shopping and office noise categories also have a number of misclassifications. Many of these are classified as being unknown samples. The traffic category is not well classified, which may be because the traffic category contains so many different sounds. Many of these sounds are categorized as unknown, which may be an indication that the sample is being classified as being in more than one category. There is also some difficulty classifying the birds category. Both of these categories have a relatively large amount of spread in the SOM, which is likely why the classifier has difficulty in their classification.

For the tests using the feature vector with the redundant features manually removed, the average accuracy of the three feature testing set is highest with the

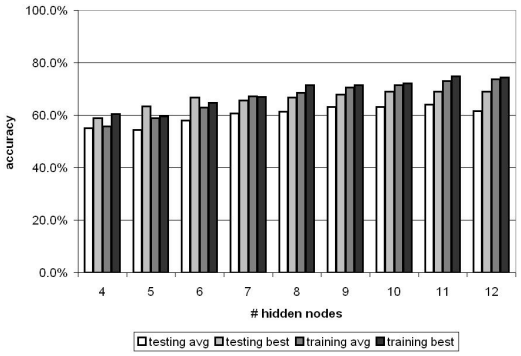
	selected									
actual	in-car	traffic	birds	washing	running	office	restaurant	shopping	music	unknown
in-car	80.0%	0%	0%	0%	0%	0%	0%	20.0%	0%	0%
traffic	0%	50.0%	0%	0%	0%	10.0%	0%	0%	0%	40.0%
birds	0%	0%	60.0%	0%	0%	0%	0%	0%	0%	40.0%
washing	0%	0%	0%	90.0%	0%	0%	0%	0%	0%	10.0%
running	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%
office	0%	20.0%	0%	0%	0%	50.0%	0%	0%	0%	30.0%
restaurant	0%	0%	0%	0%	0%	20.0%	50.0%	30.0%	0%	0%
shopping	0%	20.0%	0%	0%	0%	0%	0%	70.0%	0%	10.0%
music	0%	0%	0%	0%	0%	0%	0%	20.0%	70.0%	10.0%

Table 7.7: Confusion matrix for the best-run of the 11 node MLP using the features vector selected with SFFS that includes redundant features

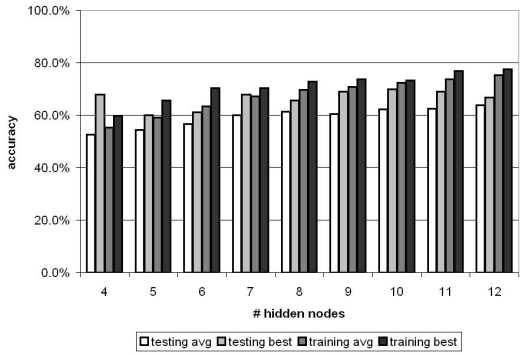
model that uses 11 nodes. The highest best-run accuracy is 68.9% and this accuracy is attained using the 10, 11 and 12 hidden node models. The average accuracy of the four feature testing set is highest with the 12-node model, and the best-run accuracy is highest with the 10-node model. The average accuracy of the five feature testing set is highest with the 12-node model, and the best-run accuracy is highest with the 11-node model. Please see Figure 7.8. The 10, 11 and 12 node models all give similar results and would all likely be acceptable models in a system. The 11-node model is used for comparison to facilitate comparison with the other models

The average accuracy of the three-feature set is better than the four and five-feature sets, but the difference is quite small. The best-run accuracies of the three and four features sets are the same, and the best-run accuracy of the five-feature set is slightly lower. Please see Figure 7.9. Hence, the three-feature set will be used for comparison.

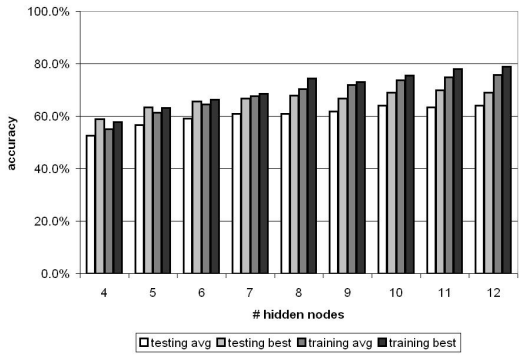
The confusion matrix for the three feature, 11 node MLP is presented in Table 7.8. From this matrix it can be seen that, similar to the MLP trained on the feature set with redundant features included, this classifier also has difficulty classifying the traffic and birds classes. It also has some difficulty with the restaurant class. The



a) three-feature vector



b) four-feature vector



c) five-feature vector

Figure 7.8: Accuracy of the MLP for different SFFS feature vectors with repeated features removed using different numbers of hidden nodes

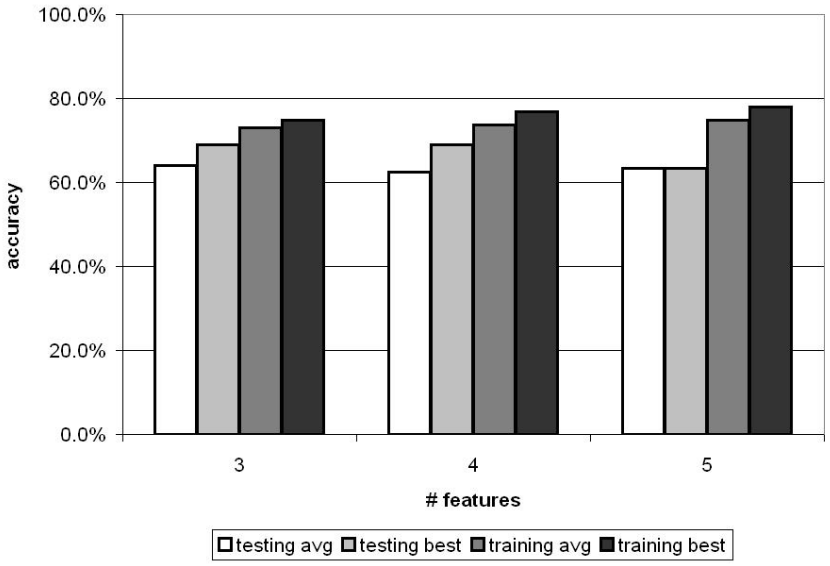


Figure 7.9: Accuracy of MLP with different sizes of SFFS feature vectors with the redundant features removed

in-car class and both the water classes are all very well classified. This classifier matches closely with the results from the SOM and K-means clustering, which is unsurprising as they use similar inputs.

The MLP is also tested with larger feature sets selected with SFFS. The average accuracy of the testing sets of feature vectors with six to nine features range from 66.3% to 68.2% and the best-run accuracies range from 72.2% to 74.4%. The training set average accuracies range from 84.4% to 86.6% and the best-run accuracies range from 84.7% to 85.3%. The accuracy increases as the number of features increases. These results are presented in Figure 7.10. This is an indication that adding more features may increase the accuracy of the models, and the model has not yet reached the point where the curse of dimensionality has become a problem. Starting from Kates' feature vector, adding more good features may increase the

	selected									
actual	in-car	traffic	birds	washing	running	office	restaurant	shopping	music	unknown
in-car	90.0%	0%	0%	0%	0%	0%	0%	0%	0%	10.0%
traffic	10.0%	40.0%	10.0%	0%	0%	10.0%	0 10.0%	0%	20.0%	
birds	0%	0%	40.0%	0%	0%	10.0%	10.0%	0%	20.0%	20.0%
washing	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%
running	0%	0%	0%	0%	90.0%	0%	0%	0%	0%	10.0%
office	0%	0%	0%	0%	0%	80.0%	0%	0%	0%	20.0%
restaurant	10.0%	10.0%	0%	0%	0%	0%	40.0%	0%	0%	40.0%
shopping	0%	0%	0%	0%	0%	0%	0%	90.0%	0%	10.0%
music	0%	0%	0%	0%	10.0%	0%	0%	20.0%	50.0%	20.0%

Table 7.8: Confusion matrix for the best-run of the 11 node MLP using the features vector selected with SFFS with redundant features removed

accuracy even further.

7.2.2.3 Memory and processing requirements

The memory requirements of the MLP are relatively small as compared to the other classifiers tested. An MLP with N_i inputs, N_h hidden nodes and N_o output nodes requires enough memory to store $(N_i N_h) + (N_h N_o)$ weights, and $N_h + N_o$ threshold values. Hence, the required memory size is:

$$Mem_{MLP} = N_h(N_i + N_o + 1) + N_o \tag{7.1}$$

Since the system will be pre-trained before it is implemented in an actual hearing aid, the classification system will only need to process the forward phase. Hence, the required number of multiplications and additions is:

$$P_{MLP} = N_h(N_i + N_o) \tag{7.2}$$

In a DSP this can be implemented in a relatively small number of cycles using the multiply-accumulate functions. Alternately, this could be implemented directly

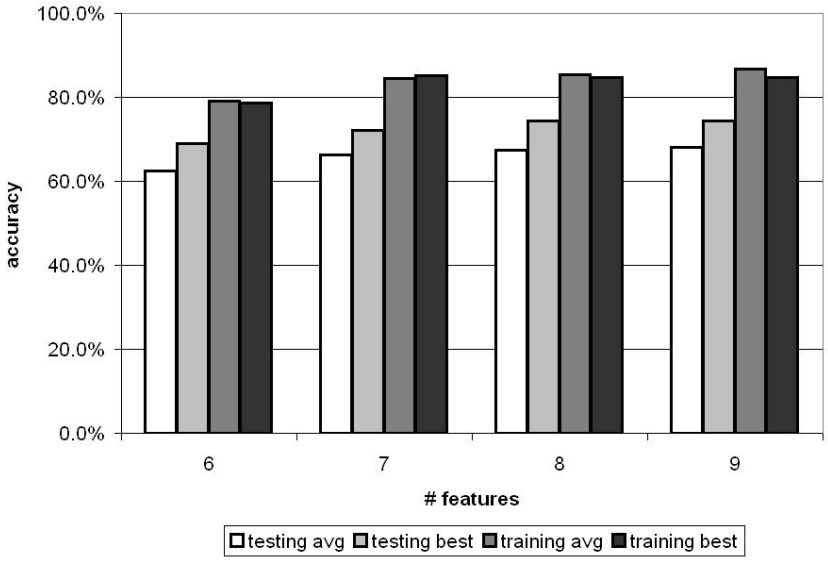


Figure 7.10: Accuracy of MLP using larger features vectors and 11 hidden nodes

in hardware and the nodes of each layer could be processed in parallel. Overall, the computational load and memory requirements of the MLP are small and could be easily implemented in a hearing aid chip. This makes the MLP a more suitable choice than the K-NN, even though the accuracy of the non-windowed MLP and the K-NN classifier are similar.

For the tests using Kates' feature vector, there are four inputs, nine outputs and the best model uses ten hidden nodes. This model therefore requires 149 memory locations and requires 130 multiplications and additions to process each input.

For the tests using the feature vector selected with SFFS with repeated features included, the best feature vector is the four-feature vector. The best model uses 11 hidden nodes, and there are four inputs and nine outputs. This model therefore requires 163 memory locations and 143 multiplications and additions.

For the tests with the redundant features removed, the best model also uses the four-feature vector and 11 hidden nodes. It therefore also requires 163 memory locations and 143 multiplications and additions.

For tests with larger features vectors, the best model uses the nine-feature vector and 11 hidden nodes. It therefore requires 218 memory locations and 198 multiplications and additions to evaluate each input.

7.2.3 Hidden Markov Model

7.2.3.1 Kates' feature vector

The results from the HMM are actually slightly worse than the results from the MLP and are much worse than the results obtained in the initial tests. The ranges of accuracies for the HMM tested with Kates' feature vector are presented in Table 7.5 on page 121.

The HMM still appears to be a more reliable model than the MLP. Within each set, the average and best-run accuracies for the testing set are all the same, which indicates that the HMM normally converges to a similar solution when trained on a certain training set and using the same codebook and number of states.

The number of states used in the model has a slight effect on the accuracy of the model, with three states giving the best accuracy. Please see Figure 7.11.

The number of codebook values used also has a small effect on the output. There are no definite trends with respect to the number of codebook values. In the three-state model, the highest average accuracy occurs in the model using seven codebook values. However, the highest best-run accuracy occurs in the six-codebook value

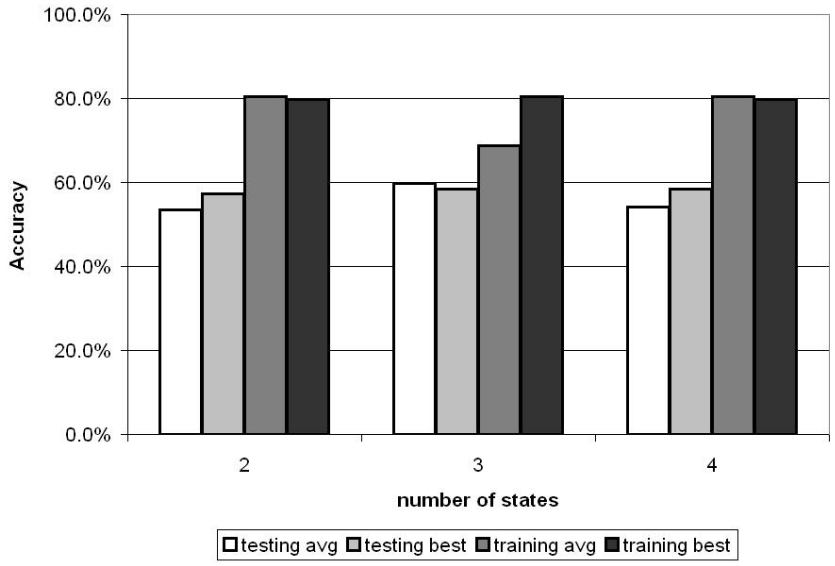


Figure 7.11: Accuracy of HMM with seven codebook values and different numbers of classes using Kates' feature vector

model. Please see Figure 7.12. For this thesis, the three-state model with seven codebook values is used for comparison. This model is chosen because the best-run accuracy on the testing set is closer to the average value, indicating that this model is slightly more reliable. Additionally, the accuracies on the testing set are closer to the accuracies on the training set, indicating that the model may generally be a better fit and may work with a larger number of possible input vectors.

The confusion matrix for the best run of the testing set of the three-class, seven-codebook value HMM is presented in Table 7.9. It is clear from this table that the HMM has significant difficulty identifying samples in the traffic category. A large number of these samples are misclassified as office noise or shopping noise. This may be because some of the traffic samples also contain babble noise. However, it appears that a large number of the errors in the HMM are misclassifications of a

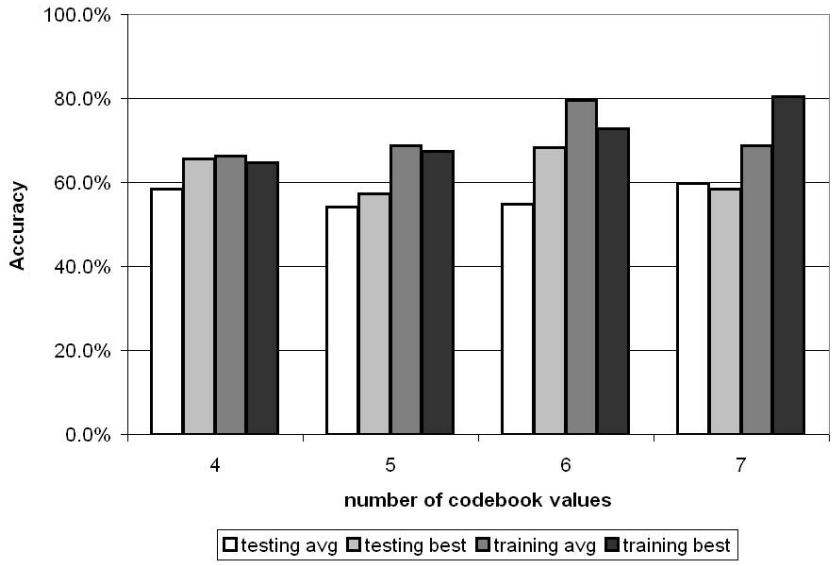


Figure 7.12: Accuracy of HMM with three classes and different numbers of codebook values using Kates' feature vector

sample as shopping noise, so this may also be simply that the HMM has difficulty training this category. Similarly, the matrix also has some difficulty distinguishing the in-car category, with many samples being misclassified as either restaurant or shopping noise, and one sample being misclassified as traffic. It is unclear why samples would be misclassified as shopping or restaurant noise. However the traffic and in-car categories contain similar sounds, which may explain this misclassification.

The water washing and water running categories are very well classified. This is expected from their tight clustering in the SOM and K-means tests. The office noise category is also quite well categorized, with one sample being misclassified as restaurant noise. However, these categories are all taken from a relatively small number of recorded tracks, which may also explain these good results.

There is some confusion between the restaurant and shopping noise categories,

	selected								
actual	in-car	traffic	birds	washing	running	office	restaurant	shopping	music
in-car	50.0%	10.0%	0%	0%	0%	0%	20.0%	20.0%	0%
traffic	0%	20.0%	0%	0%	0%	50.0%	0%	30.0%	0%
birds	10.0%	0%	60.0%	0%	0%	0%	0%	30.0%	0%
washing	0%	0%	10.0%	90.0%	0%	0%	0%	0%	0%
running	0%	0%	0%	0%	90.0%	0%	0%	10.0%	0%
office	0%	0%	0%	0%	0%	90.0%	10.0%	0%	0%
restuarant	0%	0%	0%	10.0%	0%	10.0%	40.0%	40.0%	0%
shopping	10.0%	0%	30.0%	0%	0%	0%	0%	60.0%	0%
music	0%	10.0%	0%	0%	0%	0%	0%	60.0%	30%

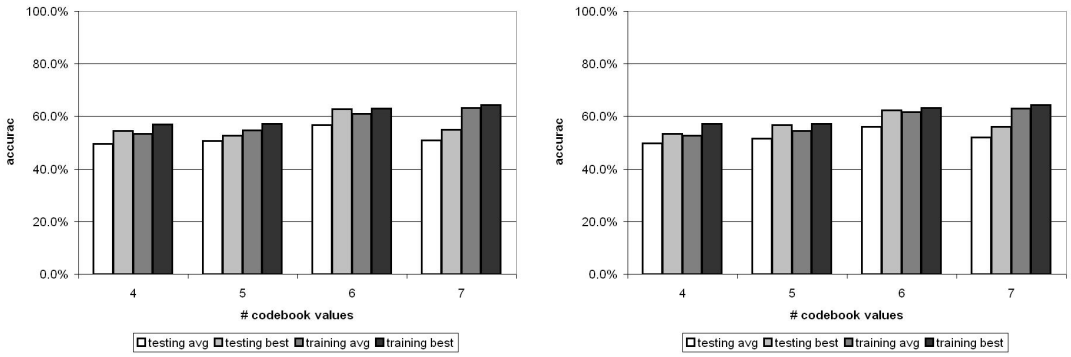
Table 7.9: Confusion matrix for the best-run of the three-class, seven-codebook value HMM using Kates' feature vector

likely due to the fact that both contain a significant amount of babble noise. The HMM also has difficulty properly identifying the music category, most of which is categorized as shopping noise.

7.2.3.2 Inputs selected with SFFS

For the feature sets that include redundant features, the three-feature set could only be tested using two groups. Likely because of the relatively small number of features, the initial clustering of the three and four group models is very difficult, and these models could not be trained. The two group models can be trained. The ranges of accuracies for the HMM using the selected feature vectors are presented in Table 7.5 on page 121.

The accuracies of the testing sets of the four and five feature vector models are both best with six codebook values for all numbers of states. The training sets are all best when using seven codebook values. Please see Figure 7.13. The testing set of the three-feature, two state model is best with seven codebook values. However, because the four and five feature vector testing sets are best with six codebook values, six codebook values are used for comparison.



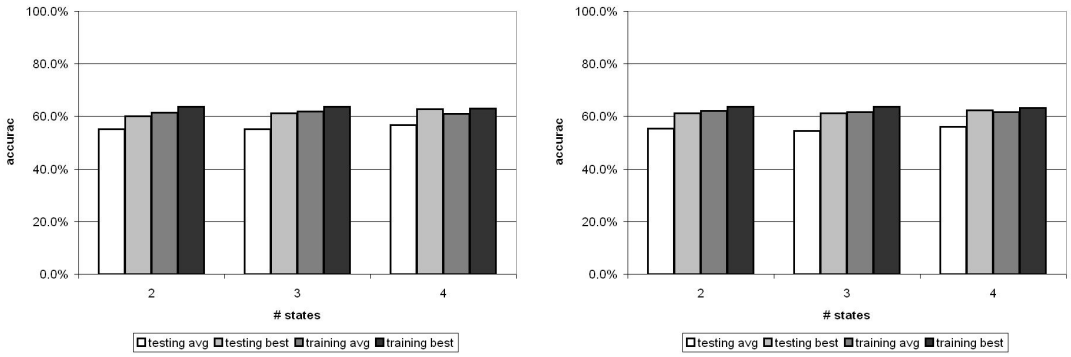
a) four-feature vector b) five-feature vector

Figure 7.13: Accuracy of four and five feature, four state HMM models with different numbers of codebook values using SFFS vectors with redundant features included

The accuracies of the four and five feature vector models are both best with four states. Please see Figure 7.14.

The accuracies of both the four and five feature vectors are very similar. This is likely because the five-feature set includes a repeated feature from the four-feature set. The accuracy of the four feature set is slightly better, most likely because the same amount of information is being represented more compactly, which would make the training somewhat easier. The four-feature set is also smaller.

The confusion matrix for the best run of the four-state six-codebook value HMM using the four-feature vector that includes redundant features is presented in Table 7.10. It is clear from this matrix that the HMM has particular difficulty identifying the restaurant and birds categories. There are also a significant number of non-bird samples that are misclassified as being in the bird class. The bird category includes a large number of samples that are combinations of many sounds, making this category more difficult to classify. This may also be a problem with the training



a) four-feature vector b) five-feature vector

Figure 7.14: Accuracy of four and five feature HMM models with different numbers of states using a codebook size of six using SFFS vectors with redundant features included

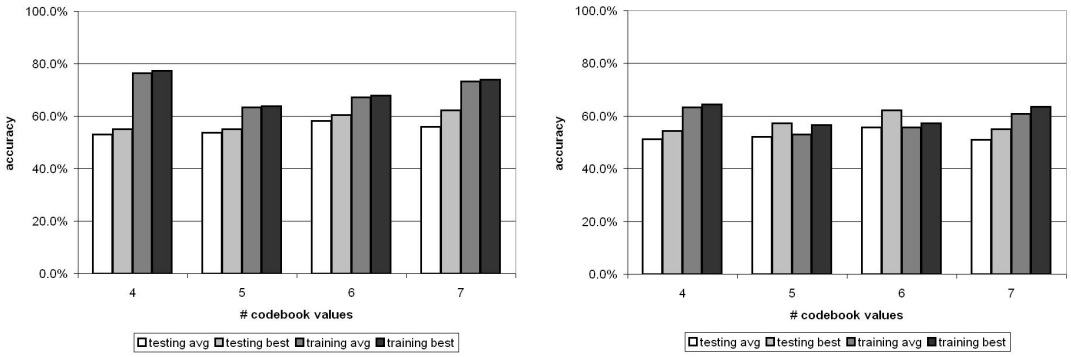
of this model. The fact that so many samples appear in this category can be an indication that the training did not result in a model that was specific enough. The samples in the restaurant category tend to be classified as being traffic or in-car samples.

The model also has some difficulty identifying the office and traffic classes. The office sounds tend to be misclassified as either shopping or restaurant noise, which makes sense since all three include babble noise. The traffic class is difficult to classify likely because the traffic class comes from a large number of different recordings and includes many samples that are combinations of traffic and other sounds.

The two water classes are identified easily, and there are also very few other samples that are misclassified as being in one of these categories. This may be because these categories do not include many combination samples, and also because these samples come from a very small number of recordings, making the test

	selected								
actual	in-car	traffic	birds	washing	running	office	restaurant	shopping	music
in-car	90.0%	0%	10.0%	0%	0%	0%	0%	0%	0
traffic	10.0%	50.0%	20.0%	0%	0%	0%	0%	20.0%	0
birds	10.0%	0%	10.0%	0%	20.0%	0%	0%	20.0%	40.0
washing	0%	0%	0%	100%	0%	0%	0%	0%	0
running	0%	0%	0%	0%	100%	0%	0%	0%	0
office	0%	0%	20.0%	0%	0%	50.0%	20.0%	10.0%	0
restaurant	20.0%	30.0%	10.0%	10.0%	0%	0%	20.0%	10.0%	0
shopping	0%	0%	10.0%	0%	0%	0%	10.0%	80.0%	0
music	0%	0%	10.0%	0%	0%	0%	0%	20.0%	70.0

Table 7.10: Confusion matrix for the best-run of the four state, six codebook value HMM using the four-feature vector selected with SFFS with redundant features included



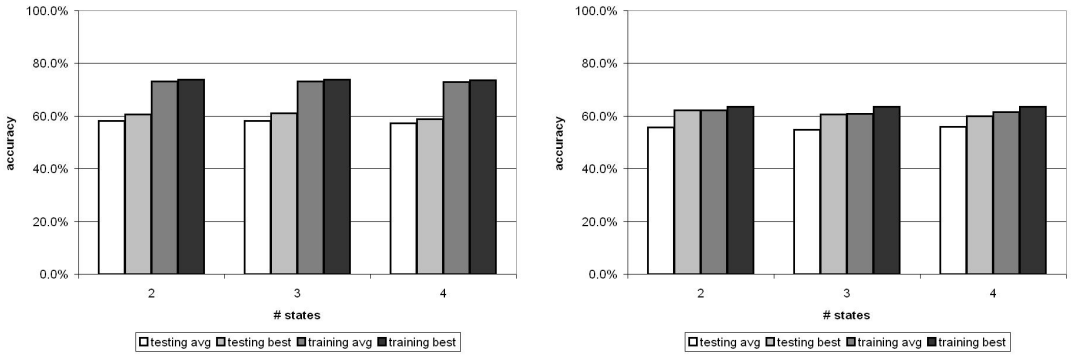
a) four-feature vector b) five-feature vector

Figure 7.15: Accuracy of four and five feature, two state HMM models with different numbers of codebook values using SFFS vectors with redundant features removed

samples more similar to the training samples.

For the feature sets that have the redundant features removed, the three-feature set could again only be tested using two groups. Likely because of the relatively small number of features, the initial clustering of the three and four group models is very difficult, which makes these models more difficult to train.

Both the four and five-features sets have the best average accuracy with six codebook values, regardless of the number of states. Please see Figure 7.15. They are also both best when using two states in the models. Please see Figure 7.16.



a) four-feature vector b) five-feature vector

Figure 7.16: Accuracy of four and five feature HMM models with different numbers of states using a codebook size of six using SFFS vectors with redundant features removed

The accuracies of both the four and five-feature vectors are very similar. This is likely because the five-feature set includes a repeated feature from the four-feature set. The accuracy of the four-feature set is slightly better, most likely because the same amount of information is being represented more compactly, which would make the training somewhat easier. The four-feature set is also smaller, which is good for a hearing aid.

The confusion matrix for the best run of the two-state six-codebook value HMM using the four-feature vector that includes redundant features is presented in Table 7.11. Similar to other HMM confusion matrices, the two water categories and the in-car category are all well classified. The birds category is categorized very badly, as is the traffic category. Both of these contain a large number of samples and samples that are combinations of many different types of noises. There is also confusion between the office, shopping and restaurant classes, all of which contain babble noise.

	selected								
actual	in-car	traffic	birds	washing	running	office	restaurant	shopping	music
in-car	80.0%	0%	0%	0%	0%	10.0%	0%	10.0%	0%
traffic	10.0%	40.0%	0%	0%	0%	30.0%	10.0%	0%	10.0%
birds	0%	0%	20.0%	10.0%	0%	30.0%	0%	30.0%	10.0%
running	0%	0%	0%	90.0%	0%	0%	10.0%	0%	0%
washing	0%	0%	0%	0%	100%	0%	0%	0%	0%
office	0%	0%	0%	0%	0%	60.0%	10.0%	30.0%	0%
shopping	10.0%	10.0%	0%	0%	0%	10.0%	50.0%	20.0%	0%
restaurant	30.0%	0%	0%	0%	0%	10.0%	0%	60.0%	0%
music	0%	20.0%	0%	0%	20.0%	0%	0%	10.0%	50.0%

Table 7.11: Confusion matrix for the best-run of the two state, six codebook value HMM using the four-feature vector selected with SFFS with redundant features removed

The accuracy of these models is actually lower than the accuracy using Kates' feature vector. The accuracy may be able to be improved by using a different distance metric. This is discussed further in section 7.2.4.2.

Even though the results from the HMM are not as good as the MLP they are more consistent. The best-run and average accuracies are much closer than for the MLP or the WMLP. The models are also more general, as the results from the testing and training sets are very similar.

7.2.3.3 Memory and processing requirements

HMMs normally require more memory and are more computationally expensive then an equivalent MLP. One HMMs is used for each environment, hence for an M environment system, M models are required.

For the HMM the B matrix is the most important factor in the memory size. An HMM with N classes, a codebook size of Q and K inputs has a B matrix that is $Q^K N$. Increasing the number of codebook values or the number of inputs therefore increases the size of the HMM dramatically. Additionally, the A matrix requires N^2

memory locations, and the π vector requires N. Hence, the total memory required for an HMM classifier using M HMMs is:

$$Mem_{HMM} = MN(Q^K + N + 1) \tag{7.3}$$

However, the amount of memory can be reduced in a number of ways. The B matrix is clearly the largest part of the model. The B matrix can be significantly reduced by not storing the probability of every possible vector, but by storing the probability of each individual input and calculating the probability of the input vector as a joint probability of each of the input vectors. This type of a model will not capture the relationship between the different inputs, but uses a significantly smaller amount of memory and should be able to capture different relationships between the inputs by using different states. This would also allow the model to be trained with a larger number of codebook values. However, this cannot be implemented directly and would require retraining.

There is, however, an easy way to reduce the amount of memory used without significantly changing the model. There are a large number of entries in the B matrix that have a zero probability from the training set and B is therefore a sparse matrix. These can be eliminated when this model is implemented in a hearing aid and only the non-zero entries can be stored. This will reduce the amount of memory required, but would require a slight change in the way that the B matrix is searched.

Once the system is implemented in a hearing aid, there will be no further training, but the probability of each model must still be calculated. This is done using the forward probabilities. From equation 2.7, the model requires N multiplications

for the initialization, and it also requires that N values be found in the B matrix. From equation 2.8, the recursive calculation of the α values requires N^2T multiplications, additions and searches of the A matrix, and NT multiplications and searches of the B matrix. Finally, the calculation of the probability in equation 2.9 requires N additions. For all of the M models in the system, an HMM classifier therefore requires $MN(NT+2)$ multiplications, $MN(NT+1)$ additions, $MN(T+1)$ searches of the B matrix and MN^2T searches of the A matrix. The actual computational time is therefore dependant on the search algorithm used; however, it is clear that the HMM is significantly more computationally expensive than the MLP.

For the tests using Kates' feature vector, there are four inputs and nine models required. The number of samples in the window is five. The best model uses three states and seven codebook values. The model therefore requires 64935 memory locations and each input requires 459 multiplications, 432 additions, 162 searches of the B matrix and 415 searches of the A matrix.

For the tests using the feature vector selected with SFFS with repeated features included, the best feature vector is the four-feature vector. The best model uses four states and six codebook values. The model therefore requires 46836 memory locations and each input requires 792 multiplications, 765 additions, 216 searches of the B matrix and 720 searches of the A matrix.

When the redundant features are removed, the best feature vector is the four-feature vector. The best model uses two states and six codebook values. The model therefore requires 23382 memory locations and each input requires 216 mul-

tiplications, 198 additions, 108 searches of the B matrix and 180 searches of the A matrix.

7.2.4 Windowed MLP

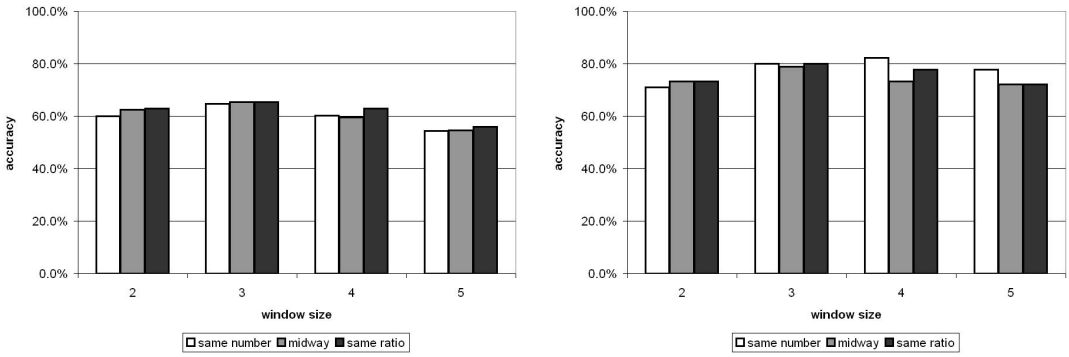
7.2.4.1 Kates' feature vector

Surprisingly, windowing the MLP does not increase the accuracy with the new database. The ranges of accuracies for the windowed MLP using Kates' feature vector are presented in Table 7.5 on page 121.

The number of hidden nodes does not seem to have a large effect on the overall accuracy. Using the same ratio of hidden nodes as the best MLP (largest number) slightly increases the average accuracy on the testing set for all window sizes. Please see Figure 7.17. When using the best-run accuracies, the highest accuracies occur using the same number of hidden nodes as the best MLP (smallest number) in all except the two-sample window model. However, the best-run results are based only on a single case, which is not reliable. Therefore a model that uses the same ratio of hidden nodes is used for comparison, as it is better on average.

The window size also affects the overall accuracy. As seen in Figure 7.18, both the average and best-run accuracies for the testing set increase up to a sample window of size three then decrease slightly. This decrease may be due to the dimensionality problem, where the addition of extra features may actually decrease the accuracy of the system.

Therefore, the system that will be used for comparison is the WMLP that uses a sample window of size three and 30 hidden nodes (same ratio).



a) average accuracy b) best-run accuracy

Figure 7.17: Average and best-run accuracy of WMLP on the testing set with different numbers of hidden nodes

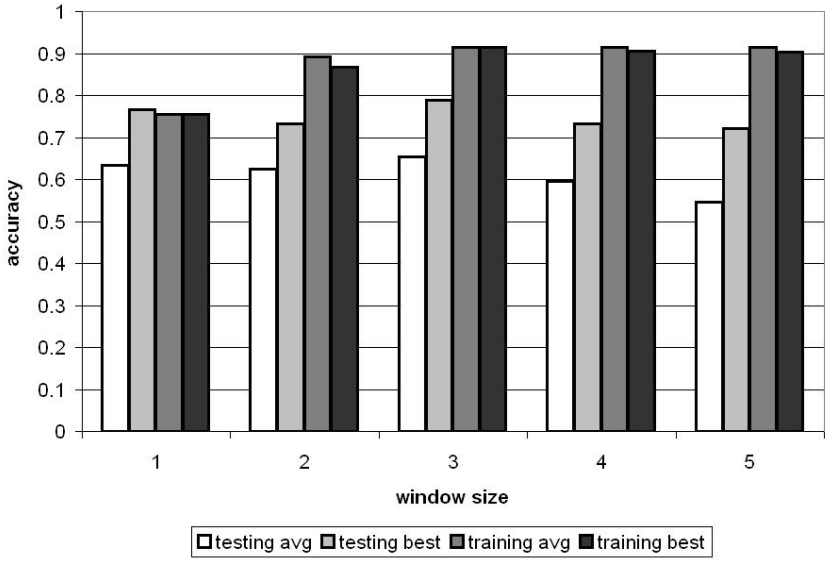


Figure 7.18: Accuracy of the WMLP using the same ration of hidden nodes as the best MLP with different window sizes

	selected									
actual	in-car	traffic	birds	washing	running	office	restaurant	shopping	music	unknown
in-car	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%
traffic	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%
birds	0%	0%	70.0%	30.0%	0%	0%	0%	0%	0%	0%
washing	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%
running	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%
office	30.0%	0%	0%	0%	0%	0%	50.0%	20.0%	0%	0%
restaurant	0%	0%	20.0%	0%	0%	0%	50.0%	30.0%	0%	0%
shopping	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%
music	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%

Table 7.12: Confusion matrix for the best-run of the 30 node WMLP using a window size of three and Kates' feature vector

The confusion matrix for the WMLP with a window size of three and 30 hidden nodes using Kates' feature vector is presented in Table 7.12. The in-car, traffic, music and both water classes are very well classified. There is some confusion with the birds class, which is likely due to the fact that the samples are so varied and contain other sounds in combination. The office noise sound is very badly classified. None of the samples are correctly classified, with most being identified as being either restaurant or shopping noise. All the shopping noise samples are correctly identified, but the restaurant noises are all categorized poorly. The initial tests indicate that this feature vector has some difficulty identifying babble noises. This is likely the cause for these misclassifications in this model.

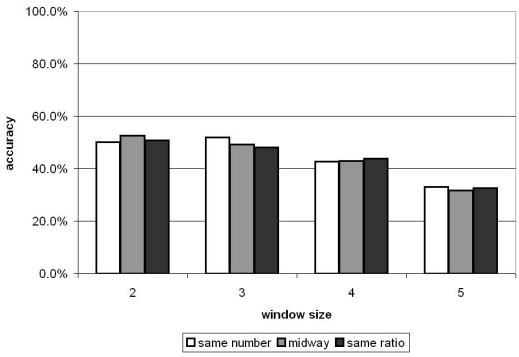
7.2.4.2 Inputs selected with SFFS

The ranges of accuracies for the windowed MLP using the selected feature vectors are presented in Table 7.5 on page 121.

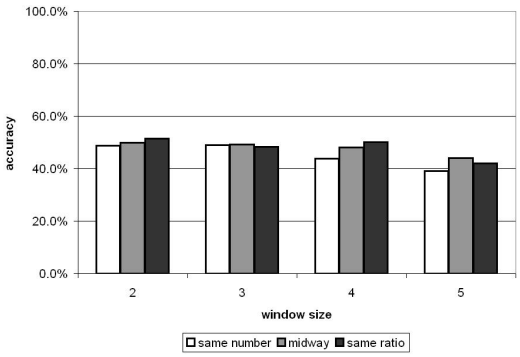
Figure 7.19 shows the average accuracy of the testing set for the three, four and five-feature sets using different numbers of hidden nodes. For the three-feature set, it appears that using the same number of nodes as the best MLP gives the

best results. For both the four and five feature sets, using either the same ratio of hidden nodes, or the midway point gives better results. There does not appear to be a clear pattern indicating which one is better. The midway point is selected for comparison because it allows more flexibility. Particularly as the number of input nodes and window size increases, using the same ratio of hidden nodes as the non-windowed MLP becomes somewhat unreasonable. Using the midway point reduces the number of nodes in the model, but still allows the hidden layer to grow with the input layer. Because only three configurations are tested, it is likely that a different number of hidden nodes will be able to produce better results. However, the results from the models are all relatively similar so it is unlikely that testing many different configurations would result in a large increase in accuracy.

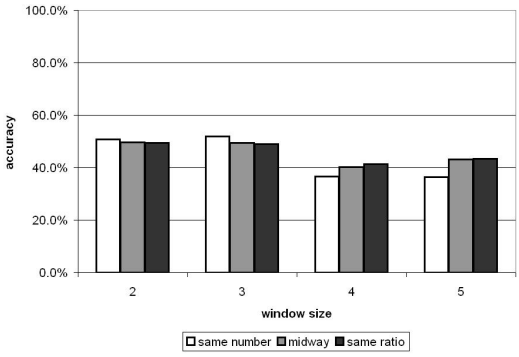
Similar to the MLP, the WMLP trained on the three-feature set gives the best accuracy. Please see Figure 7.19. However, windowing the input does not appear to increase the accuracy of the model. The non-windowed MLP gives the best results and the accuracy decreases as the window size increases. Please see Figure 7.20. This is in direct contradiction to the results achieved in the initial tests. This also contradicts the results in the tests with Kates' feature vector, which showed an increase in accuracy for the two and three window tests. It is possible that with this data set, there actually is very little temporal progression in the samples. Unlike speech, background noise is fairly random in that certain sounds do not necessarily follow others. Because Euclidean distance is not able to track the within-class scatter, the autocorrelation features actually have a fairly large spread within each class. With Kates' feature vector the classes are closer together, but also have less



a) three-feature vector



b) four-feature vector



c) five-feature vector

Figure 7.19: Average accuracy of WMLP with three, four and five SFFS feature sets with redundant features included with different numbers of hidden nodes

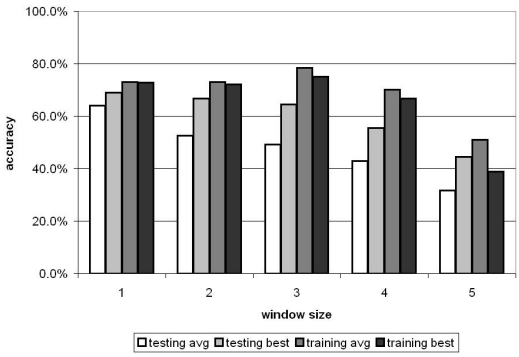
spread. Since the spread is low, adding additional features in the form of a window simply provides more information to the classifier. When samples are further apart, if they do not follow a temporal progression, then adding additional information in the form of a window may actually decrease the chances that the input vector will match. The training will be more difficult, making a full match harder.

Additionally, although the SFFS is run on each different window size, the algorithm is run on the individual pieces of data, not on the window as a whole. Running the SFFS on the entire window may improve the data set for the windowed classifiers.

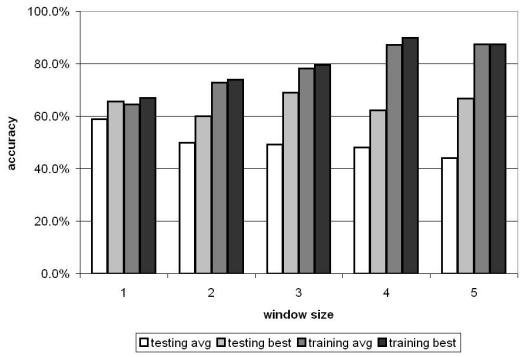
For the feature vectors where the redundant features are manually removed, the models are again tested with same numbers of hidden nodes as the best MLP. The average testing set accuracies with different numbers of hidden nodes are presented in Figure 7.21. Again, there is not a clear trend with respect to the number of hidden nodes. The midpoint value is selected for the reasons described for the previous models.

The window sizes for the different feature vectors are presented in Figure 7.22. These are the values using the midway point of the hidden node values. From this figure it can be seen that the three-feature vector and the window size of one (non-windowed) again produce the best results for this set of tests. It is likely that this distance measure is simple unsuitable for this particular model. Using a distance metric that is more in line with the way the MLP functions would likely produce better results.

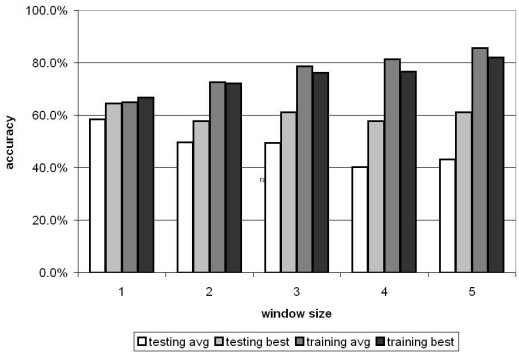
The accuracy using these features sets is quite low. In fact, the accuracy is



a) three-feature vector

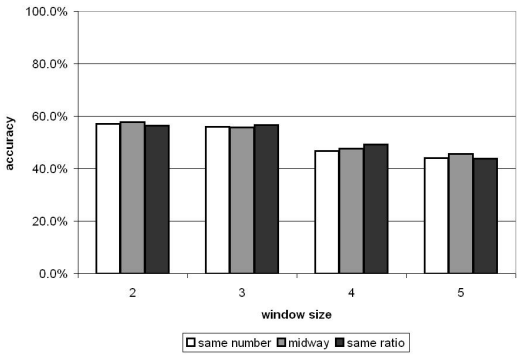


b) four-feature vector

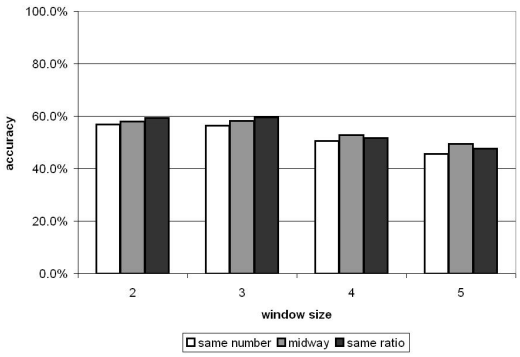


c) five-feature vector

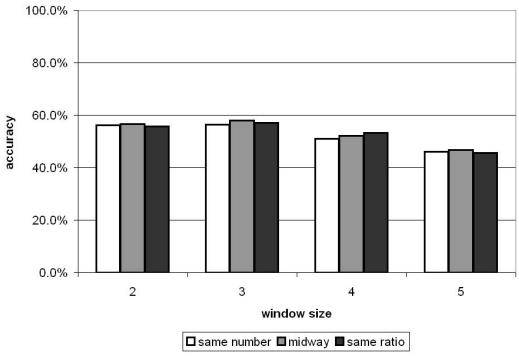
Figure 7.20: Average accuracy of WMLP with three, four and five SFFS feature sets with redundant features included with different window sizes



a) three-feature vector

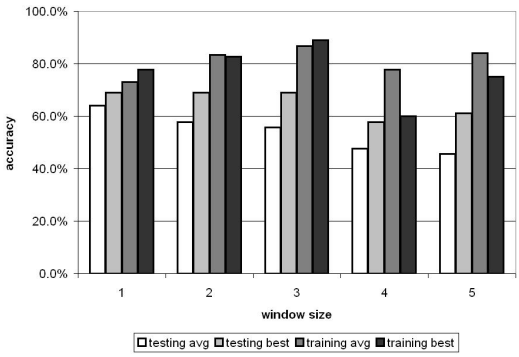


b) four-feature vector

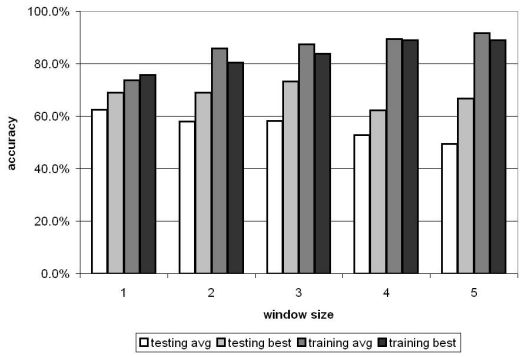


c) five-feature vector

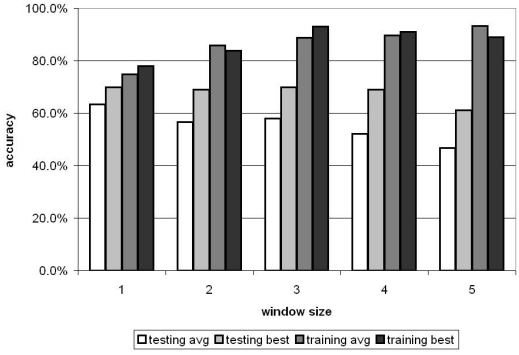
Figure 7.21: Average accuracy of WMLP with three, four and five SFFS feature sets with redundant features removed with different numbers of hidden nodes



a) three-feature vector



b) four-feature vector



c) five-feature vector

Figure 7.22: Average accuracy of WMLP with three, four and five SFFS feature sets with redundant features removed with different window sizes

actually lower than when using Kates' feature vector, which is not selected using any type of formal feature selection algorithm. The distance metric used clearly does not capture all the necessary aspects to improve the feature set. In the first set of tests, the features selected clearly contain redundant information. Adding additional features therefore adds very little extra information, but makes the classifiers more complex and more difficult to train.

7.2.4.3 Memory and processing requirements

The windowed MLP is more computationally expensive and requires more memory than the non-windowed MLP. For a windowed MLP with W sets of inputs, there are WN_i inputs and WN_o hidden nodes. From equations 7.1 and 7.2, both the memory and processing requirements are a function of N_iN_h. Therefore, the amount of memory and processing required by the windowed MLP is on the order of W^2 times the amount of memory required for a non-windowed MLP.

For the tests using Kates' feature vector, there are four inputs that are windowed and nine outputs. The best model uses a window size of three, which makes the size of the input layer 12 nodes. The best models use a hidden layer that is midway between having the same number and same ration of hidden nodes as the un-windowed MLP. This gives 20 hidden nodes for this mode. This model therefore requires 449 memory locations and 420 multiplications and additions to process an input.

For the SFFS tests with redundant features not removed, the best feature vector uses three features, and the best window size is two. This gives an input layer with six nodes. The best model uses 17 hidden nodes. The model therefore requires 281

memory locations and 255 multiplications and additions to process an input.

When the redundant features are removed, the three-feature vector is again the best feature vector, and the best model also has a window size of two and uses 17 hidden nodes. It therefore also requires 281 memory locations and 255 multiplications and additions to process an input.

7.3 Comparison of Classifiers

The final accuracies of the tested classifiers and their implementation details are described in Table 7.13.

7.3.1 KNN

The size of the KNN is the second largest of the models. It is also highly dependent on the training size. If more training vectors are added to increase the coverage of the set, both the size and processing required will also increase.

Surprisingly, the KNN actually has the highest accuracy of the tested classifiers, using the higher order feature vector selected using SFFS with the redundant features removed. The nine-feature vector has an average accuracy of 77.6% and a best-run accuracy of 84.4% using the 5-NN model. This is surprising considering that the KNN is simplest of the tested classifiers. The accuracy of the model using the feature vector from Kates was much lower, and the KNN did not have the highest accuracy when it was tested with this feature vector. This large increase in accuracy was not seen in the other classifiers, which all experienced a decrease in accuracy. However, despite having good accuracy, the large amount of memory

Table 7.13: Comparison of different models and feature vectors

model	features	parameters	memory	processing	avg accuracy	best accuracy
KNN	Kates	$N = 4$ $V = 360$	1440	1440 mult 1440 add 1440 sub 360 sqrt	57.1%	62.2%
KNN	SFFS rep	$N = 3$ $V = 360$	1080	1080 mult 1080 add 1080 sub 360 sqrt	71.1%	75.5%
KNN	SFFS no rep	$N = 5$ $V = 360$	1800	1800 mult 1800 add 1800 sub 360 sqrt	72.4%	76.7%
KNN	SFFS extend	$N = 9$ $V = 360$	3240	3240 mult 3240 add 3240 sub 360 sqrt	77.6%	84.4%
MLP	Kates	$N_i = 4$ $N_h = 10$ $N_o = 9$	149	130 mult 130 add	63.3%	76.7%
MLP	SFFS rep	$N_i = 4$ $N_h = 11$ $N_o = 9$	163	143 mult 143 add	58.9%	65.6%
MLP	SFFS no rep	$N_i = 4$ $N_h = 11$ $N_o = 9$	163	143 mult 143 add	62.6%	68.9%
MLP	SFFS extend	$N_i = 9$ $N_h = 11$ $N_o = 9$	218	198 mult 198 add	68.2%	74.4%
HMM	Kates	$M = 4$ $N = 3$ $Q = 7$ $K = 4$ $T = 5$	64935	459 mult 432 add 162 B 405 A	59.7%	58.3%
HMM	SFFS rep	$M = 9$ $N = 2$ $Q = 6$ $K = 4$ $T = 5$	23382	216 mult 198 add 108 B 180 A	55.3%	61.1%
HMM	SFFS no rep	$M = 9$ $N = 4$ $Q = 6$ $K = 4$ $T = 5$	46836	792 mult 756 add 216 B 720 A	61.0%	63.1%
WMLP	Kates	$N_i = 12$ $N_h = 30$ $N_o = 9$	669	630 mult 630 add	65.5%	80.0%
WMLP	SFFS rep	$N_i = 6$ $N_h = 17$ $N_o = 9$	281	255 mult 255 add	52.6%	66.7%
WMLP	SFFS no rep	$N_i = 6$ $N_h = 17$ $N_o = 9$	281	255 mult 255 add	57.7%	68.9%

and processing required make the KNN algorithm impractical for use in a hearing aid.

The discrepancy between the KNN improvement and the results from the other classifiers is likely due to the distance metric used in the SFFS algorithm. The increase for the KNN indicates that the SFFS is capable of picking a feature vector set that can give good results. However, the distance metric is very simple and is clearly not suitable. The KNN, however, matches well with this distance metric. The KNN directly uses Euclidean distance in its calculation of the class. The other classifiers do not use this measure so directly; hence these classifiers do not experience the same increase in accuracy. In [37] Bucler et. al. note that different classifiers perform best with different sets of input features. This is likely also applicable to the feature selection techniques, where different distance metrics may improve the performance of different classifiers. A better distance metric would be able to account for a number of different potential problems, including within-class scatter and redundancy of the features vectors, which was clearly seen in this feature set. A more complex distance metric would ideally also be able to account for issues other than accuracy, particularly that some features are more attractive than others in that they require less processing to extract. Finding a good balance between these desired traits, however, is something that will likely prove to be quite challenging.

7.3.2 HMM

The HMM, unfortunately, did not perform very well in any of the tests. There may be a number of reasons for this poor performance. Firstly, the use of multiple inputs means that using a large number of possible quantization values results in a very large model. The way the HMM is updated requires that each individual input vector have its own probability. This problem could likely be avoided by changing the update procedure to track each feature's probability independently, and combining them to form an overall probability. The drawback of this approach would be that the HMM would not be able to track the correlation between input vectors. However, with a sufficient number of states in the model, this technique could work well and would allow a significantly larger number of possible quantization values.

The HMM also requires the largest amount of memory. However, it is possible to reduce this size by changing the way the B matrix is stored. Instead of storing the individual probability for each input vector, the probability would be calculated as the combined probability of each individual input. The size could also be reduced by not storing the vectors that have zero probability.

Additionally, the tests with the WMLP show that using the larger data set, the accuracy of the model decreases for a larger window. The HMM was tested with a window of five. This may also be one of the reasons for the lower accuracy.

7.3.3 MLP

The MLP is actually able to generate reasonably good results using Kates' feature vector. The average accuracy is 63.4% and the best-run result has an accuracy of

76.7%. Considering the number of classes and the fact that the data set is quite diverse, this is a reasonable accuracy for this system. Using the features vectors selected using SFFS actually produced worse results. This is likely due to the fact that the distance metric used was quite simple. A different distance metric might produce better results.

7.3.4 WMLP

Using a windowed MLP increases the accuracy for the models using Kates' feature vector when the window size is two or three. The window size of three gives an average accuracy of 65.5% and a best-run accuracy of 71.1%. The accuracy decreases for window sizes of four and five. However, windowing the input decreases the accuracy in all cases for the feature vectors selected with SFFS.

There is an additional advantage to the MLP in that it is able to indicate when a sample is unknown. Many of the misclassified samples are actually classified as unknown. It is possible that these are actually being grouped into more than one category. This at least gives an indication that the results should not be trusted. It may also be possible to mimic this behaviour with the HMM by classifying a sample as unknown if two or more environment models give results which are within a certain threshold, or if the highest probability environment model has a probability that falls below a certain threshold. These values would likely have to be determined through trial and error.

7.4 Summary

When using the SFFS feature vectors, as the window size on the MLP increases, the accuracy of the model decreases. This is not the case when using Kates' vector. There may be a number of reasons for this difference. The initial tests show that the Euclidean distance between the class means decreases as the window size is increased. However, this trend is also seen in Kates' vector. In all cases, Kates' vector has a smaller Euclidean distance between classes than any of the SFFS vectors and still performed better.

The issue of scatter may also be one of the reasons for the decline in accuracy for the windowed models when using SFFS. Background environments do not necessarily contain a temporal component. Unlike speech, which does have a natural temporal progression, background noise can be somewhat random, and have sounds that are specific to the model but do not necessarily follow or precede other sounds. There is most likely at least some temporal progression in most of the environments, but the windows operate on the order of 1 s, which may be too long. The autocorrelation already contains a temporal component and also contains a fairly large amount of scatter.

Since Kates' feature vector has a relatively small amount of within-class scatter, then each of the inputs should be relatively similar. The model would not have much difficulty tracking the relationship between the different windows because they are very similar and do not change much across the time frames. The training would be relatively simple, and windowing the data would make it more likely that at least one of the inputs would be close to the trained model. If the inputs contain a large

amount of scatter, however, then the training would be more difficult. Windowing the data would not be helpful as none of the inputs would necessarily be close to the trained model.

Overall, the classes that are more scattered in the SOM and K-means clustering tended to be the samples that were most often misclassified. This includes the babble and the traffic classes. There was some confusion between the classes that contained the most babble noise. These not only contain similar sounds, but are also clustered within a relatively small region on the SOM and tend to be slightly scattered. Conversely, the most tightly clustered sounds tended to be the most easily classified.

Changing the implementation of the feature selection algorithm may help the models to better classify some of these more difficult classes. The current algorithm looks at the classes together. If one is already far away from the other classes, the feature selection algorithm makes no differentiation between adding a feature that increases the distance of that same class, or adding a feature that increases the distance to another class. It may be better to apply the feature selection individually to each class and produce a feature vector that consists of features that are individually good at separating one or more classes. This would be more similar to the way the HMM functions, where each class is evaluated separately in its own model, as opposed to MLP which evaluates each of the classes together.

Given the extent of the scattering in the SOM and K-means clustering, it is also possible that some of the samples actually have autocorrelation matrices that are closer to different classes. In this case, they would be better removed using the

weights for a different class, but would still appear as being misclassified. These types of samples would be very difficult to evaluate in this classification system, where samples are still categorized using logical classes. Perhaps a better solution would be simply to identify which of the pre-set weights the samples is closest to, by using the weights themselves or something similar as an input to the classifier. The evaluation of the system could be done by connecting it to an actual noise reduction model and evaluating the SNR and the intelligibility. This is clearly a more complex system, but may yield good results with real-world sounds.

Chapter 8

Conclusions

This thesis examined environment classification for hearing aids. The work focuses on identifying a good classifier for this application, identifying a reasonable set of classes, and testing different features using a formal feature selection algorithm.

Initial tests are performed using KNN, MLP and HMM classifiers, as well as an MLP with windowed input. For this initial phase, the classes and features are assumed. The results indicate that the windowed MLP is a strong candidate for this application. The data set, however, is somewhat limited and only four classes were used.

Following the initial tests, SOM and K-means clustering are used to identify candidate classes. Samples are tagged as containing one or more of 28 different possible sounds. The final set of classes selected are in-car, traffic, birds, water washing, water running, office noise, restaurant noise, shopping and music.

Feature selection is performed using SFFS and Euclidean distance and Fisher's interclass separability criterion. Fisher's interclass separability criterion does not

work for this application as it requires the inversion of a matrix and many combinations are not invertible. The Euclidean distance is also not a distance metric for this application, as it tends to select redundant classes. It also does not consider the within-class scatter. A better distance metric would also consider the processing power and memory required to calculate the features.

The classifiers are then tested using the selected classes and features selected using Euclidean distance with and without the redundant features removed. These results are compared against classifiers that use a simpler feature vector from Kates [10]. The accuracy of the final tests are much lower than the initial tests in all cases, likely due to the addition of more classes and a more varied sound database.

The selected features do not work as well as Kates' feature vector in most cases. A different distance metric might produce better results. More work is required in this area.

With the feature vector from Kates, the windowed MLP gives the best results. The HMM does not perform as well as either the windowed on the non-windowed MLP. The WMLP with a window size of three is able to give an average accuracy of 65.5% and a best-run accuracy of 80.0% on the testing set.

When using the SFFS feature vectors, the MLP performs better than the HMM and the windowed MLP. However, the KNN actually gives the best results. This is likely because the features are assessed using Euclidean distance, and this distance metric is also directly used by the KNN. Of the trained models, the non-windowed MLP has the best accuracy. It is able to achieve an average accuracy of 62.6% and a best-run accuracy of 68.9% on the testing set, using the three-feature vector with

the redundant features manually removed. Using a larger feature vector increases the average accuracy of the MLP to 68.2% and gives a best-run accuracy of 74.4% using a nine-feature vector. This is actually a slightly higher average accuracy than the WMLP achieves using Kates' feature vector.

There are three major contributions from this thesis:

1. The analysis of sound samples using clustering and the selection of a set of output classes suitable for a hearing aid user.

2. The use of a formal feature selection technique to select features for the environment classifier

3. The testing of a multi-layer perceptron using windowed input.

Future work should focus on finding a more appropriate distance metric for feature selection. The distance metric should account for both scatter and redundancy, neither of which is accounted for with Euclidean distance. The results from the KNN indicates that using a distance metric that is closely related to the classifier could also improve the performance. Another possibility is to change the feature selection algorithm itself and allow the feature selection to pick a feature based on its ability to differentiate an individual class that is not well represented.

Other future work should focus on the subjective results of the classification, by combining the system with a noise reduction algorithm. It is possible that some of the misclassified samples would actually be well removed with the weights selected by the classifier. Using inputs that are related to the weights would allow an input to be classified as being a good match to a particular set of weights, rather than as

being part of a specific class.

Overall, this thesis shows that the MLP can work well for environment classification. The accuracy is relatively high and it uses a smaller amount of memory and processing power than other models. If the feature vector has a low scatter, windowing the input can also be helpful. Clustering gives a reasonable set of candidate classes that can likely be used for future work in this area. However, the distance metric used for feature selection is clearly unsuitable for this application and future work will need to focus on finding a new way to perform feature selection.

Bibliography

[1] Statistics Canada, "Participation and Activity Limitation Survey", 2002, URL: http://www.statcan.ca/Daily/English/021203/d021203a.htm, accessed:June 20, 2007.

[2] Statistics Canada, "Statistical Report on the Health of Canadians", 1999, URL: http://www.statcan.ca/english/freepub/82-570-XIE/82-570-XIE1997001.pdf, accessed:June 20, 2007.

[3] Health Canada, "Hearing Loss Info-sheet for Seniors", 2006, URL: http://www.phac-aspc.gc.ca/seniors-aines/pubs/info_sheets/hearing_loss/index.htm, accessed:June 20, 2007.

[4] Health Canada, "It's Your Health: Hearing Loss and Leisure Noise", 2005, URL: http://www.hc-sc.gc.ca/iyh-vsv/environ/leisure-loisirs_e.html, accessed:June 20, 2007.

[5] American Speech-Language-Hearing Association, "Type, Degree and Configuration of Hearing Loss", 2005, URL:

http://www.asha.org/public/hearing/disorders/types.htm, accessed: August, 2006.

[6] Navy Environmental Health Center, "Ear disorders and hearing loss", 2004, URL: http://www-nehc.med.navy.mil/downloads/occmed/aud_hc_course/EAR%20DISORDERS%20&%20HEARING%20LOSS.ppt, accessed: August, 2006.

[7] Reinier Plomp, "Auditory handicap of hearing impairment and the limited benefit of hearing aids", *Journal of the Acoustical Society of America*, vol. 63, n. 2, pp. 533 – 549, 1978.

[8] Nathaniel A. Whitmal and Janet C. Rutledge, "Noise reduction algorithms for digital hearing aids", *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, vol. 16, n. 2, pp. 1294 – 1295, 1994.

[9] D. Graupe, S.P. Basseas and J.K. Grosspietsch, "Self adaptive filtering of noise from speech: a hearing aid application", *1988 IEEE International Symposium on Circuits and Systems. Proceedings*, pp. 2619 – 2624, 1988.

[10] J.M. Kates, "Classification of background noises for hearing-aid applications", *Journal of the Acoustical Society of America*, vol. 97, n. 1, pp. 461 – 470, 1995.

[11] A.M. Engebretson, "Benefits of digital hearing aids", *IEEE Engineering in Medicine and Biology Magazine*, vol. 13, n. 2, pp. 238 – 248, 1994.

[12] Brent W. Edwards, "Signal processing techniques for a DSP hearing aid", *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 6, pp. 586 – 589, 1998.

[13] Nathaniel A. Whitmal, Janet C. Rutledge and Jonathan Cohen, "Reducing correlated noise in digital hearing aids", *IEEE Engineering in Medicine and Biology*, vol. 15, n. 5, pp. 88 – 96, 1996.

[14] Rob A.J. De Vries and Bert De Vries, "Towards SNR-loss restoration in digital hearing aids", *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 4, pp. 4004–4007, 2002.

[15] Simon S. Haykin, *Neural networks: a comprehensive foundation*, Macmillan, New York, 1999.

[16] N.A. Whitmal and J.C. Rutledge, "Noise reduction in hearing aids: a case for wavelet-based methods", *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 3, pp. 1130 – 1135, 1998.

[17] Simon Haykin and Barry Van Veen, *Signals and Systems*, John Wiley and Sons, Inc., 1999.

[18] J.M. Kates and M.R. Weiss, "A comparison of hearing-aid array-processing techniques", *Journal of the Acoustical Society of America*, vol. 99, n. 5, pp. 3138 – 48, 1996.

[19] N. Magotra, P. Kasthuri, Y. Yang, R. Whitman and F. Livingston, "Multichannel adaptive noise reduction in digital hearing aids", *ISCAS '98. Proceedings*

of the 1998 IEEE International Symposium on Circuits and Systems, vol. 6, pp. 582 – 585, 1998.

[20] Neeraj Magotra and Sudheer Sirivara, "Real-time digital speech processing strategies for the hearing impaired", *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2, pp. 1211 – 1214, 1997.

[21] Barry Van Veen and Kevin M. Buckley, *The Digital Signal Processing Handbook*, ch. 61 - Beamformning Techniques for Spatial Filtering, pp. 61–1 to 61–23, CRC Press, 1998.

[22] Volkmar Hamacher, "Comparison of advanced monaural and binaural noise reduction algorithms for hearing AIDS", *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 4, pp. 4008 – 4011, 2002.

[23] J.M. Kates, "Superdirective arrays for hearing aids", *Journal of the Acoustical Society of America*, vol. 94, n. 4, pp. 1930 – 1933, 1993.

[24] G.H. Saunders and J.M. Kates, "Speech intelligibility enhancement using hearing-aid array processing", *Journal of the Acoustical Society of America*, vol. 102, n. 3, pp. 1827 – 1837, 1997.

[25] T. Wittkop and V. Hohmann, "Strategy-selective noise reduction for binaural digital hearing aids", *Speech Communication*, vol. 39, n. 1-2, pp. 111 – 138, 2003.

[26] L.R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", *Proceedings of the IEEE*, vol. 77, n. 2, pp. 257 – 286, 1989.

[27] David Calvert, "CIS6420 Lecture Notes: Self Organizing Map".

[28] D. Zongker and A. Jain, "Algorithms for feature selection: An evaluation", *Proceedings of the 13th International Conference on Pattern Recognition*, vol. 2, pp. 18 – 22, 1996.

[29] Qi Huang, "Assisted analysis of ultrasound tendon images based on neural network texture segmentation", Master's thesis, University of Guelph, 2004.

[30] P. Pudil, J. Novovicova and J. Kittler, "Floating search methods in feature selection", *Pattern Recognition Letters*, vol. 15, n. 11, pp. 1119 – 1125, 1994.

[31] J.M. Kates, "Constrained adaptation for feedback cancellation in hearing aids", *Journal of the Acoustical Society of America*, vol. 106, n. 2, pp. 1010 – 1019, 1999.

[32] Claus Elberling, "Loudness scaling revisited", *Journal of the American Academy of Audiology*, vol. 10, n. 5, pp. 248–260, 1999.

[33] Peter Nordqvist and Arne Leijon, "An efficient robust sound classification algorithm for hearing aids", *Journal of the Acoustical Society of America*, vol. 115, n. 6, pp. 3033 – 3041, 2004.

[34] A et. al. Ringdahl, "Does the heraing aid user take advantage of different settings in multiprogrammable hearing aids in acoustically various listening

environments", *Recent Developments in Hearing Instrument Technology: Proceedings of the 15th Danavox Symposium*, pp. 453–464, 1993.

[35] Vesa Peltonen, Juha Tuomi, Anssi Klapuri, Jyri Huopaniemi and Timo Sorsa, "Computational auditory scene recognition", *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2, pp. 1941–1944, 2002.

[36] Andrew W. Moore, "Executive summary: Bayes nets", URL: http://www.cs.cmu.edu/ awm/tutorials, accessed: January 2006.

[37] M. Buchler, S. Allegro, S. Launer and N. Dillier, "Sound classification in hearing aids inspired by auditory scene analysis", *EURASIP Journal on Applied Signal Processing*, vol. 2005, n. 18, pp. 2991 – 3002, 2005.

[38] Andrew W. Moore, "Clustering with Gaussian Mixtures", 2001, URL: http://www.autonlab.org/tutorials/gmm.html, accessed: January 2006.

[39] R.G. Malkin and A. Waibel, "Classifying user environment for mobile applications using linear autoencoding of ambient audio", *2005 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. Vol. 5, pp. 509 – 512, 2005.

[40] S. Ravindran and D.V. Anderson, "Audio classification and scene recognition and for hearing aids", *IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 860 – 863, 2005.

[41] Peter Nordqvist and Arne Leijon, "Automatic classification of the telephone listening environment in a hearing aid", *Royal Institute of Technology De-*

partment of Speech, Music and Hearing Quarterly Progress and Status Report, vol. 43, pp. 45–49, 2002.

[42] H. Sheikhzadeh, H. Sameti, L. Deng and R.L. Brennan, "Comparative performance of spectral subtraction and HMM-based speech enhancement strategies with application to hearing aid design", *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 1, pp. 13 – 16, 1994.

[43] Xi Shao, Changsheng Xu and M.S. Kankanhalli, "Unsupervised classification of music genre using hidden Markov model", *2004 IEEE International Conference on Multimedia and Expo (ICME)*, vol. 3, pp. 2023 – 2026, 2004.

[44] A. Temko and C. Nadeu, "Classification of meeting-room acoustic events with support vector machines and variable-feature-set clustering", *2005 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 505 – 508, 2005.

[45] C.K. Dagli, "Adaptive music classification using neural network architectures", *Smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining, and Complex Systems. Vol.10. Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE 2000)*, pp. 917 – 921, 2000.

[46] Chris Felton and Chia-Jiu Wang, "Intelligent music classification with support vector machines", *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 13, pp. 963 – 968, 2003.

[47] M. Kashif Saeed Khan, Wasfi G. Al-Khatib and Muhammad Moinuddin, "Automatic classification of speech and music using neural networks", *MMDB 2004: Proceedings of the Second ACM International Workshop on Multimedia Databases*, pp. 94 – 99, 2004.

[48] Alessandro Bugatti, Alessandra Flammini and Pierangelo Migliorati, "Audio classification in speech and music: A comparison between a statistical and a neural approach", *Applied Signal Processing*, vol. 2002, n. 4, pp. 372 – 378, 2002.

[49] Beth Logan, "Mel frequency cepstral coefficients for music modelling", 2000, URL: http://ciir.cs.umass.edu/music2000/papers/logan_abs.pdf, accessed: April 2006.

[50] F. Feldbusch, "A heuristic for feature selection for the classification with neural nets", *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, vol. 1, pp. 173 – 178, 2001.

[51] Areibi S.M. Freeman C., Dony R., "Audio Environment Classification for Hearing Aids Using Artificial Neural Networks with Windowed Input", *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Image and Signal Processing*, pp. 183 – 188, 2007.

[52] Universitat Pompeu Fabra Institut Universitari de L'audiovisual, "The Freesound Project", URL: http://freesound.iua.upf.edu/, accessed: February, 2006.

[53] Richard P. Lippmann, "An introduction to computing with neural nets", *IEEE ASSP Magazine*, vol. 4, n. 1, pp. 4–22, 1987.

[54] Guang-Bin Huang, Yan-Qiu Chen and H.A. Babri, "Classification ability of single hidden layer feedforward neural networks", *IEEE Transactions on Neural Networks*, vol. 11, n. 3, pp. 799 – 801, 2000.

[55] Lawrence R. Rabiner and Biing-Hwang Juang, "Introduction to hidden Markov models", *IEEE Acoustics, Speech, and Signal Processing Magazine*, vol. 3, n. 1, pp. 4 – 16, 1986.

[56] Robert Dony, "ENGG6560: Advanced Digital Signal Processing Lecture Notes".

[57] Mathworks, "MATLAB help file: lpc (signal processing toolbox)".

[58] Richard Mammone and Xiaoyu Zhang, *The Digital Signal Processing Handbook*, ch. 27: Robust speech processing as an inverse problem, CRC press and IEEE press, 1998.

[59] Gin-Der Wu and Chin-Teng Lin, "Word boundary detection with mel-scale frequency bank in noisy environment", *IEEE Transactions on Speech and Audio Processing*, vol. 8, n. 5, pp. 541 – 54, 2000.

[60] E. Pollastri and G. Simoncelli, "Classification of melodies by composer with hidden Markov models", *Proceedings First International Conference on WEB Delivering of Music. WEDELMUSIC 2001*, pp. 88 – 95, 2001.

[61] Patricio De La Cuadra, "Pitch detection methods review", URL:

http://ccrma-www.stanford.edu/ pdelac/154/m154paper.htm, accessed: March 21, 2007.

[62] Jonathan Stein, *Digital Signal Processing A computer Science Perspective*, ch. 9: Correlation, John Wiley and Sons, 2000.

Appendix A

Feature Selection

A.1 In-car

The in-car samples form a very cohesive group in all the different SOMs and in the K-means clustering. An example SOM can be seen in Figure A.1.

In the K-means clustering, the in-car samples are always in single group for the five-lag autocorrelation tests, for all numbers of groups (five to ten). The 10, 15 and 20 lag tests each have one or two samples in a different group, for all numbers of groups. Overall, the K-means clustering also indicates that this is a very cohesive category.

In all cases, both the in-car/music and the in-car/signal samples are clustered with the in-car noise. This is also true for the SOM tests. Music samples not also tagged as in-car do not form part of the cluster of the in-car samples. Samples tagged as being in-car/music or in-car/car signal noise are therefore categorized with in-car noise.

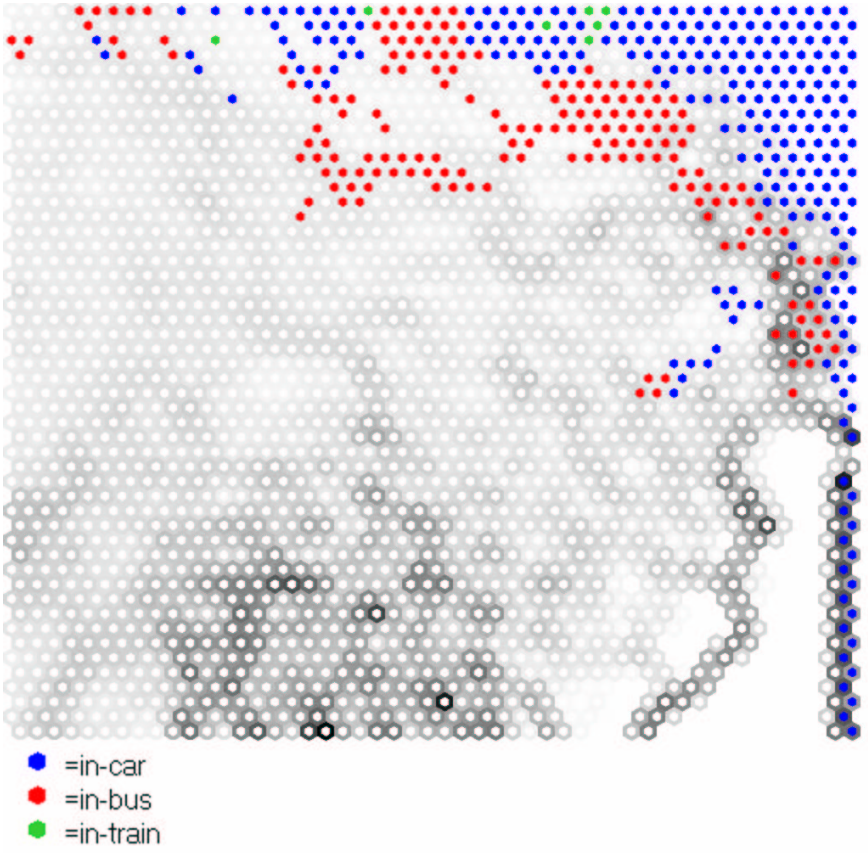


Figure A.1: SOM distribution of in-car, in-bus and in-train samples using a 50x50 map and 20 autocorrelation lags

Surprisingly, the traffic category is not consistently clustered with the in-car noise, either in the SOM or in the K-means clustering. Hence, traffic sounds are not included in this category. This is discussed further in Section A.14.

A.2 In-bus

The SOM clustering indicates that the in-bus category is much more scattered than the in-car. Generally, however, the in-bus category is clustered next to the

in-car category, and the distance between these two categories is quite low, both spatially and based on the distances between the adjacent node values. There are, however, clusters of outlying samples that are not included with the main group in several maps. This is illustrated in Figure A.1. Often these become clustered near the traffic samples, which makes categorizing these samples more difficult. The samples that include footsteps or babble do not create their own clusters, indicating that the in-bus noise is the dominant noise for these samples. Therefore, samples that include babble or footsteps do not need to be separated.

The K-means clustering also shows that the in-bus is a less cohesive category than the in-car category. However, the majority of the in-bus samples are clustered together in all of the K-means tests. The in-bus samples are clustered in at most two categories in all cases, and the separation of the group tends to increase as the number of autocorrelation lags increases. Samples including babble or footsteps are not separated from the regular in-bus samples, which indicates that the in-bus sounds are dominant.

In all cases of the K-means clustering, the cluster containing the majority of the in-bus samples is the same as the cluster containing the majority of the in-car samples. In all cases, if there are in-car samples that are separate from the main group, the second group of in-bus samples is clustered in the same category as these in-car samples. This is likely because the in-bus samples have the same types of sounds as the in-car samples, including muffled motor noise and some reverberation from being in an enclosed space.

The in-bus samples are therefore categorized with the in-car samples, and the

samples including babble or footsteps are not be separated.

A.3 In-train

The SOM clustering indicates that the in-train category is more scattered than the in-car category. Generally, however, a large portion of the in-train category is located directly next to the in-car category. In most cases, the majority of the in-train category is actually closer than the in-bus category. Please see Figure A.1.

The K-means also shows that the in-train is less cohesive than the in-car category. The in-train samples are split between two or more categories in all of the K-means tests. The number of samples not clustered in the main group tends to increase as the number of autocorrelation lags increases.

In all cases of the K-means clustering, the category containing the majority of the in-train samples is the same as the cluster containing the majority of the in-car and in-bus samples. In all cases, the in-train samples that are separate from the main grouping are clustered with the in-car and in-bus samples that are also separate.

The samples containing babble or electronic sounds are not separate from the main group in either the SOM or the K-means clustering. This indicates that the in-train noise is dominant and the samples with babble or other noises should not be separated.

From these tests, it appears that the in-train noises are similar to the in-bus and in-car noises, likely because they all contain muffled motor noise and some reverberation from being in an enclosed area. The in-train noises are therefore

categorized with the in-car noises. Samples with babble or electronic noise are not be separated.

A.4 In-subway

The samples from inside a subway train are actually further away from the in-car samples than the in-train samples in the SOM clustering. The distance between the in-car and the in-subway classes is further than the distance between the in-subway and the traffic in most cases. The in-subway samples also appear to be close to the subway station samples in most cases. However, the in-subway samples tend to be quite spread out, and in some cases appear to be closer to the in-bus/in-train/in-car samples, and sometimes appear to be closer to the traffic samples. Please see Figure A.2.

The K-means clustering also shows the same spread, as the category is often split into two or more clusters, often with a similar number of samples in each category. In some cases the majority of the samples are categorized with the in-car samples, other times they are categorized with the traffic samples.

There is no differentiation of the samples that also include babble or footsteps, indicating that the subway noise is the dominant noise.

In all cases, the cluster with the largest number of in-subway samples also has the largest number of subway station samples. This indicates that the in-subway and subway station samples should be categorized together. This intuitively makes sense, as the subway noise is present in both the in-subway and the subway station samples.

These should not be categorized with the in-car samples, as there is not a strong correlation between these two categories. This may be because there is less reverberation in subways than in cars, and because they use a different type of engine, which would therefore create a different type of engine noise.

These samples will be clustered with the traffic samples, as the in-subway samples tend to be closer to the traffic samples than the in-car samples.

A.5 Car Turn Signal

The car signal noise occurs only in samples that are also tagged as in-car, since this is the only time the signal noise would be audible. The SOM tests show the car signal samples are normally near the edge of the in-car group, but they are not separated by any other group. Please see Figure A.3. In the K-means tests, the car signal samples are always categorized with the in-car samples.

The car signal samples are therefore categorized with the in-car samples. This is logically correct as well, as the time it would take to switch the hearing aid weights is more than one second, and car signals are not often on for much longer than this. It is not desirable to switch the hearing aid program this often, as the user may find it distracting.

A.6 Nature

The nature category is not a cohesive category in either the SOM or the K-means clustering. It is likely that "nature" is too broad a category to properly account

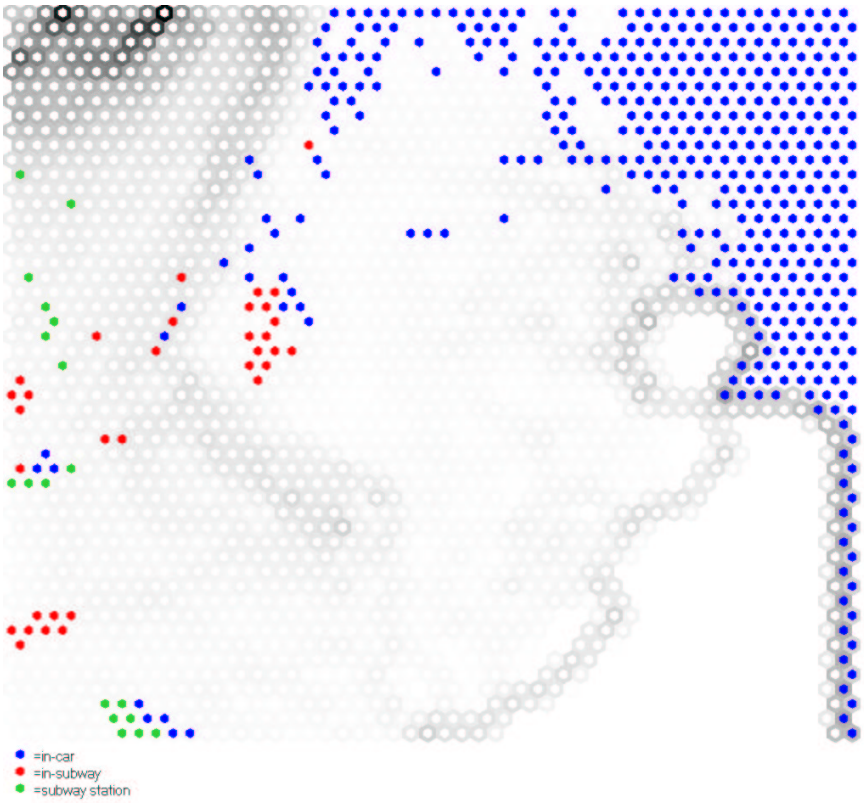


Figure A.2: SOM distribution of in-car, in-subway and subway samples using a 50x50 map and five autocorrelation lags

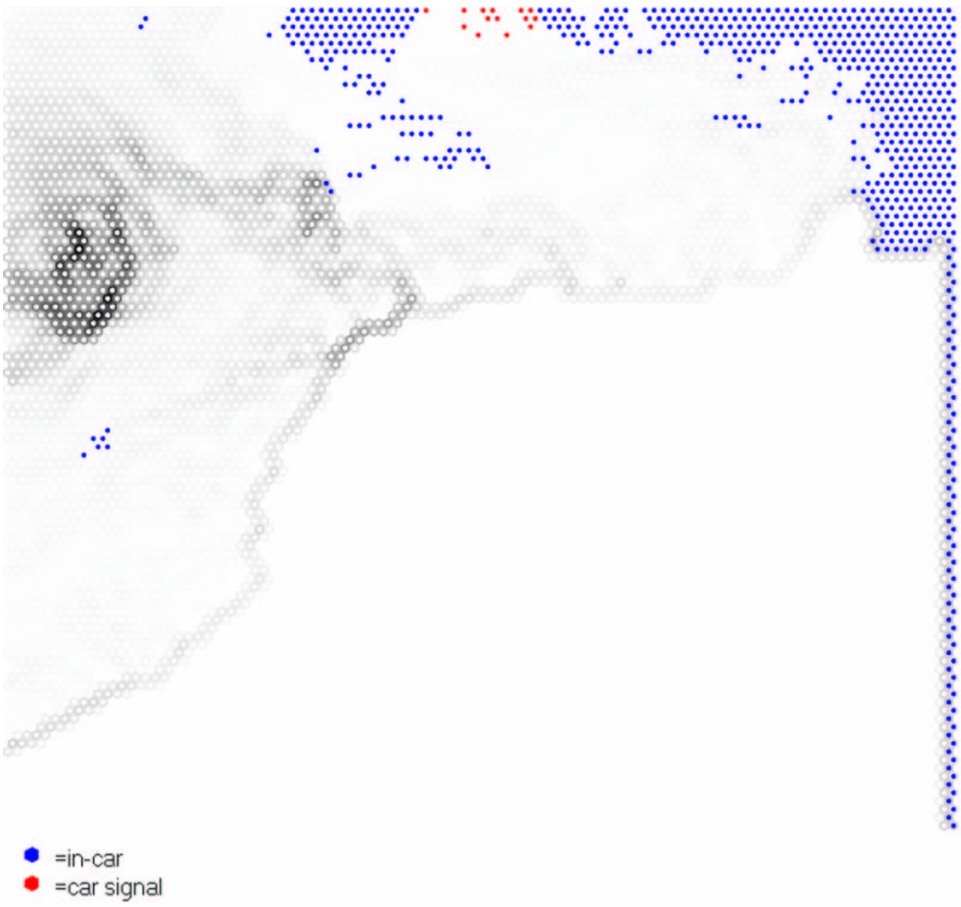


Figure A.3: SOM distribution of in-car and in-car with car signal samples using a 100x100 map and five autocorrelation lags

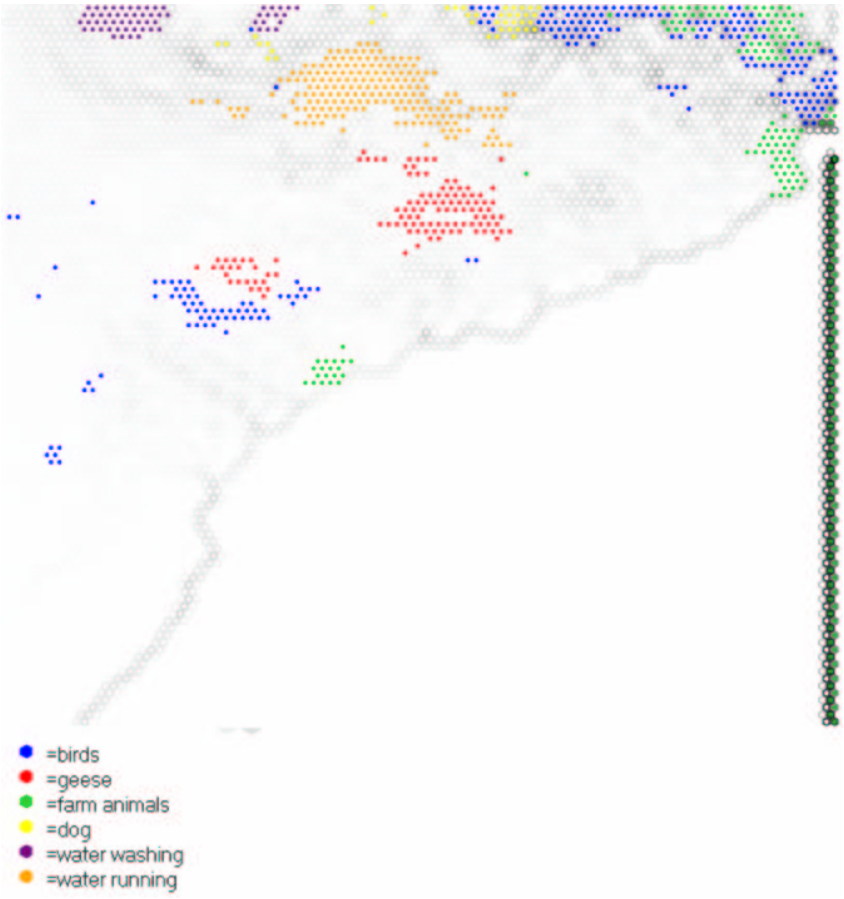


Figure A.4: SOM distribution of all nature samples using a 100x100 map and 10 autocorrelation lags

for everything heard in an outdoor/rural setting. There are, however, very cohesive sub-categories, which are discussed further in Sections A.7, A.12 and A.13 below. These separate groups are illustrated in Figure A.4.

A.7 Birds

The samples with bird noise are quite well clustered in the majority of the tests, but have a large spread in some cases. The SOM tests indicate that the addition of geese to a sound sample is sufficient to create a separate cluster. The addition of farm animals also creates a separate group in some cases, but in the majority of the cases the addition of farm animals does not create a separate group. The addition of dog sounds does not seem to create a separate group. This is discussed further in Sections A.8, A.9 and A.10. Please see Figure A.4 for an illustration of this.

In the K-means, the bird samples are always grouped into at least three, and in most cases four or more different clusters. However, in the large majority of the cases there is one group with a significant majority of the samples.

Bird sounds are also quite a common sound to hear in an outdoor environment and this environment was identified by hearing aid users as being important [10]. These sounds therefore need to be included in the final model. Bird sounds are therefore used as the basis for a cluster in the final system. Neither farm animal sounds nor dog sounds are separated.

A.8 Geese

The samples with both geese and bird tend to be separate from the normal bird sounds in the SOM. Please see Figure A.4 for an illustration of this separation.

In the K-means clustering, the geese sounds are most often clustered in the same

cluster, or a maximum of two different clusters, with one or two outlying samples. Conversely, the bird samples are often clustered into many different groups. There does not appear to be a correlation between the grouping of the majority of the bird samples and the grouping of the geese samples. It appears that the geese noises are dominant and form their own cluster of sounds, separate from the bird sounds.

However, geese sounds are fairly uncommon, and do not tend to be continuous sounds. Hence, it does not make logical sense to include a geese category, as the hearing aid would not likely be able to switch programs in time to actually remove the noise. Therefore, a geese category will not be included in the system, and the birds category will be trained without geese sounds samples.

A.9 Farm Animals

In this database, the farm animal sounds always occur in conjunction with bird sounds. In the majority of the SOMs, the farm animals are grouped with the bird sounds. Please see Figure A.4. In some cases, the farm animals are somewhat separate because the bird sounds are more spread out, and in some cases only some of the farm animal sounds are separate.

In the K-means clustering, the farm animal sounds are clustered into at least three different categories in all of the tested cases. However, the bird samples are also clustered into a number of different categories. The farm animal samples do not appear to be separate from the bird samples.

Hence, the farm animal samples will not be separated from the regular bird samples for the final system.

A.10 Dog

In this database, the dog sounds always occur in conjunction with bird sounds. In the majority of the SOM tests, the dog samples are not separated from the main cluster of bird sounds. Please see Figure A.4. However, this may also have been because there are not many samples with the dog sound, and the dog is fairly quiet in the samples. Tests where the dog sounds are more separate tend to have bird sounds that are also less clustered.

In the K-means tests, the dog samples tend to be spread among more than one cluster. However, the bird samples are also spread across a number of clusters. Because of this spread, it is difficult to determine from the K-means clustering whether or not these two categories are linked.

Because the dog sounds are clustered with the bird sounds on the SOM, the dog samples will be clustered with the bird samples for training. This is also a logical clustering as dog sounds are not continuous and it would not make sense to change the hearing aid program for each dog bark.

A.11 Water Splashing

The samples that include the sound of water splashing are taken from a pool scene with many children and adults talking. The water splashing sounds do not form a cohesive group in either the SOM or the K-means clustering. The SOM tests indicate that there is a large spread between the water splashing sounds. Please see Figure A.5.

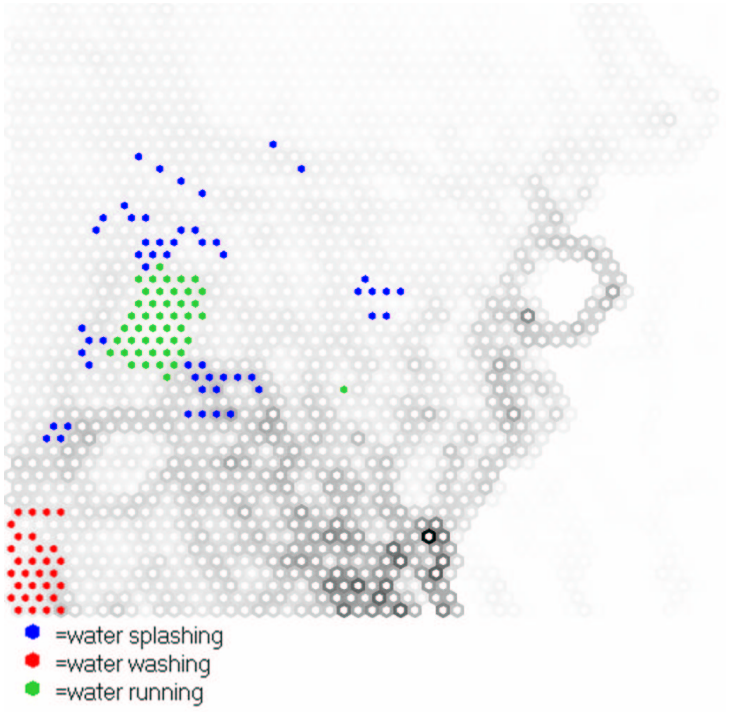


Figure A.5: SOM distribution of water splashing, water washing and water running samples using a 50x50 map and 15 autocorrelation lags

In the K-means tests, the water splashing sounds are always grouped into at least three different clusters, and are grouped into four or more clusters in the majority of the tests. The number of groups this category spans increases as the number of autocorrelation lags increases, and also as the number of overall groups increases.

The sounds of the water splashing appear to be dominated by the other sounds in the samples. Overall, this does not appear to be a good basis for a category and will therefore not be included in the training of the final system.

A.12 Water Washing

The water washing sounds are tightly clustered in both the SOM and the K-means clustering. In the SOM the sounds are always separated from other sounds by higher-distance nodes, indicating that these sounds are not close to any other sound in the sample set. Please see Figure A.5. However, these sounds all come from a single track, which may partially explain this tight clustering.

The K-means clustering also shows that the water washing samples are very tightly clustered. In the majority of the tested cases, the water washing samples are all clustered in the same group. In some cases there are one or two samples in a different group. The samples including bird sounds are not separate from the main grouping in either the SOM or the K-means clustering. Overall, this group appears to be quite cohesive, and quite different than all the other groups. Water washing sounds are therefore used as the basis for a group of sounds.

A.13 Water Running

The water running sounds are tightly clustered in both the SOM and the K-means clustering. In the SOM the sounds are always separated from other sounds by higher-distance nodes, indicating that these sounds are not close to any other sound in the sample set. Please see Figure A.5. However, these sounds all come from a single track, which may partially explain this tight clustering.

The K-means clustering also shows that the water washing samples are very tightly clustered. In the majority of the tested cases, the water running samples

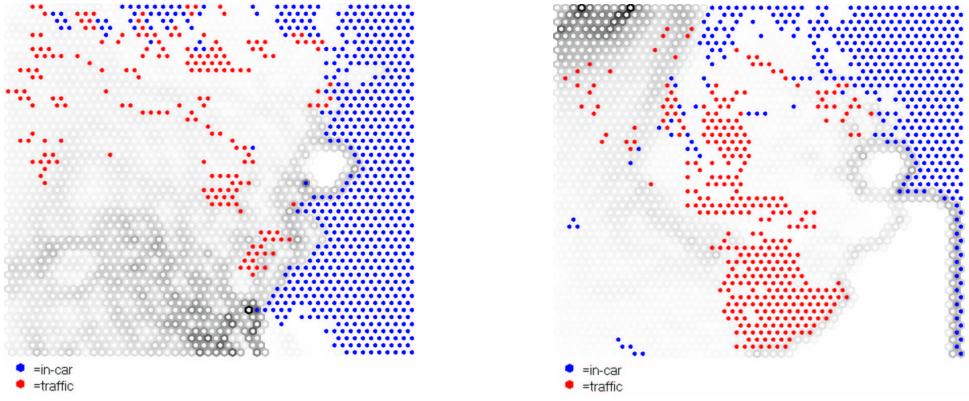
are all clustered in the same group. In some cases there are one or two samples in a different group. This separation occurs when the number of autocorrelation lags is large.

Surprisingly, the sound of water washing is quite different than the sound of water running. In the SOM these two groups are separated by either a large number of nodes, or by nodes that are a far distance apart. In all cases except one, these are also clustered separately in the K-means clustering. Although these are both water sounds, the sound of water running is fairly continuous and sounds similar throughout the sample. The sound of water washing is more periodic and changes throughout the sample. This would affect the autocorrelation, which may explain why there is such a large difference in these samples.

A.14 Traffic

In some SOM tests, the traffic samples are spread across a large portion of the map. This may be partially due to the fact that there are a large number of samples that contain traffic sounds as secondary noises. However, in these cases even the samples that contain only traffic noise have a great deal of spread. In other tests, the traffic samples form a more cohesive cluster. Please see Figure A.6. The traffic noises do not seem to be correlated with the in-car sounds. The placement of the traffic samples is most often between the in-subway and the in-car sounds.

The K-means clustering also shows this spread. In all cases, the traffic samples are placed into two or more separate categories, and in most cases the traffic is clustered into three or more clusters. There does not appear to be a dominant



a)spread b)clustered

Figure A.6: SOM distribution of in-car and traffic samples using a 50x50 map. a) uses 5 autocorrelation lags and shows a map where the traffic samples are spread. b) uses 15 autocorrelation lags and shows a map where the samples are more clustered.

cluster in any case. Additionally, the K-means clustering classifies some traffic samples with the in-car samples, and some with the subway samples. This is similar to the results from the SOM, which indicate that the traffic samples are spread between these two groups.

The reasons for this large split are unclear. It is possible that the different types of cars, different rates of acceleration or different numbers of cars in the audio scene may produce different types of autocorrelation profiles for the samples.

However, despite this large spread, the traffic samples are used to form the basis of a cluster simply because it is such a common and important sound. These are clustered with the subway samples because they are quite close on the SOM.

A.15 Train/Tram

In both the SOM and the K-means clustering, the train/tram samples are sometimes clustered with the traffic samples, and are sometimes clustered with the in-car samples. Training the network with the train/tram samples included in either one of these categories would likely reduce the ability of the system to distinguish these categories from each other. Hence, the train/tram samples will not be included in the training of the system. In a real implementation of this system, a hearing aid encountering these sounds would likely be categorize them as being in one of the traffic or in-car categories, depending on which cluster the particular sound was closest to.

A.16 Subway Station

The subway station samples are categorized with the in-subway samples. For further discussion of the subway station samples, please see Section A.4.

A.17 Footsteps

The footsteps samples are spread across a large portion of each of the SOMs. Please see Figure A.7. They are also spread across at least three clusters in all cases of the K-means clustering. Even the samples containing only footsteps sounds are clustered in two or more different clusters in all cases. The large spread may occur because there are so many samples that include footsteps as a secondary sound that they cover a large portion of the SOM and spread across many clusters simply due

to their number. However, samples containing only footsteps are also not clustered into a single region in the SOM or in a single group in the K-means clustering. Samples that have footsteps in combination with other sounds tend to be clustered with the primary sounds.

From this, it would appear that footsteps are not a good basis for a sound cluster. Footsteps that occur in combination with other sounds should be clustered with the primary sounds where appropriate. This is also logically appropriate for a hearing aid, as footsteps are quite transient sounds, and it is not desirable to change the hearing aid program each time there are footsteps that interrupt a primary background sound.

A.18 Babble

The babble samples are also quite spread across the SOMs and are not consistently clustered in the K-means. This may be partially due to the fact that there are a large number of samples that contain babble.

However, babble does not appear to be a cohesive category of sound. The SOM shows that the babble category is spread across the majority of the map (see Figure A.8), and the K-means clustering shows babble samples in at least three different categories in every run. It may be that "babble" is actually a more general case of specific sound environments that include babble. This does not mean that babble is not a problem for hearing aid users, but rather that it should likely be removed differently depending on the external environment.

Therefore, babble is not included as a separate category itself, but is split into

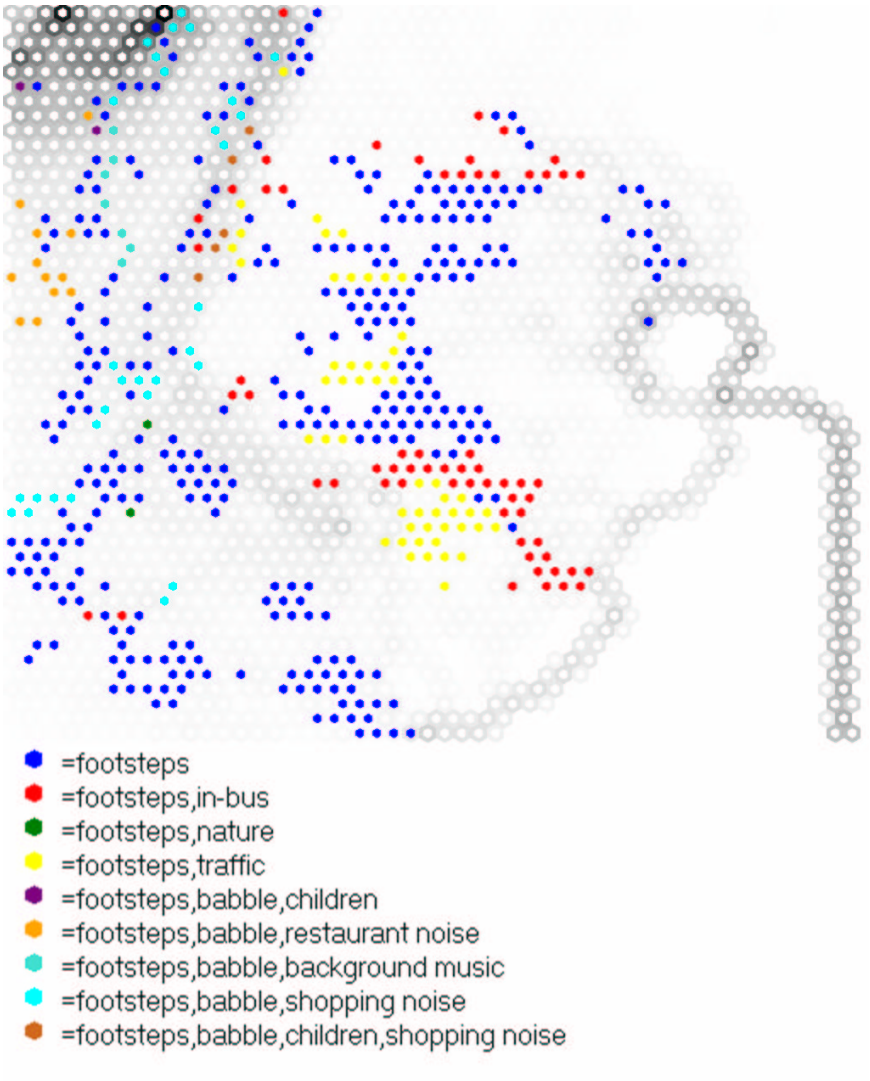


Figure A.7: SOM distribution of footsteps samples using a 50x50 map and five autocorrelation lags

several subcategories, each of which contain a large amount of babble noise. This is discussed further is Sections A.22, A.23 and A.26.

A.19 Children

The samples that include children sounds are often either intermixed with the babble samples, or are directly next to the babble samples. The samples are very scattered in the SOM tests and are grouped into at least three different groups in every K-means test. These groups are usually spread relatively evenly over a number of these groups, with no one group having a noticeable majority of the samples.

This group appears to be quite close to the babble group, and is not very cohesive. This group should not be used as the basis of a group of sounds. Sounds including children are included in other groups as appropriate.

A.20 Laughter

Laughter is not a cohesive category, and tends to be clustered with the primary sound in the sample. There is a large spread in this category in all of the SOM tests, and laughter tends to be dominated by the primary sound. Please see Figure A.9.

In the K-means clustering, laughter samples are clustered in three or more different categories in every K-means test, and the number of clusters with laughter samples increases as the overall number of categories increases.

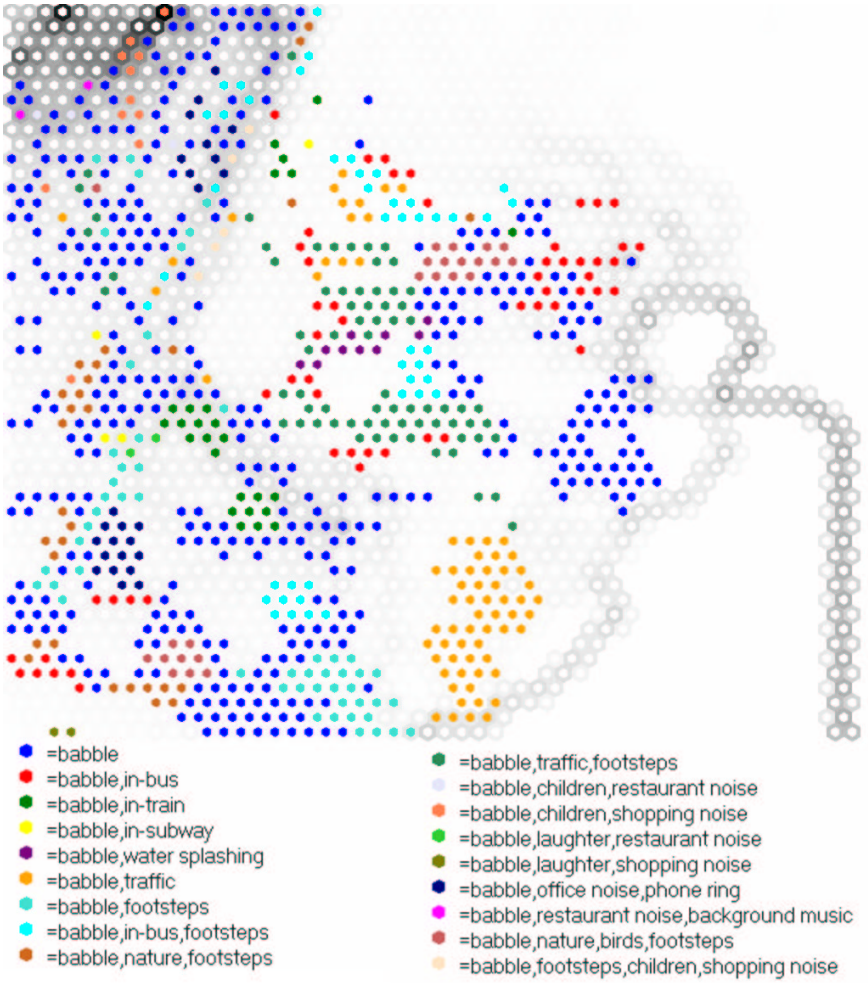


Figure A.8: SOM distribution of babble samples using a 50x50 map and five autocorrelation lags

Overall, it appears that laughter samples should not be separated as their own category, but should instead be classified with the primary sound where appropriate. This also makes sense logically as laughter is often quite short in duration and it would not be appropriate to switch the hearing aid program each time someone laughed.

A.21 Applause

Applause is actually a very cohesive category. In the SOM tests, the applause category is always highly cohesive, and does not appear to be correlated with any other sound. Please see Figure A.10.

In the K-means clustering, the applause samples are also always heavily clustered, with applause samples occupying only one category in the majority of cases. In cases where the applause appears in more than one cluster, it is at most two samples that are outliers.

However, logically, applause does not make a good basis for a category. Applause would be heard only on rare occasions, and it is not likely that a user would be attempting to hear a principle sound over the sound of applause. It also does not appear to be similar to any other category, since it is not regularly clustered with any other category in the K-means and does not regularly appear next to any category in the SOM clustering. Hence, applause will not be included in the training of the final classification system.

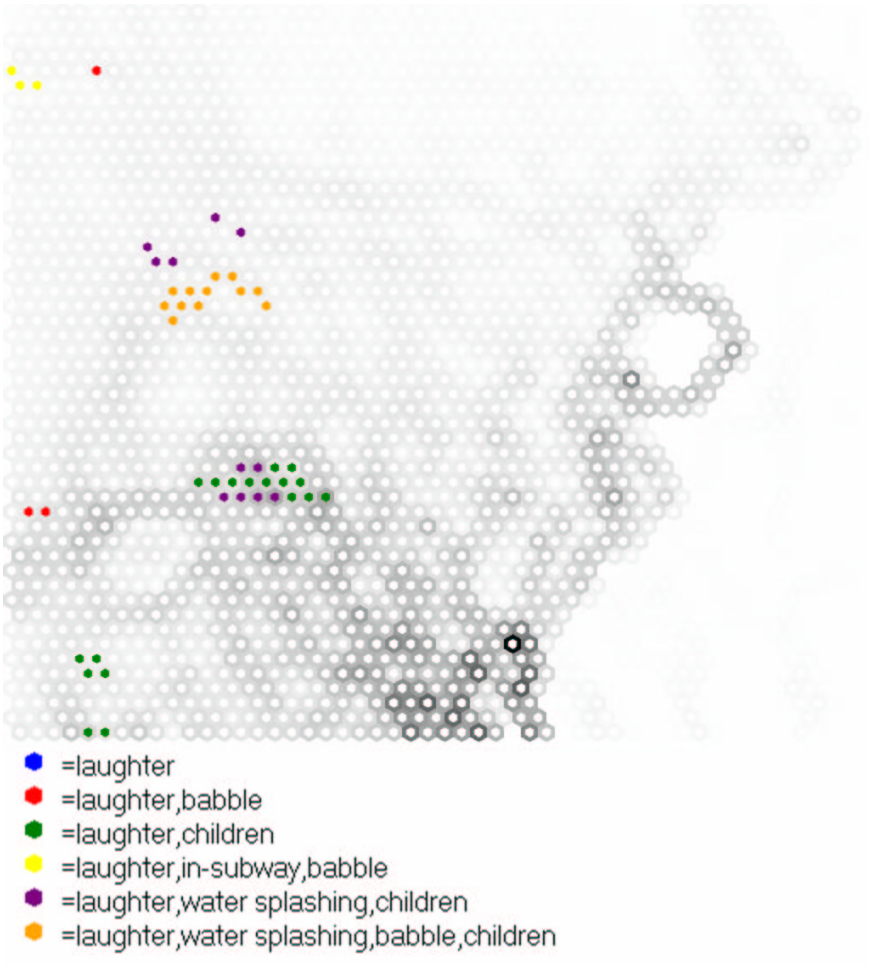


Figure A.9: SOM distribution of laughter samples using a 50x50 map and 15 auto-correlation lags

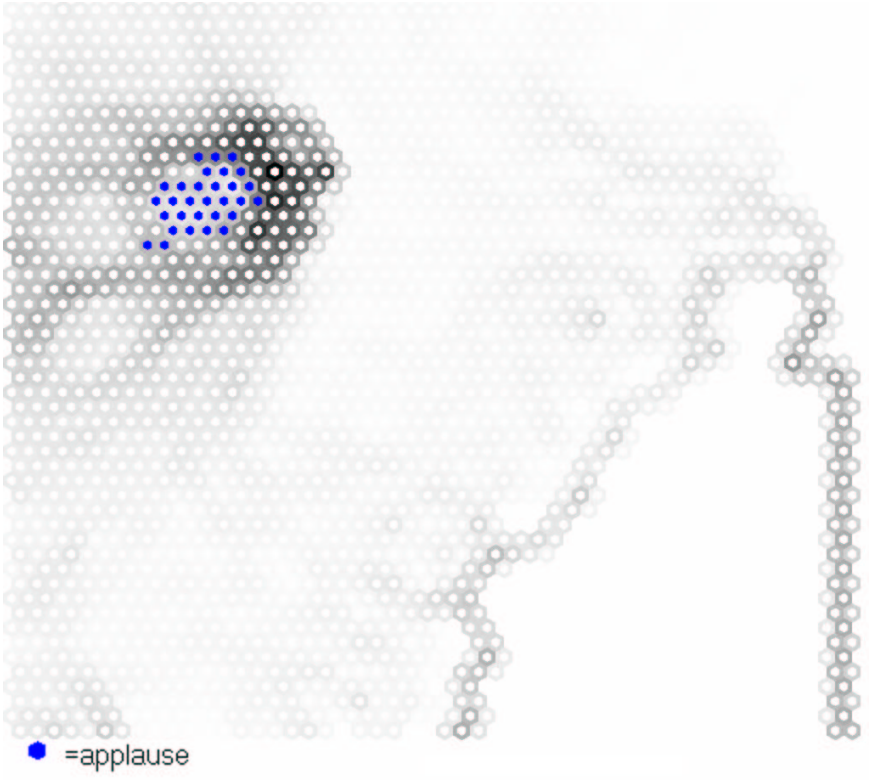


Figure A.10: SOM distribution of footsteps samples using a 50x50 map and 10 autocorrelation lags

A.22 Office Noise

The office noise samples form a relatively cohesive category in the majority of the SOM tests. However, these groups are quite large and can be spread over a relatively large portion of the map. Please see Figure A.11. The samples that include the phone noises are somewhat separate from the other office noise samples in the SOM, but are located next to the regular office noises. They are also not normally separated in the K-means clustering. Please see Figure A.12.

In the K-means clustering, the office noises are always clustered into at least three different groups, which is an indication that this group may not actually be as cohesive as SOM tests indicate. However, there is usually one group that contains a noticeable majority of the samples. The cluster with the majority of the samples is different than the cluster with the majority of the shopping or restaurant noise in the large majority of the tests.

For these reasons, the office noises will form the basis of a cluster, and will be separated from the shopping and restaurant noises. The samples that contain phone noise will not be separated. This also makes sense logically as the phone ring sounds are not continuous and occur only infrequently. It would be difficult and distracting to switch the hearing aid program each time this sound occurred, especially since the duration of the phone sound is likely shorter than the time it would take to switch the program. Additionally, phone ring sounds are likely sounds that the user would like to hear.

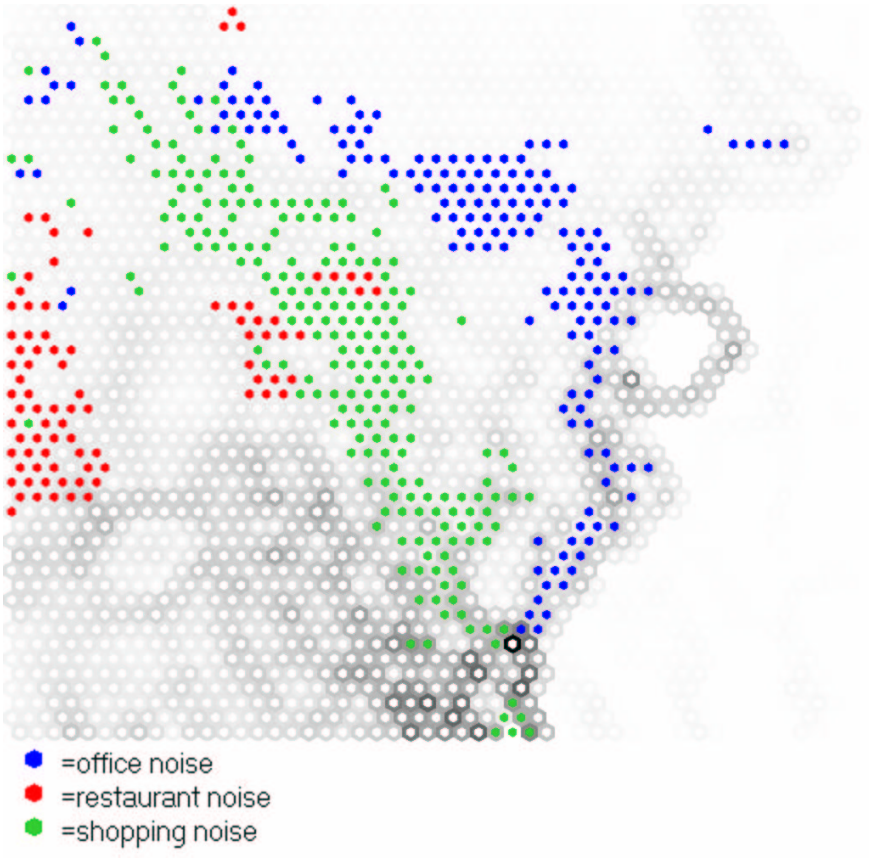


Figure A.11: SOM distribution of shopping, office and restaurant noise samples using a 50x50 map and 15 autocorrelation lags

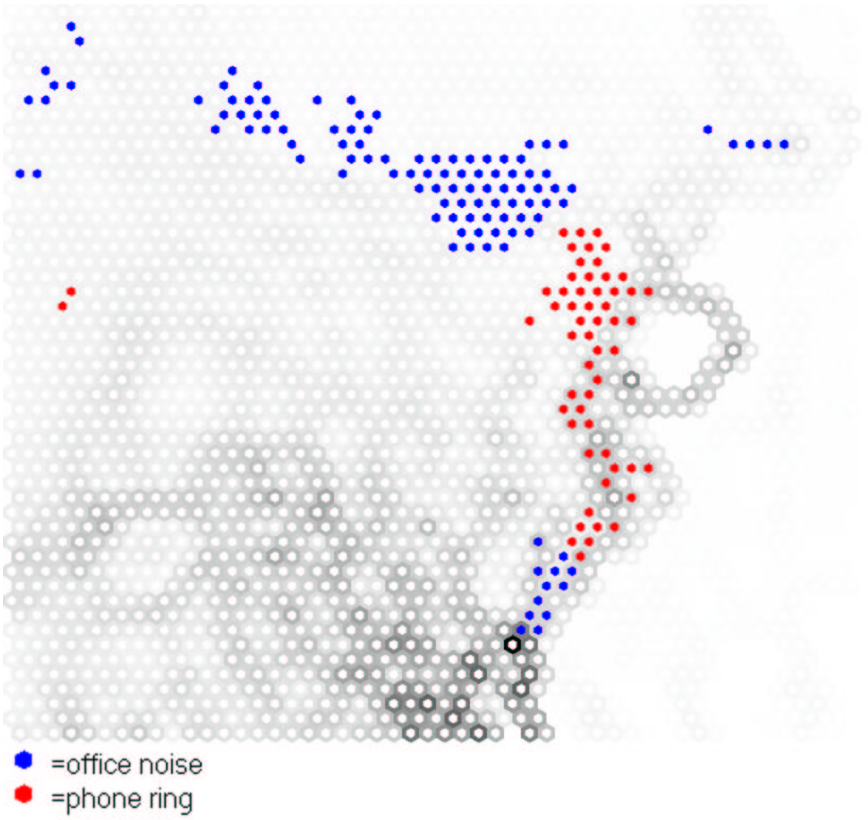


Figure A.12: SOM distribution of shopping, office and restaurant noise samples using a 50x50 map and 15 autocorrelation lags

A.23 Restaurant Noise

Of the three main types of babble noise (restaurant noise, shopping noise and office noise), the restaurant noise is the least cohesive category in the SOM tests. In some cases the restaurant noise is relatively tightly clustered, but in other cases there is a large spread. In the majority of the cases, the shopping noise samples are clustered between the restaurant noise and the office noise, which indicates that the restaurant noise and the office noise should not be clustered together if the shopping noise is separate. Samples that include secondary noises do not appear to be separated from the main group of restaurant noise samples. Please see Figure A.11.

In the K-means clustering, the restaurant noises are always clustered into at least three different groups, which is also an indication that this group is not very cohesive. However, there is usually one group that contains a noticeable majority of the samples. The cluster with the majority of the samples is different than the cluster with the majority of the shopping or office noise in most cases.

Therefore, the restaurant noise is clustered separately from both the shopping and office noise sounds, and the samples with the secondary sounds are not separated.

A.24 Electronic Sounds

The electronic sounds have a large spread in all of the SOM tests. Please see Figure A.13. The electronic sounds are also spread across a number of different groups in

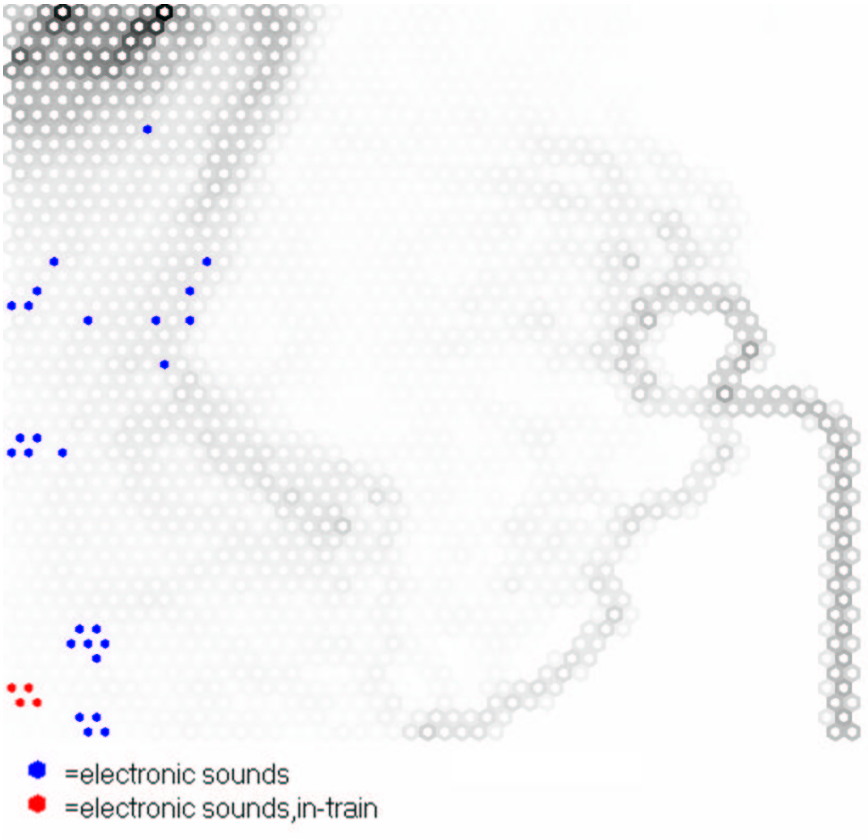


Figure A.13: SOM distribution of electronic sound samples using a 50x50 map and five autocorrelation lags

all the K-means tests. This does not appear to be a very cohesive category.

Electronic sounds also tend to be infrequent and short, hence it would not make sense to include these as a separate category. Instead, these types of sounds can be grouped with the other sounds where appropriate.

A.25 Music

The music category actually tends to be dominated by the other sounds in the sample. There is a large correlation between the in-car samples and the in-car/music samples in both the SOM and K-means tests. In both, the in-car and in-car/music samples are categorized together in a separate cluster from the other music samples.

The sets of samples tagged as restaurant noise and music or shopping noise and music are both also correlated. However, the correlation is not as strong as the in-car samples. This may also be because the restaurant noise and shopping noise samples are not as tightly clustered. However, the addition of the music clearly does change the samples enough to create a separate cluster.

The samples with only music, however, do tend to be clustered in the SOM. The K-means clustering process also tends to have one group that contains a large majority of the samples of music. Therefore, music sounds that occur with another sound will be classified with the primary sound, but sounds that are only music will be clustered in their own group. This also makes sense logically as music that occurs alone or as a primary sound is likely something that the user actually wants to hear, and which would likely require different settings on the hearing as the frequency range would be different.

A.26 Shopping Noise

In the SOM tests, the samples with shopping noise tend to be somewhat spread. The samples either form large, long clusters, or two or three smaller clusters. In

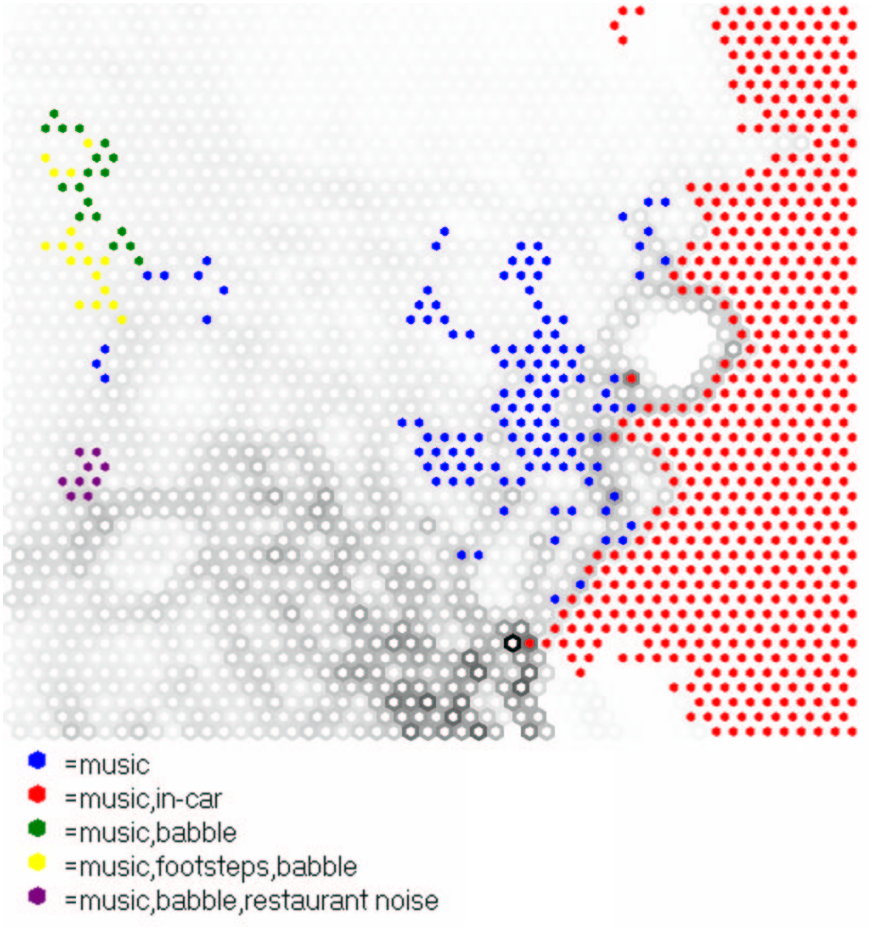


Figure A.14: SOM distribution of music samples using a 50x50 map and 15 auto-correlation lags

most cases, the shopping noise samples are between the office noise and restaurant noise. Please see Figure A.11. In all cases, however, these clusters of shopping noise appear to be separate. Additionally, samples that include secondary noises are not separate from the main clusters, indicating that the shopping noise is actually the primary sound in these samples. Please see Figure A.11.

In the K-means clustering, the shopping noise always spans at least three clusters. There is normally a noticeable majority in one of the clusters. The clusters occupied by the shopping samples are the same clusters as the restaurant noise and office noise samples. However, the office noise and restaurant noise samples normally have the majority of the samples in different clusters. Similar to the SOM clustering, the samples with secondary noises do not form their own separate clusters.

For these reasons, the shopping noise category is clustered separately from the restaurant noise and office noise categories. The samples with secondary noises are included with the shopping noise samples.

A.27 Phone Ring

The phone ring sounds occur in combination with office noise, and also in combination with office noise/babble. The samples that include the phone ring noise are somewhat separate from the office noise in the SOM, but they are normally connected or in the same region. Please see Figure A.12. These samples are not separate in the K-means clustering. Therefore the samples with the phone ring sounds will be included with the office noise sounds for the final system. This also

makes sense logically as the sound of a phone ringing is not constant and occurs infrequently. It would be difficult and distracting to switch the hearing aid program each time this sound occurred, especially since the duration of the phone sound is likely shorter than the time it would take to switch the program. This category would also be used very infrequently. Additionally, phone ring sounds are likely sounds that the user would like to hear and should therefore not be removed.

A.28 Industrial

The industrial sounds do appear to be somewhat clustered in the SOM. However the samples cover a relatively large area of the map, and they are usually located in nodes which are neighbours, but where the distance between the nodes is relatively large. Please see Figure A.15.

In the K-means clustering, the industrial samples are always distributed among four or more groups, and the division between these groups is relatively even. It does not appear that industrial sounds are a very cohesive group. They also do not appear to be similar to any other group.

Because these sounds are quite diverse, industrial sounds should not be used to form the basis of a cluster, and will therefore not be included in the final system.

Additionally, the sheer volume of most industrial workplaces would make it difficult to use a hearing aid. Most industrial workplaces would also require their own hearing protection. This type of environment is likely not a place where a user would be attempting to use a hearing aid.

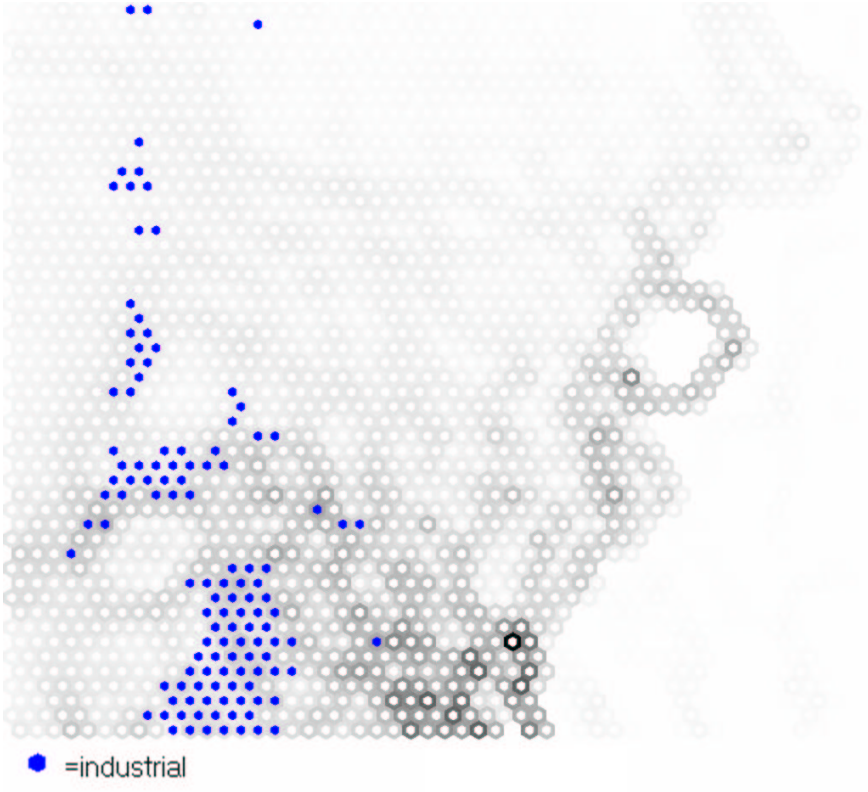


Figure A.15: SOM distribution of industrial samples using a 50x50 map and 15 autocorrelation lags