Fall Detection Using Acoustic Features and One Class Classifiers

A Thesis presented to the Faculty of the Graduate School

University of Missouri-Columbia

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

by

ABHISHEK MAHNOT

Dr. Mihail Popescu, Thesis Supervisor

July 2009

The undersigned, assigned by the Dean of Graduate School, have examined the thesis entitled

Fall Detection Using Acoustic Features and One Class Classifiers

Presented by Abhishek Mahnot

A candidate of degree of Master of Science

And hereby certify that in their opinion it is worthy of acceptance.

Dr. Mihail Popescu

Dr. James Keller

Dr. Marjorie Skubic

ACKNOWLEDGEMENTS

I would like to thank Dr. Mihail Popescu for his assistance, guidance, and support with the research and writing of this thesis. All of his help and dedication is greatly appreciated.

I would also like to express my sincere gratitude to my committee members Dr. James Keller and Dr. Marjorie Skubic for spending their valuable time and giving helpful suggestions to improve my research. Also, student colleague Yun Li, provided valuable collaboration and assistance throughout my graduate studies

I thank University of Missouri – Columbia, for giving me the opportunity to pursue my Master's degree and obtain high quality education and wonderful experience that added a lot in my personality and made me a better person to face my future life.

Abstract

With the increasing of elderly population, there are more and more health problems occurring in everyday activities among this group of people. An investigation shows many elderly people get injures or trigger more serious health problems due to falling on the floor at their home or hospitals without artificial monitoring. There are many techniques to monitor the fall remotely and provide assistance as soon as possible. For this purpose video cameras are deployed at the place of living of an elderly but his might lead to an uncomfortable feeling of being spied on, hence we try to use just the sound (mainly frequency domain features) instead of video to detect a fall remotely. Sound signal is collected for a falling person along with normal everyday sounds, a classifier is trained using these sounds and using these we try to do the classification of an unknown sound as fall or non-fall. The next problem though is how to collect exact sound sample of a falling person as that is the first thing we want to avoid. In this work therefore we try to train our classifier using data from only one of the two classes (fall and non-fall). We try to do the classification using sound samples of normal everyday sounds only. These classifiers are called one class classifiers as they do the classification using samples from only one of the two classes. We compare the performance of these classifiers with the conventional two class classifiers which uses examples from both the classes by testing both on same dataset. Acoustic feature that we use to do classification is Mel Frequency Cepstrum coefficients or MFCC in short

Table of Contents

List of Tables

List of Figures

Figure **Page**

Chapter 1: Introduction: Fall Detection

1. Introduction

Pattern recognition or **classification** is a sub-topic of machine learning. It is the act of taking in raw data and classifying it in one of the known classes. This thesis will focus on this classification problem in which only one class is considered (Tax. 2004). Here an object is classified as a genuine or an outlier object based on samples of only one type of objects. The emphasis in this thesis is on the classification of the sound signals. The genuine object would be any ordinary sound like sound of a door closing, keys or books falling and an outlier object would be the sound of a person falling. Hence, in a nut shell by performing this classification we are actually doing: 'Fall Detection'. And since this classification is done using one-class classifier, hence the title of this thesis: *'Fall Detection Using One Class Classifier'*. Before we go into the details of one class classifier algorithms, it is important to understand why this problem of detecting a person's fall is important.

1.1 Importance of Fall Detection

In Europe (Istrate et al. 2006) it is estimated that 1/3 of population would be more than 65 years old by the year 2035 whereas in US the number of people 65 years or older is estimated to be around 53 million by the year 2020. This clearly indicates that elderly population is the fastest growing segment of population in developed nations. Falls are the leading cause of injury-related visits to emergency departments in the United States

and the primary etiology of accidental deaths in persons over the age of 65 years. The mortality rate for falls increases dramatically with age in both sexes and in all racial and ethnic groups, with falls accounting for 70 percent of accidental deaths in persons 75 years of age and older. Falls can be markers of poor health and declining function, and they are often associated with significant morbidity. Fear, anxiety or depression also increases due to risk of fall event among elderly. More than 90 percent of hip fractures occur as a result of falls, with most of these fractures occurring in persons over 70 years of age. One third of community-dwelling elderly persons and 60 percent of nursing home residents fall each year. Falls are a major health hazard for the elderly and a major obstacle to independent living. Many elderly people also wish to live in their own houses or apartments away from nursing home, therefore the problem of fall can be a hindrance to that as they might be required to live under constant care. In fact, elderly people underline that an efficient fall detection system would dramatically improve their life quality.

Beside eldercare, it is also desirable to reduce Public Health Services costs. Having a system in one's house which sounds an alarm in situation of distress would greatly reduce these costs (Jennett et al. 2003, Bashshur 2002) as there would be no need of a nurse (for constant supervision) or to admit someone in nursing home (which are very often under staffed) just to avoid fall. Clearly, detection of a falling person in an unsupervised area is a practical problem with applications in safety and security areas including supportive home environments. All this indicates the necessity of **automatic fall detection system**.

1.2 Motivation for Using Sound

For monitoring domestic environments for a distress situation (example: falling of an elderly), both video and various forms of motion sensors have been used. It is also suggested that networking various household appliances can also be used as a measure of domestic activity. But we believe there are some strong advantages of using audio (Schmandt et al. 2003).

First, it is easier to place microphones as compared to cameras or motion sensors, where field of view is a major consideration. Microphones are more nearly omni-directional than even wide-angle camera lenses, and because sound is more accurately reflected off more surfaces than light, there are fewer "blind" spots when *eavesdropping* instead of *peeping in*. Microphones may be more easily hidden, or much less visible. While we are not advocating surreptitious monitoring, residents of a monitored domicile may be more comfortable without the monitoring technology in their faces.

Second, audio requires significantly less bandwidth to transmit. Actual implementation of automatic fall detection architecture calls for multiple points of capture in the house, with wireless transmission to a server. It is also desirable to support mobile clients such as phones on current wireless telephone networks. Video, even slow at slow frame rates, may tax such networks.

Finally, a video window on a computer screen requires visual contact, i.e., being close to the screen and looking at it. Auditory cues can be heard regardless of where the recipient is looking. Even if a computer user is monitoring via a small video window in the corner of his or her screen while doing other work, motion in one's peripheral vision evokes a reflex to look at the source of motion, and would be continually distracting. Using sound

as an alternative solution to video improves the patient's comfort and reduces hospitalization costs without intruding anyone's privacy.

In this Thesis, we present a system for the detection and classification of everyday life sounds. The aim of our research is to develop a fall detection system using sound sensors (microphones). In future we might consider an array of microphones to improve beam-forming. The telemonitoring system must cover all the areas of the apartment, including the toilets, the bathroom, and the bedroom. If a video camera is installed in every room, the patient could have the uncomfortable feeling of being spied on. The originality of this research is to use sound as an informative source. We propose to extract and classify everyday life sounds such as: door banging, glass breaking, sounds of telephone, falling objects, person, books, etc. in the aim of detecting falls. Thus, our approach consists of replacing the video camera by a multichannel sound acquisition system that analyzes the sound environment of the apartment in real time and detects distressful situations. The sound analysis system is divided in two stages: sound detection and classification. The first analysis stage (sound detection) must extract significant sounds from a continuous signal flow. The second stage of the system is sound classification, which uses a statistical approach to identify unknown sounds. For first stage we use Digital Signal Processing concepts like Wiener filter and setting an energy threshold. For second stage i.e. classification, which is more important, pattern recognition concepts are used like k-nearest neighbor algorithm and support vector algorithm. In this thesis one class classification is used for second stage which is then compared with the conventional classifiers. Next chapter gives the details of our *acoustic feature set* and *one class Classifiers*.

Chapter 2: Literature Review

Before discussing classifiers, it is important to see what features we use for doing the classification, as the features are going to decide how well our classifier works.

In our project, we prefer to use microphone array to detect falls. The experimental procedures and details about the test data and training data will be described in Chapter 3. In the following we will state our approach for acoustic detection. Details of computation of MFCC features are given in the next section:

2.1 Mel Frequency Cepstral Coefficients or MFCC feature set

MFCCs (Mel-scale Frequency Cepstral Coefficients) are the most commonly used acoustic features in speech/speaker recognition system. MFCCs take human perception sensitivity with respect to frequencies into consideration and are therefore best for speech/speaker recognition (Kinnunen et al. 2007, Li et al 2005, Schmandt et al 2003). Moreover, most audio/acoustic signal retrieving or detection techniques use MFCCs as the core features for recognition, since MFCCs have a good modeling of sound signals. In our fall detection project, we propose to use MFCC features to distinguish fall and non-fall sounds using two class and one class classifiers. The computation of MFCCs is done through 6 steps illustrated by the following block diagram (Ripley, B 1996).

Step 1: Pre-emphasis

The purpose of pre-emphases filters is to compensate the high-frequency part that was suppressed during the sound production. It is implemented by sending the input to a high-pass filter.

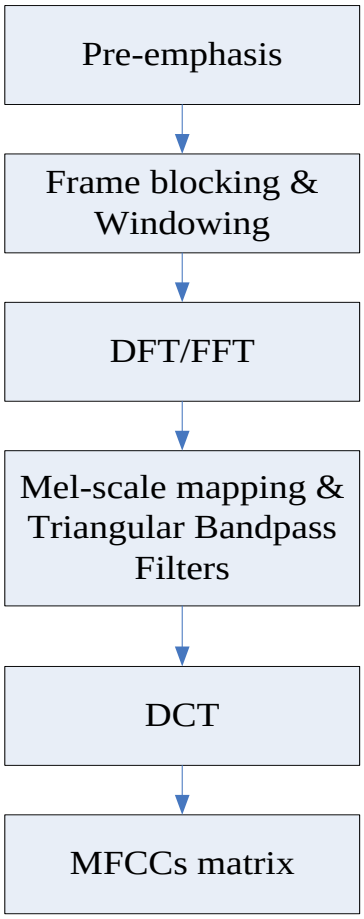

Figure 1: Flow blocks for computing MFCCs

Step 2: Frame blocking & Hamming Windowing

The input frame of signal is further segmented into sub-frames, each of which has a frame size of 20-30ms. There is an overlap of 1/2-1/3 between each of the sub-frames. The sub-frame rate is 100 sub-frames/second. The sampling frequency for the sub-segmentation is picked at 12000samples/second based on experimental results. Since we have defined that in that each processing frame we will have 20000 samples (equal to sampling frequency), we can compute the number of sub-frames in a processing frame:

[(20000-256)/(12000/100)]=164. Each sub-frame has to be multiplied with a hamming window in order to keep the continuity of the first and the last points in the sub-frame.

Step 3: Fourier transform

Spectral analysis shows that different timbres in the acoustic signal correspond to different energy distribution over frequencies. Generally to make the computation more efficient, we use FFT (Fast Fourier Transform) instead of DFT (Discrete Fourier Transform). Thus FFT is applied to each sub-frame which has been windowed by Hamming window.

Step 4:Mel-scale mapping & Triangular Bandpass Filters

Mel-scale mapping is an important technique in cepstrum extraction. Mel-frequency is proportional to the logarithm of the linear frequency, reflecting similar effects in the human's aural perception. The mapping relation is given by the equation

$$Mel(f) = 1125\ln(1 + f/700) \qquad (1)$$

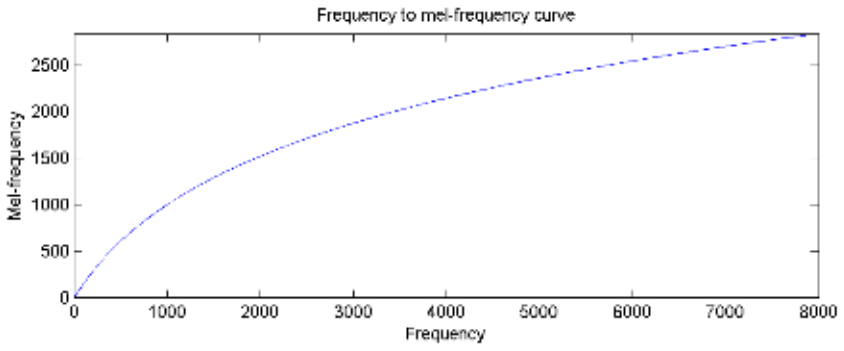

Figure 2: Frequency to Mel-Frequency curve

where f denotes the linear frequency. Based on this relationship, we can assign an array of triangular bandpass filters over the Mel-frequency with equal interval spaces but with logarithm-based spaces over linear frequency to multiply the frequency response to get the energy of each bandpass filter. Here we define total 30 triangular bandpass filters: 10 linear filters below 185Hz and 20 log filters above 185Hz.

Step 5: Discrete Cosine Transform (DCT)

We apply DCT on the 30 log energy denoted as E_n obtained from the triangular bandpass filters to have M Mel-scale cepstral coefficients. That is

$$C_m = \sum_{n=1}^{N} \cos\left[\frac{m(n-0.5)\pi}{N}\right].E_n \quad m = 1,2 \ldots . . M \qquad (2)$$

In the above equation N is the number of triangular bandpass filters and in our case $N = 30$, M is the number of mel-scale cepstral coefficients which is specified based on the optimum operation of ROC curve. The obtained features C_m are referred to as MFCC.

Step 6: MFCCs matrix

Now we have got the MFCC features for each sub-frame. However, in order to compare the whole processing frame with others, we have to arrange the MFCCs in a matrix indicating the features of all sub-frames. The number of sub-frames in each 1 second frame is 164 , therefore the feature matrix is defined as

$$\overrightarrow{C_k} = \begin{vmatrix} C_{k1-1} \cdots & C_{k1-164} \\ \vdots & \vdots \\ C_{km-1} \cdots & C_{km-164} \end{vmatrix} \qquad (3)$$

where k denotes the k th processing frame in the recording file.

2.2 Spectral Based Features extraction methods & analyzing

Besides using above explained MFCC as features for doing our classification we can also use other spectrum based features to do the classification of sound signals (Liu et al. 1998). These features are: Band-Width, Centroid Frequency, Zero Crossing Rate (ZCR), Energy Ratio Sub-Band (ERSB) which may characterize each kind of the sound event. Only ZCR is extracted in time domain all other features are obtained from the Energy Density Spectrum. Below we describe Energy Density Spectrum (EDS) of falls and non-falls. Following that we describe the features based on energy density spectrum.

2.2.1 Energy Density Spectrum (EDS)

In the frequency domain, we are more interested in the distribution of signal energy as a function of frequency. Experiments tell us that energy distribution could make significant difference among kinds of acoustical signals. Let us consider the computation of the spectrum of a deterministic signal from a finite sequence of data. Suppose a digital sequence $x(n)$ is the sampling representation of a continuous-time signal $x(t)$ with sampling frequency f_s. Now provided that $x(t)$ is a finite-energy signal which satisfies

$$E = \int_{-\infty}^{\infty} |x(t)|^2 dt < \infty \qquad (4)$$

Since the energy is up bounded, the Fourier transform of $x(t)$ exists and is computed as

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j\,2\pi f t}\, dt \qquad (5)$$

According to Parseval's theorem, we get

$$E = \int_{-\infty}^{\infty} |x(t)|^2 \, dt = \int_{-\infty}^{\infty} |X(f)|^2 \, df \quad (6)$$

And energy density spectrum of the signal is defined as

$$S(f) = |X(f)|^2 \quad (7)$$

Here$|X(f)|^2$denotes the distribution of energy among the frequency.

Suppose we have a sampled sequence $x(n)$ with fixed size of N. The computation of $X_n(f)$ is based on a set of N points by means of the DFT (Discrete Fourier Transform)

$$X_N(f) = \sum_{n=0}^{N-1} x(n) e^{-j\,2\pi f n} \quad (8)$$

Then

$$S_N(f) = |X_N(f)|^2 = |\sum_{n=0}^{N-1} x(n)\, e^{-j\,2\pi f n}\,|^2 \quad (9)$$

In most cases we want the frequency to be represented by a set of indices from 0 to N-1, and also from signal processing we know that $f = \frac{k}{N}$ hence we have

$$S_N(f)|_{f=k/N} = |X_N(f)|^2 = |\sum_{n=0}^{N-1} x(n) e^{-j\,2\pi \frac{k}{N} n}\,|^2 \qquad k = 0, \ldots . N-1 \quad (10)$$

and we use a new symbol$\hat{S}_N(k)$ to describe the EDS as

$$\hat{S}_N(k) = S_N(f) \quad (11)$$

This is to obtain a sequence of indices by sampling the frequencies, which is easy to process by computer. So far, the EDS is built for estimating the spectrum of each window-size signal. For the purpose of analyzing the EDS of the interested kinds of sound events, we plot EDS of one training piece of human fall and non fall signals. Each

training piece of signal has the same size as the processing window size N , that is $2X10^4$ points, the same as the signal sampling frequencyf_s. Specially, according to the property of frequency sampling in DFT, we have the representation of frequency f in Hz as

$$f = \frac{k}{N} f_s \tag{12}$$

Below we show the typical EDS for a fall and a non-fall (Note that we only draw symmetric EDS over half of the total points). To save the space, we provide the EDS of only one fall and non-fall here. However, from analyzing all the EDS of the sound samples, we conclude that most of the human falls have a concentrated energy distribution within a very low frequency band of 0Hz to average 600 Hz (shown below). While a non-fall has a more evenly distributed spectrum.

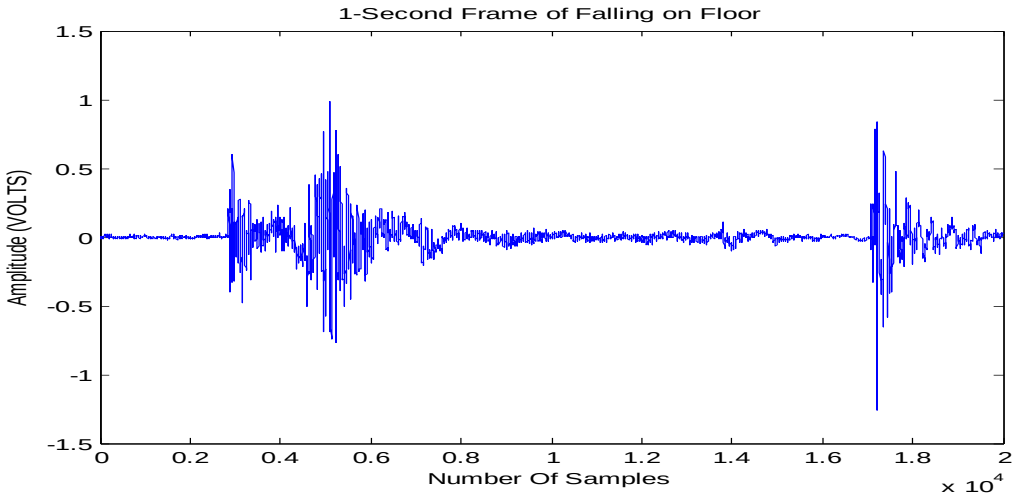

Figure 3: 1-second frame of falling on the floor

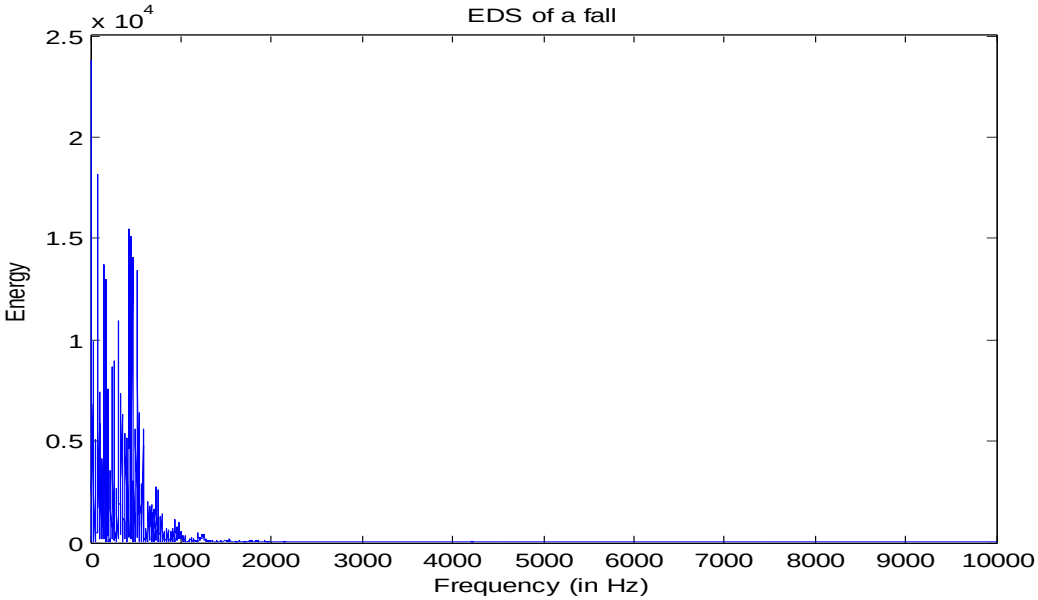

Figure 4: EDS of Above Fall Signal

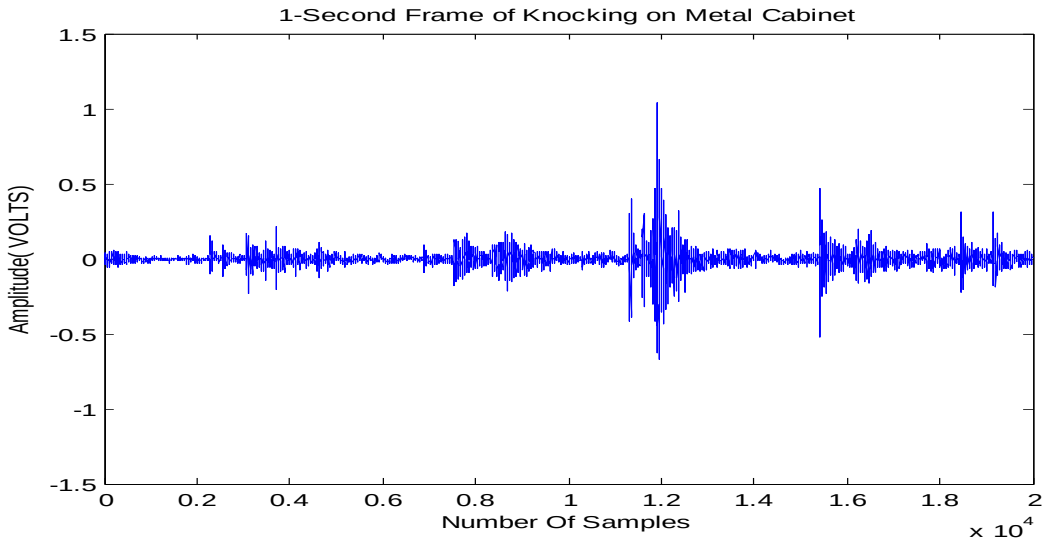

Figure 5: 1-second frame of knocking at the metal cabinet

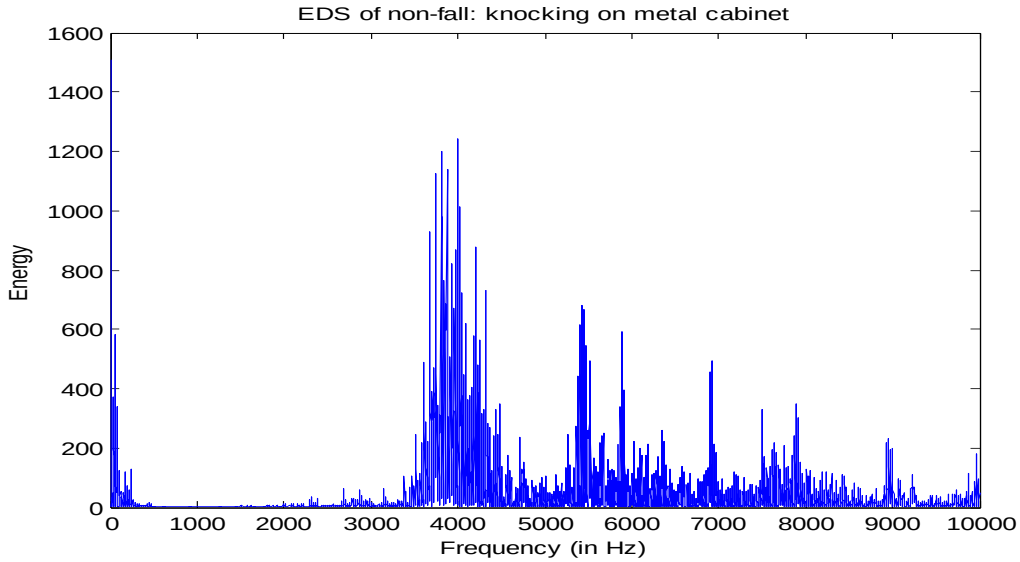

Figure 6: EDS of 1-second frame of knocking on metal cabinet

As can be seen above the two types of signals fall and Non-Fall both have similar amplitude in time domain (between -1.5 volts to +1.5 volts), but in frequency domain the fall has maximum magnitude in lower frequency range whereas non-fall has maximum energy in very high frequency range. The 4 figures above show EDS of a fall and non-fall signal.

2.2.2 Band Width and Centroid Frequency (BW&CF)

Let $S_i(\omega)$ represents the EDS of the i^{th} frame in terms of radial frequency $\omega = 2\pi f$, Thus the frequency centroid $C(i)$ and bandwidth $B(i)$ of this frame are defined as

$$C(i) = \frac{\int_0^\pi \omega |S_i(\omega)|^2 d\omega}{\int_0^\pi |S_i(\omega)|^2 d\omega} \quad (13)$$

$$B(i) = \sqrt{\frac{\int_0^\pi (\omega - C(i))^2 |S_i(\omega)|^2) d\omega}{\int_0^\pi |S_i(\omega)|^2 d\omega}} \quad (14)$$

$C(i)$ is the centroid frequency and $B(i)$ is the average bandwidth associated with $C(i)$. We calculate these two features for each training frame (We have 30 falls and 60 false alarm training frames). Then we plot a 2-Dimensional feature space in which X axis represents the Centroid Frequency and Y axis represents the Band Width. The red star points indicate the non falls and the blue circle ones indicate the falls.

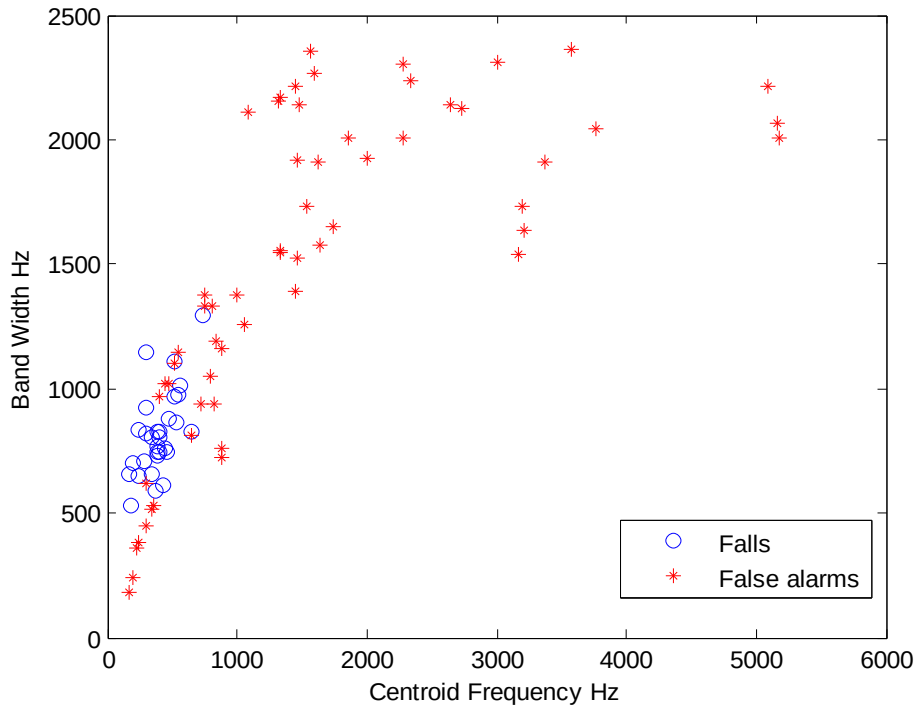

Figure 7: CF & BW of falls and non-falls

The figure shows this is a class classification problem which could be solved by different kinds of classification methods proposed in the field of pattern recognition.

2.2.3 Energy Ratio of Sub-bands (ERSB)

Division of the whole frequency band into several sub-bands (Liu et al. 1998) is a good idea for extracting features from different kind of acoustic signals. We use ratios of the

Power or Energy in different subbands to the total Power or Energy as subband energy ratios. It is defined as:

$$ERSB_k(i) = \frac{\int_{kl}^{kh} |S_i(\omega)|^2}{\int_0^{\pi} |S_i(\omega)|^2} \qquad (15)$$

kh and *kl* denote the high frequency bound and low frequency bound of *kth* subband in the *ith* frame. Through the observation of all the frequency spectrum of falls and false alarms, we carefully divide the whole frequency band into 3 sub-bands which could be distinguished from each other very well. Specifically, the frequency ranges of the three sub-bands are ERSB1 (0-600Hz), ERSB2(601-2206Hz), ERSB3(2206-20000Hz). We plot the three sub-bands features for each of the training frames (30 falls and 60 non-falls) in one figure.

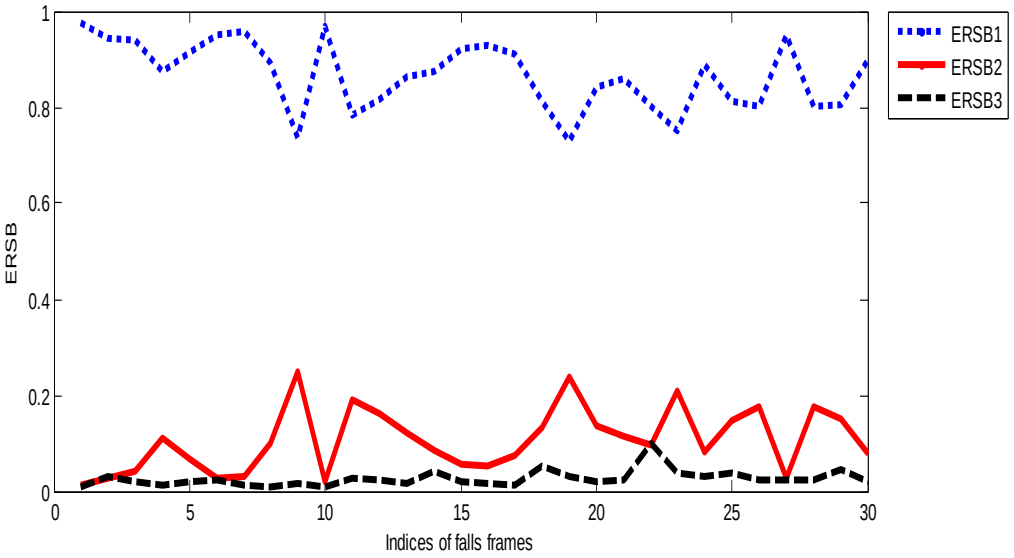

Figure 8(a): ERSB1-3 of falls

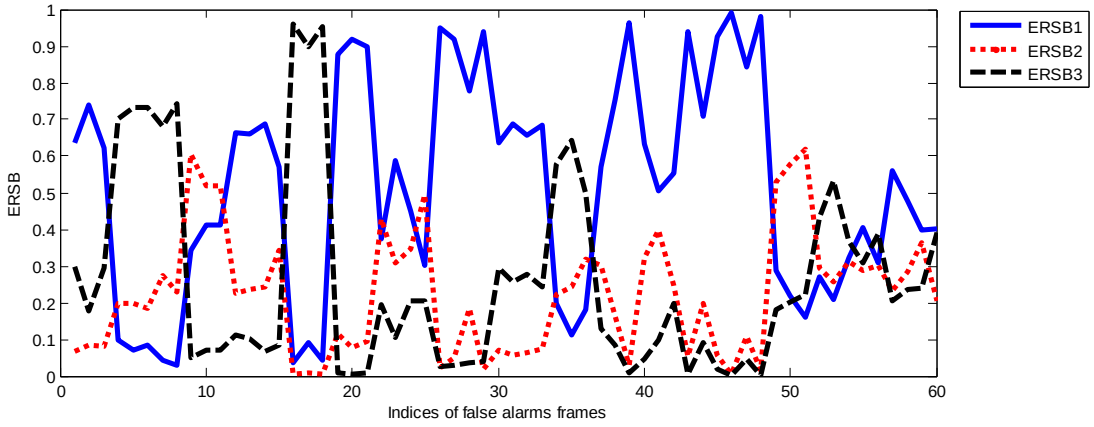

Figure 8(b) ERSB1-3 of non-falls

Obviously, the distribution of the ERSB in fall samples is more stable. However, the ERSB in false alarm samples is disorderedly, irregularly distributed among the frame indices because of abundant spectrums of sound events generated in the nature. For this reason we expect that doing one class classification may not give good results as it might be difficult to obtain a boundary around our non-falls.

2.3 Motivation for Using One Class Classifier

One class classification problem (Tax. 2004) differs from conventional classifier in one essential aspect. In one-class classification it is assumed that only information of one of the classes, the target class, is available. This means that just example objects of the target class can be used and that no information about the other class of outlier objects is present. The boundary between the two classes has to be estimated from data of only the normal, genuine class. The task is to define a boundary around the target class, such that it accepts as much of the target objects as possible, while it minimizes the chance of accepting outlier objects. In the literature a large number of different terms have been used for this problem. The term one-class classification originates from (Moya et

al.1993). The different terms such as fault detection, anomaly detection, novelty detection and outlier detection originate from the different applications to which one class classification can be applied. The applications are as follows:

The first application for one class classification (also called data description as it forms the boundary around the whole available data) is outlier detection, to detect uncharacteristic objects from a dataset; examples which do not resemble the bulk of the dataset in some way. These outliers in the data can be caused by errors in the measurement of feature values, resulting in an exceptionally large or small feature value in comparison with other training objects. In general, trained classifiers only provide reliable estimates for input objects resembling the training set. Extrapolations to unknown and remote regions in feature space are very uncertain (Roberts et al. 1996). Neural networks, for instance, can be trained to estimate posterior probabilities (Bishop 1995, Richard et al 1991, Ripley 1996) and tend to give high confidence outputs for objects which are remote from the training set. In these cases outlier detection should first be used to detect and reject outliers to avoid unfounded confident classifications.

Secondly, data description can be used for a classification problem where one of the classes is sampled very well, while the other class is severely under sampled. The measurements on the under sampled class might be very expensive or difficult to obtain. For instance, in a machine monitoring system where the current condition of a machine is examined, an alarm is raised when the machine shows a problem. Measurements on the normal working conditions of a machine are very cheap and easy to obtain. On the other hand, measurements of outliers would require the destruction of the machine in all possible ways. It is very expensive, if not impossible, to generate all faulty situations

(Japkowicz et al. 1995). Only a method trained on just the target data can solve the monitoring problem.

Finally, the last possible use of the outlier detection is the comparison of two data sets. Assume that a classifier has been trained (in a long and difficult optimization process) on some (possibly expensive) data. When a resembling problem has to be solved, the new data set can be compared with the old training set. In case of non-comparable data, we would know that the training of a new classifier is needed.

2.3.1 Using One Class Classifier for Fall Detection

As explained above, the second application of outlier detection is in the classification problem where one of the classes is sampled very well but it is very hard and expensive, if not impossible, to obtain the data of the second class. One of the major difficulties inherent in the data (as in many medical diagnostic applications) is this highly skewed class distribution. The problem of imbalanced datasets is particularly crucial in applications where the goal is to maximize recognition of the minority class. The problem of fall detection is also a similar problem. It is quite easy to obtain samples of everyday life sounds like door banging, keys falling, telephone ringing and almost a million other sounds but we **cannot** make an elderly person fall to collect the sound samples as this is the very first thing we want to avoid. Besides the everyday life sounds are quite similar hence we can capture these sounds in laboratory setting and apply it to practical applications, whereas sounds of falling person are quite different and it might be very difficult to design a classifier from the sound of a person (a stunt actor) falling in laboratory setting and apply the classifier to an actual case of an elderly person falling in

his apartment. The idea of our research is to use only these everyday life sound samples and design a decision boundary around them so that a sound of distress situation fall outside this boundary and can be easily detected as an outlier. Issue of class imbalance has been actively investigated and remains largely open. It is handled in a number of ways (Cohen et al. 2008), including**:** over sampling the minority class, building cost-sensitive classifiers that assign higher cost to misclassifications of the minority class, stratified sampling on the training instances to balance the class distribution and rule-based methods that attempt to learn high confidence rules for the minority class (Tarassenko et al. 1995). In general, in this thesis we investigate two types of methods of doing classification using one class classifier. These are as follows:

i) Density Method ii) Boundary Method:

In the next chapter we discuss the method of one class classification in detail along with the corresponding two class algorithm.

2.4 One class and Two Class Classification Algorithms

2.4.1 Density Based Methods

The most straightforward method to obtain a one-class classifier is to estimate the density of the training data and to set a threshold on this density. In (Bishop 1993), Bishop used the following approach: denote target by C_1 and novelty or outlier by C_2. The corresponding prior probabilities are $P(C_1)$ and $P(C_2)$ and the probability density functions are $p(x|C_1)$ and $p(x|C_1)$. According to Bayes theorem, $x \in C1\ if\ P\ (C1|x) = p(x|C1) * P(C1) > P(C2|x) = p(x|C2)\ P(C2)$. Deciding whether a new example x is abnormal or novel or outlier depends on the comparison between $p(x|C1)$ and $p(x|C2) * P(C2)/\ P(C1)$, where the latter is

equivalent to a threshold on our estimated density $p(x)$. Several distributions can be assumed, such as a Gaussian or a Poisson distribution, and numerous tests, called discordancy tests, are then available to test new objects. In this thesis we will consider the Parzen density Model.

2.4.1.1 Parzen Density Model for Two Class

Parzen density estimation is a Non-Parametric way (also called Parzen window) of doing pattern classification. This method is derived from Parzen window estimation where an *l*-dimensional space is divided into hypercubes with length of side $\boldsymbol{h}$ and volume h^l. Let $x_i\ i = 1,2, \ldots . N$ be the available feature vectors. Define the function $\varphi(x)$ so that (Theodoridis et al. 2003)

$$\varphi(x_i) = \begin{cases} 1, & for\ \ |x_{ij}| \leq 1/2 \\ 0, & otherwise \end{cases} \qquad (16)$$

where $|x_{ij}| = |x_{ij} - 0|\ and\ j = 1,2 \ldots . l$ are the components of x_i (actually the dimensions of a particular feature vector). In words we can say that the value of function $\varphi(x_i)$ will be 1 if the distance of feature vector from origin is less than ½ i.e. the vector lies within the hypercube of unit (h=1) dimension centered at origin. This idea is the extended for the estimation of density at particular feature vector $\boldsymbol{x}$**.** The density for the feature vector $\boldsymbol{x}$ is given by

$$\hat{p}(\boldsymbol{x}) = \frac{1}{V}\left(\frac{1}{N}\sum_{i=1}^{N} \ \varphi\left(\frac{\boldsymbol{x_i} - \boldsymbol{x}}{h}\right)\right) \qquad (17)$$

Here $V = h^l$ is the volume of hyper-sphere around the point where density has to be estimated. The major drawback of above equation is that a continuous density estimate has to be made using a discontinuous step function. The function φ is therefore replaced by a Gaussian Kernel function described below:

$$\exp\left(-\frac{||\boldsymbol{x_i}-\boldsymbol{x}||}{2h^2}\right) \qquad (18)$$

Here, $||\boldsymbol{x_i}-\boldsymbol{x}||$ is the distance measure and h is the width parameter which can be varied or kept same for different classes.

In case of Parzen density method when data from both the classes is available posterior probability is estimated for both the classes for a particular test object and the object is assigned to the class for which it has higher probability. The equation for posterior probability estimation with a Gaussian Kernel is given by:

$$p_p(\boldsymbol{x}, \boldsymbol{x_i}, hI) = \frac{1}{(h)^l \mathrm{N_k}} \sum_{\mathrm{i}=1}^{\mathrm{N_k}} \exp\left\{-\frac{||\boldsymbol{x}-\boldsymbol{x_i}||^2}{(2h^2)}\right\} \qquad (19)$$

The equal width h in each feature direction means that the Parzen estimator assumes equally weighted features and it will, therefore, be sensitive to the scaling of the feature values of the data, especially for lower sample sizes. Here h is optimized using maximum likelihood solution (equation 27). Volume $V = h^l$. Now in case of two class classifier, classification rule can be summarized as follows:

Assign test object $\boldsymbol{x}$ to $\omega_1(\omega_2)$ if: $$\frac{\frac{1}{(h)^l \mathrm{N}_1} \sum_{\mathrm{i}=1}^{\mathrm{N}_1} \exp\left\{-\frac{||\boldsymbol{x}-\boldsymbol{x_i}||^2}{(2h^2)}\right\}}{\frac{1}{(h)^l \mathrm{N}_2} \sum_{\mathrm{i}=1}^{\mathrm{N}_2} \exp\left\{-\frac{||\boldsymbol{x}-\boldsymbol{x_i}||^2}{(2h^2)}\right\}} > (<) \frac{P(\omega_1)\lambda_{21}-\lambda_{11}}{P(\omega_2)\lambda_{12}-\lambda_{22}} \qquad (20)$$

In our project of fall detection, a test object is MFC coefficients of a particular test sound frame and is given by equation (3) is a matrix of size 6X164 and the two classes for which posterior probability have to be estimated are fall and non-Fall. The training objects are also MFC coefficients of training sound frames which have the same dimension test object. Now, to compute Gaussian *Kernel* we have to compute the distance between a test object and training object. For distance measure we use the Euclidean distance. The distance between any pair of two vectors is defined by the 2-D Euclidean distance as follows:

$$dist_{p-q} = |\vec{p} - \vec{q}| = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \quad (21)$$

$$\text{Where } \vec{p} = (p_x, p_y) \; and \; \vec{q} = (q_x, q_y) \quad (22)$$

In our case the distance has to be calculated between matrices and not vectors. We have MXN dimensional MFCC feature matrix C_{MXN} where M is the number of coefficients and N is the number of sub-frames in one frame as explained in previous section. Experimentally, we found that higher the number of coefficients, better the discrimination between the two types of signals (fall and non-fall). But due to memory constraints we find a best match (optimum point) between the number of coefficients used and the time (memory) required for calculation so that for minimum possible memory usage we can get good discrimination (*decided by highest area under the ROC curve*). Hence the distance between two objects in our case is given as follows:

$$dist_{p-q} = |p_{MXN} - q_{MXN}| = \sum_{i=1}^{M} \sum_{j=1}^{N} (abs(p_{ij} - q_{ij})) \quad (23)$$

$$where, \; p_{MXN} = \begin{bmatrix} p_{11} & \cdots & p_{1N} \\ \vdots & \ddots & \vdots \\ p_{M1} & \cdots & p_{MN} \end{bmatrix}, \quad q_{MXN} = \begin{bmatrix} q_{11} & \cdots & q_{1N} \\ \vdots & \ddots & \vdots \\ q_{M1} & \cdots & q_{MN} \end{bmatrix}$$

Here *abs* represents the absolute value of the term in parenthesis following that. p_{MXN} is a feature object of the test sound frame set and q_{MXN} is the training feature set including features of fall and non-falls. Now that we have defined the distance measure, we can give a little modified version of equation (20) (but in essence the same).

$$\text{Assign test object } \boldsymbol{x} \text{ to } fall(no\ fall) \text{ if: } \quad \frac{\frac{1}{N_1}\sum_{i=1}^{N_1} \exp\{-\frac{||\boldsymbol{x}-\boldsymbol{x_i}||^2}{(2h^2)}\}}{\frac{1}{N_2}\sum_{i=1}^{N_2} \exp\{-\frac{||\boldsymbol{x}-\boldsymbol{x_i}||^2}{(2h^2)}\}} > (<) thr \quad (24)$$

In equation (24) the volume term has been dropped because h is same for both the classes. N_1 is the number of objects in fall and N_2, the number of objects in No-Fall. The term on right hand side of equation (20) has been replaced by thr in equation 24, which is a threshold value (0,1]. In equation (20) we have $\frac{P(\omega_1)\lambda_{21}-\lambda_{11}}{P(\omega_2)\lambda_{12}-\lambda_{22}}$. Prior for both the classes are assumed to be the same and also the cost for any misclassification is same. Hence our thr takes an initial value of 1. Now since we would like to estimate the whole Receiver Operator Characteristic curve we need to shift the boundary between the two classes, so that we can observe that for what particular false alarm ratio we get a fall detection ratio. For getting the whole ROC curve therefore we will vary this threshold thr.

In next sub-section classification procedure is explained when training objects from only one class are used to do the classification.

2.4.1.2 Parzen Density Model for one Class

In case of two class Parzen density estimate we estimated the posterior probability for both the classes i.e. fall and no-fall. The test object is assigned to the class with higher posterior probability. A threshold is then set on the ratio of the posterior probabilities of both the classes and a ROC curve is obtained. But as explained in section 2.2 sound samples of one of the classes i.e. fall is difficult to obtain as we cannot ask an elderly person to fall so that we can obtain the training data for our classifier. Hence, in this section we explain the one class approach to do the classification and detect falls. Here we estimate the density of one of the classes for which data is easily obtained. Classification of the test object is done as target, if it belongs to the class for which we estimated the density or outlier if it does not belong to this class. The Parzen window

(Cohen et al. 2008) is a simple kernel based estimator for estimating density function. The main idea of Parzen density estimation is to estimate density function using Gaussian kernels for one class and determine whether a test object will be target or outlier based on this density. Now if we have 'N_k' sample vectors of well sampled class, the density (or posterior probability) is estimated by the equation given below:

$$p_p(\boldsymbol{x}, \boldsymbol{x_i}, h) = \frac{1}{(h)^l N_k} \sum_{i=1}^{N_k} \exp\{-\frac{||\boldsymbol{x}-\boldsymbol{x_i}||^2}{(2h^2)}\} \qquad (25)$$

'h' is the smoothing parameter or band-with and l is the dimensionality of the feature space. The density estimator $p_p(x)$ obtained from the training set give us a quantitative measure of the degree of novelty for each new example. This is used to reject examples where the estimate $p_p(\boldsymbol{x}, \boldsymbol{x_i}, h) < \rho$ for some threshold ρ, effectively generating a new class of "novel" data. Thus any point where the posterior probability $p_p(\boldsymbol{x}, \boldsymbol{x_i}, h)$ is below some threshold is considered to be novel or outlier. Now in our case the class for which density is estimated is Non-Fall class and density is estimated using equation (25). The term $\exp\{-\frac{||x-\boldsymbol{x_i}||^2}{(2h^2)}\}$ is calculated in the same way as explained in Parzen density model for two classes. The distance between two objects is actually the distance between two matrices as explained in equation (23). The posterior probability is then estimated and arranged in increasing order for the whole Non-Fall class. The classification rule applied is as follows:

Assign test object $\boldsymbol{x}$ to $fall$ if $p_p(\boldsymbol{x}, \boldsymbol{x_i}, h) = \frac{1}{(h)^l N_k} \sum_{i=1}^{N_k} \exp\{-\frac{||\boldsymbol{x}-\boldsymbol{x_i}||^2}{(2h^2)}\} < \rho$ (26)

Where ρ is varied to compute the whole ROC curve. N_k is the number of training samples of Non-Fall and $\boldsymbol{x_i}$, $i = 1,2 \ldots . N_k$. are the training matrices.

Estimation of h (Width Parameter): In practice, we only have a finite number of samples hence, a compromise between *h* and N must be made. The choice of suitable values for *h* is crucial and several approaches have been proposed in the literature, for examples, see (theodorodis et al. 2007). A straightforward way is to start with an initial estimate of *h* and then modify it iteratively to minimize the resulting misclassification error. The latter can be estimated by appropriate manipulation of the training set. For example, the set can be split into two subsets, one for training and one for testing. For our project estimation of width parameter was initialized using leave-one-out maximization of "pseudo likelihood" (Cohen et al. 2008) as follows:

$$h = \underbrace{argmax}_{\hat{h}} \{\frac{1}{N}\sum_{i=1}^{N} \log(f(\boldsymbol{x_i}))\} \text{ where } f(x_i) = \frac{1}{(N-1)h}\sum_{i=1}^{N} \exp(-\frac{||\boldsymbol{x}-\boldsymbol{x_i}||}{(2h)}) \quad (27)$$

The choice of *h* was made for one class and kept the same for two class parzen density estimation as well. The width parameter was then iteratively varied to get the best classification results (details in chapter 4) as the MFCC matrices have similar values for both the classes hence we don't need to change the value of width parameter for each class.

2.4.2 Boundary Based Methods

Vapnik argued in (Vapnik 1998) that when just a limited amount of data is available, one should avoid solving a more general problem (like estimating probability density) as an intermediate step to solve the original problem of doing the classification. To solve this more general problem more data might be required than for the original problem. In our case this means that estimating a complete data density for a one-class classifier might also be too demanding when only the data boundary is required. In the boundary

methods, therefore, only a closed boundary around the target set is optimized. In this project we consider the KNN-dd and the SVDD.

2.4.2.1 Support Vector Classifier or Support Vector Machine

In this section we will first explore classification between two classes using support vector machine followed by Support vector data description i.e. classification using the data from only one class or in short one class classification.

a) Support Vector Machine - The capability of any classifier to deal with the unknown data is very important. This means that a classifier designed using training data set should be able to operate satisfactorily with data outside this training set. Suppose we have a training data set consisting of a set of feature vectors $\boldsymbol{x_i},\ i = 1,2,\ldots..N$. These belong to either of the two classes $\omega_1 or\ \omega_2$. If these vectors are linearly separable then we can design a hyper-plane

$$g(\boldsymbol{x}) = \boldsymbol{w^T x} + w_0 \qquad (28)$$

That classifies all these feature vectors correctly. Such a hyper-plane is not unique.

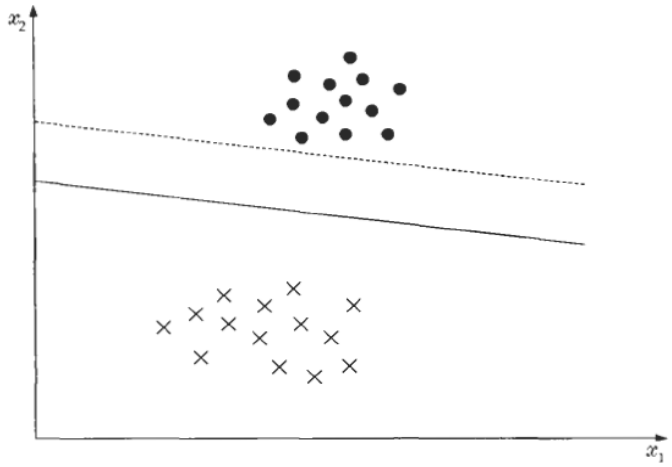

Figure 9: optimum classifier

As can be seen in the above figure, both the classifiers get the job done without any classification errors. Still, it is easy for any engineer to pick the better classifier. The one in the bold line is obviously a better choice as it leaves "*room*" or margin for the data outside the training set to be classified without any errors by the classifier (Theodoridis et al. 2003). Let us now quantify the term "room" that a hyper-plane leaves from both classes. Every hyper-plane is characterized by its direction (determined by $\boldsymbol{w}$ (equation (28)) and its exact position in space (determined by $w_0$equation (28))**.** Since we want to give no preference to either of the classes, then it is reasonable for each direction to select that hyper-plane which has the same distance from the respective nearest points in ω_1 and ω_2. This is illustrated in figure 2. Our goal is to search for the direction that gives the maximum possible margin. Now, the distance of a point (or a vector) from a hyper-plane is given by:

$$z = \frac{g(\boldsymbol{x})}{||\boldsymbol{w}||} \qquad (29)$$

Scaling $\boldsymbol{w}$ and w_0 so that the value of $g(\boldsymbol{x})$ at nearest points in ω_1 and ω_2 (circled in figure 9) is equal to 1 for $\omega_1$and equal to -1 for ω_2. This is equivalent to:

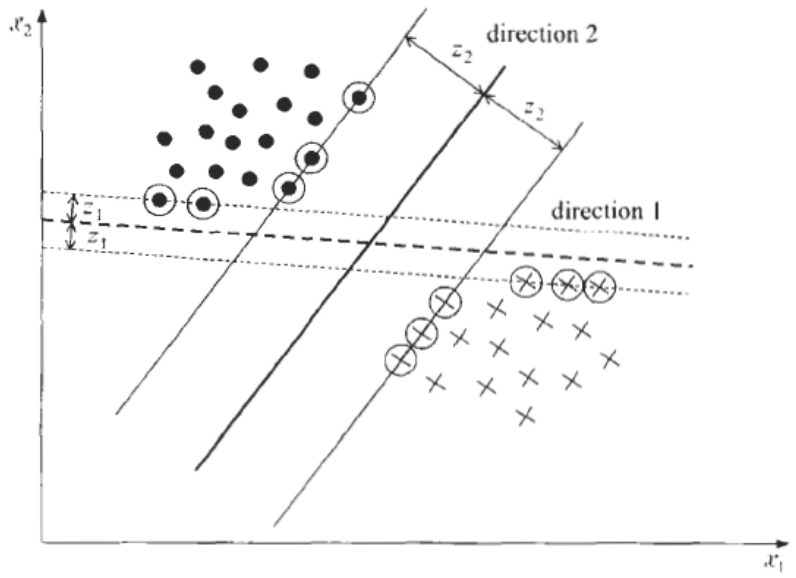

Figure 10: different possible classifiers with margins indicated

1. Having a margin of:

$$\frac{1}{||\boldsymbol{w}||} + \frac{1}{||\boldsymbol{w}||} = \frac{2}{||\boldsymbol{w}||} \qquad (30)$$

2. Requiring that:

$$\boldsymbol{w^T x} + w_0 \geq 1 \quad \forall\ x\ \epsilon\ \omega_1 \quad \text{and}$$

$$\boldsymbol{w^T x} + w_0 \leq -1 \quad \forall\ x\ \epsilon\ \omega_2 \qquad (31)$$

For each $\boldsymbol{x}$, we denote the corresponding class indicator by y_i (+ 1 for ω_1, - 1 for ω_2). Our task can now be summarized as: Compute the parameters $\boldsymbol{w}$ and w_0 of the hyper-plane so as to:

$$\text{Minimize: } J(\boldsymbol{w}) = \frac{1}{2}\ ||\boldsymbol{w}||^2$$

$$\text{Subject to: } \ y_i\left(\boldsymbol{w^T x_i} + w_0\right) \geq 1 \quad i = 1,2 \ldots \ldots N \qquad (32)$$

Minimizing *J (**w**)* makes the margin maximum. The above task is non-linear (quadratic) optimization task with linear inequality constraints. This problem falls in the category of *convex programming* family of problems. Since the cost function is convex and the set of

constraints are linear and define a convex set of feasible solutions, such problems can be solved by considering the *Langrangian duality*.

After applying langragian duality we end up with the use a bit of algebra with equivalent optimization task:

Maximize λ: $\left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j \boldsymbol{x_i^T x_j}\right)$

Subject to: $\sum_{i=1}^{N} \lambda_i \, y_i = 0$ (33)

Once the optimal *Langrangian* multipliers (λ) have been obtained by maximizing (33), optimal hyper-plane can be obtained using the two conditions given below (Theodoridis et al. 2007):

$$w = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x_i} \qquad (34)$$

And $\lambda_i[y_i(\boldsymbol{w^T x_i} + w_o) - 1] = 0 \quad i = 1,2,\ldots..N$ (35)

The value of w_o is actually estimated by taking the average of all the equations represented by equation (35).

The case discussed above is a simpler one in the sense that classes were totally separable by a linear classifier, next we move on to the case where classes are not totally separable. As can be seen in the figure given below, the circled points (vectors) are not correctly classified hence the two classes are not separable. As we did before we again make a margin of $\frac{1}{||\boldsymbol{w}||} + \frac{1}{||\boldsymbol{w}||} = \frac{2}{||\boldsymbol{w}||}$. The two parallel hyper-planes $\boldsymbol{w^T x_i} + w_0 = \pm 1$ mark the boundaries of the margin but they no longer pass through the nearest points in both the classes. This creates three types of vectors in the dataset. These are

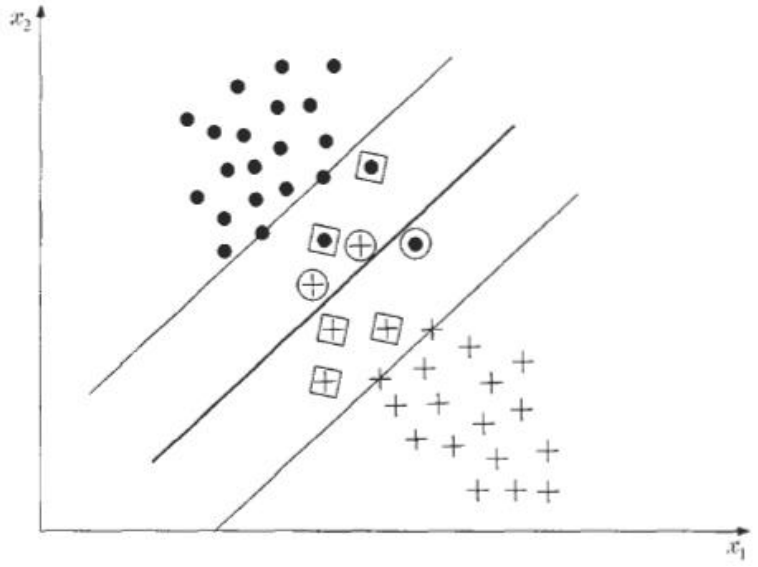

Figure 11: two classes which are not linearly separable

1. Vectors that lie outside the margin or band between the two parallel hyper-planes and are correctly classified.
2. Vectors falling inside the band which are correctly classified, these vectors are marked by squares around them in the above figure. These vectors satisfy the following inequality:

$$0 \leq y_i(\boldsymbol{w^T x_i} + w_0) < 1$$

Here all the terms are same as in the case of totally separable classes.

3. Vectors that lie in the band but are incorrectly classified which are shown by circles around them in the above figure. They follow the inequality

$$y_i(\boldsymbol{w^T x_i} + w_0) < 0$$

All these inequality can be summarized by introducing a new set of variablesξ_i:

$$y_i(\boldsymbol{w^T x_i} + w_0) \geq 1 - \xi_i$$

The first category of data correspond to $\xi_i = 0$, the second to $0 < \xi_i \leq 1$, and the third to $\xi_i > 1$. The variables ξ_i are known as slack variables. The goal now, is to make the

margin as large as possible but at the same time to keep the number of points with $\xi_i > 0$ as small as possible. In mathematical terms, this gives us the following optimization task:

Minimize: $J(\boldsymbol{w}, w_0, \xi_i) = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{N}\xi_i$

Subject to: $y_i(\boldsymbol{w^T x_i} + w_0) \geq 1 - \xi_i \qquad \xi_i \geq 0$ and i = 1,2 ….. N (36)

The parameter C is a positive constant that controls the relative influence of competing terms. The problem is again a convex programming problem and we can proceed as before using Langrangian and KKT conditions followed by little algebra to obtain the following similar expression as before when classes were totally separable:

Maximize λ: $\left(\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_i\sum_j \lambda_i\lambda_j y_i y_j \boldsymbol{x_i^T x_j}\right)$

Subject to: $for\ 0 \leq \lambda_i \leq C\ i = 1,2 \ldots . N \qquad \sum_{i=1}^{N}\lambda_i y_i = 0$ (37)

Again we can use equation 34 and equation 35 to obtain $\boldsymbol{w}$ and w_0 and therefore the optimal hyper-plane. The only difference from the case of totally linearly separable class is the introduction of C which bounds the Lagrangian multipliers λ_i. The linearly separable case corresponds to $\lambda_i \to \infty$. The presence of slack variables ξ_i is only felt through introduction of C.

The two variations of support vector classifier discussed above are next applied to our fall detection experiments. The first thing to be selected is our objects. The main advantages of support vector over density method is

1. Only objects at the boundary are required to do the classification and therefore we do not require lot of training objects.

2. The dimensionality of vectors does not come into play in any equation, moreover since we deal with only inner products of two objects as can be seen in equation (37), hence we

can keep the dimension of our objects very high without increasing much computational costs.

Our objects, as before are MFCC matrices of size MXN (6X194). For support vector though we can convert it into vectors. To do that each of the 6 coefficients of each sub-frame are arranged one after the other to make every object as a vector of dimensions M multiplied by N where M is number of coefficients and N is the number of sub-frames in each of our sound signal frame. As discussed in Parzen density estimation method, we have 7 coefficients and 164 sub-frames but since the first coefficient gives only energy we end up with a 6X164 Matrix. Hence for support vector the length or dimension of each object or vector would be 6*164=984. The choice of C does not make much of a difference in the classification results. Finally classification rule can be summarized as follows:

Assign $\boldsymbol{x}$ in $fall\ (non\ fall)\ if\ g(\boldsymbol{x}) = (\sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x_i})^T \boldsymbol{x} + w_0 > (<) thr$ (38)

The actual term on right hand side of equation (38) is 0 but that will give us only one classification result hence that term has been replaced by a threshold $\in[0,1]$ to obtain the whole ROC curve.

The above two support vector machines were for linear classifiers. Hence once $\boldsymbol{w}$ and w_0 were computed the decision was based on whether $g(\boldsymbol{x})$ is above a certain threshold to be in the class of fall or non-fall.

We now extend the support vector classifiers to non-linear classification task. Suppose we do the mapping of input feature vector from a *k* dimension space to *l* dimension space as follows:

$$\boldsymbol{x} \in R^k \rightarrow \boldsymbol{y} \in R^l$$

Now, let us suppose that it is possible to design a linear classifier in this new *l*-dimension space, then all the inner products between input vectors, as explained in previous sections would be replaced by new inner products in *l*-dimension space. This also increases the complexity as *l* is usually much higher than *k* which is necessary to make the classes linearly separable.

This can be understood if we look at a very simple example. Let *x* be a two dimension feature vector mapped as follows:

$$\boldsymbol{x} \in R^2 \rightarrow \varphi(\boldsymbol{x}) = \boldsymbol{y} = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

Where $\varphi(\boldsymbol{x})$ indicates mapping.

By simple algebra we can prove that

$$\boldsymbol{y_i^T y_j} = (\varphi(\boldsymbol{x_i}))^T * \varphi(\boldsymbol{x_j}) = (\boldsymbol{x_i^T x_j})^2$$

In essence, the inner product in new higher dimension space has been expressed as a function of inner product in lower (or original) dimension space. This function has a special significance and is called a *Kernel*. The function can be expressed as ***K(x, z)*** and must satisfy the following conditions to be an inner product:

$$\text{(i)} \quad \int \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{z})\boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{z})\boldsymbol{dxdz} \geq 0 \qquad (39)$$

For any *g(x)*, $\boldsymbol{x} \in R^l$ such that

$$\text{(ii)} \quad \int \boldsymbol{g}(\boldsymbol{x})^{\boldsymbol{2}}\boldsymbol{dx} < +\infty \qquad (40)$$

These two conditions are derived from Mercer's Theorem. According to which opposite is also true that is if there exists ***K(x, z)*** such that it satisfies the two above conditions, then this function can define an inner product in a mapped space. Mercer's theorem,

though, does not specify on how to find this space which means we cannot find the mapping $\varphi(.)$. Furthermore it is difficult to find the dimensionality of new mapped space, which can be infinite (Theodoridis et al. 2007). Typical examples of Kernel function used in Pattern Recognition are:

Polynomials:

$$K(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{z} + 1)^q, \quad q > 0 \qquad (41)$$

Radial Basis Function or Gaussian Function

$$K(\boldsymbol{x}, \boldsymbol{z}) = \exp(-\frac{||\boldsymbol{x}-\boldsymbol{z}||}{\sigma^2}) \qquad (42)$$

Hyperbolic Tangent Function

$$K(\boldsymbol{x}, \boldsymbol{z}) = \tanh(\beta \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{z} + \gamma) \qquad (43)$$

For appropriate values of β and γ so that Mercer's conditions are satisfied above. One possibility is $\gamma = 1$ and $\beta = 2$. Once an optimal Kernel is specified we go through the same optimization procedure as with linear classifiers to get $\boldsymbol{w}$ and w_0 and get the resulting classifier in terms of langrangian as follows:

Optimization Task:

$$\text{Maximize } \lambda\text{: } \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j \boldsymbol{x_i^T x_j}\right)$$

$$\text{Subject to:} \qquad for\ 0 \le \lambda_i \le C\ i = 1,2 \ldots . N \qquad \sum_{i=1}^{N} \lambda_i\, y_i = 0 \qquad (44)$$

Classification:

$$\text{Assign } \boldsymbol{x} \text{ in } fall(no\ fall)\ if\ g(x) = \sum_{i=1}^{N} \lambda_i y_i K(\boldsymbol{x_i}, \boldsymbol{x}) + w_0 > (<) thr \qquad (45)$$

The above equation is nothing else but a general linear classifier equation (38) replaced by a kernel function in place of inner product. This gives us a non linear optimum classifier.

2.4.2.2 Support Vector Data Description

As explained in the previous section a support vector machine finds the optimum hyper-plane for classification. The classification task where it is not possible to define an optimum hyper-plane, we define a Kernel function which replaces the inner product of input vector with this function. This *Kernel* function represents the inner product in a new space in which classification can be done and we can define an optimum hyper-plane. Support vector data description i.e. support vector for one class classification is derived from Support Vector machine as it also uses similar optimization procedure. In case of one class classification using support vector method we define a boundary around our target (the class for which data is easily available or non-fall in our project) dataset and try to minimize chances of accepting outliers. The model is written in a form comparable to the support vector classifier (SVC), and it is therefore called the support vector data description (SVDD). It offers the ability to map the data to a new, high dimensional feature space without much extra computational cost. By this mapping more flexible descriptions are obtained but we start with the most basic one which is comparable to the linearly separable case of support vector classifier.

Spherical Data Description: First, a structural error is defined by:

$$\varepsilon_{struct}(R, \boldsymbol{a}) = R^2 \qquad (46)$$

R is the radius of hyper-sphere and $\boldsymbol{a}$ is its center. The above structural error has to be minimized with the constraints

$$||\boldsymbol{x_i} - \boldsymbol{a}||^2 \leq R^2 \; for\; all\; i\; (all\; objects) \qquad (47)$$

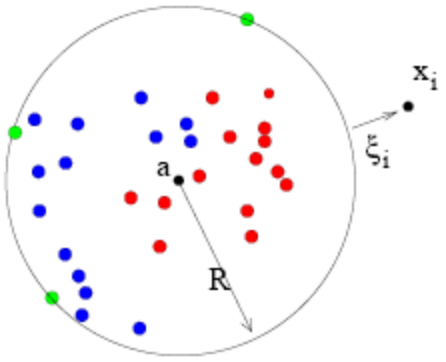

Figure 12: Hyper-sphere containing one class data

The minimization task above can be made more robust by allowing the possibility of outliers in training set. This means that distance of object $\boldsymbol{x_i}$ should not be strictly less than R^2 but larger distances should be penalized. For this purpose slack variables ξ_i are introduced. The minimization task now changes to

$$\text{Minimize:} \quad \varepsilon_{struct}(R, \boldsymbol{a}, \xi) = R^2 + C \sum_i \xi_i \quad \forall\, \xi_i \geq 0 \qquad (48)$$

With the constraint that (almost) all objects should be inside this sphere which means

$$||\boldsymbol{x_i} - \boldsymbol{a}||^2 \leq R^2 + \xi_i \;\; for\ all\ i\ (all\ objects) \qquad (49)$$

Vectors outside the description have $\xi_i > 0$ and $\xi_i = 0$ for objects inside the description or on boundary. The parameter C gives the tradeoff between the volume of the description and the errors. We proceed as before in solving this constraint problem by introducing Lagrange multipliers and constructing the lagrangian

$$L(R, \boldsymbol{a}, \xi, \alpha, \gamma) = R^2 + C \sum_i \xi_i - \sum_i \alpha_i R^2 + \xi_i - (\boldsymbol{x_i x_i} - 2\boldsymbol{a}.\boldsymbol{x_i} + \boldsymbol{a}.\boldsymbol{a})\} - \sum_i \gamma_i \xi_i \quad (50)$$

Here the Lagrangian multipliers$\alpha_i \geq 0$ and $\gamma_i \geq 0$. $\boldsymbol{x_i x_j}$ Stands for inner product between $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ i.e. in vector notation it means $\boldsymbol{x_i}^T \boldsymbol{x_j}$ **.** Note that for each object $\boldsymbol{x_i}$ a

corresponding α_i and γ_i is defined. L has to be minimized with respect to R, $\boldsymbol{a}$, ξ_i and maximized with respect to α and γ. Setting partial derivatives to zero give us

$$\frac{\partial L}{\partial R} = 0: \qquad \sum_{i=1}^{N} \alpha_i = 1$$

$$\frac{\partial L}{\partial \boldsymbol{a}} = 0: \qquad \boldsymbol{a} = \sum_i \alpha_i \boldsymbol{x_i} \text{(center of hyper-sphere)}$$

$$\frac{\partial L}{\partial \xi_i} = 0: \qquad \gamma_i = C - \alpha_i$$

From the last equation we get that $\gamma_i = C - \alpha_i$. Instead of the constraints that $\gamma_i \geq 0$ 0 and $\gamma_i = C - \alpha_i$ a new constraint on α_i can be introduced as:

$$0 \leq \alpha_i \leq C$$

This gives us the minimization task of error function L

$$L = \sum_i \alpha_i (\boldsymbol{x_i}^T \boldsymbol{x_i}) - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j (\boldsymbol{x_i}^T \boldsymbol{x_j}) \qquad (51)$$

With $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N} \alpha_i = 1$

In these constraints if we choose value of C greater than 1 then the second constraint will force all the objects to be accepted because for all the objects $\alpha_i < C$. Hence some objects can be left out during training by controlling the value of C. This is a quadratic programming problem which can be solved to obtain allα_i's. The value $\alpha_i = 0$ is for all objects satisfying the inequality in equation (49). For objects satisfying the equality $||\boldsymbol{x_i} - \boldsymbol{a}||^2 = R^2 + \xi_i \quad \alpha_i > 0$ and object lies on the boundary or outside the target set. When an object obtains $\alpha_i = C$, it is rejected as outlier. Hence, an object with $\alpha_i > 0$ can be on the boundary or outside the boundary (*note: this is further explored in chapter 4 Results*). The center of hyper-sphere is given as:

$$\boldsymbol{a} = \sum_i \alpha_i \boldsymbol{x_i} \qquad (52)$$

The objects or *vectors* with non zero lagrange's multiplier are the only objects needed for defining this center as rest of them will have zero weight and will therefore have no effect on obtaining the center of sphere. These few vectors with non-zero α_i are called *support vectors* hence, the name *Support Vector Data Description* is given to this technique. Now our next task is to classify our test vector or object $\boldsymbol{z}$. i.e. whether our test object belongs inside the hyper-sphere or outside. In simpler words, we need to test that whether our test object is part of target dataset (non-fall) or is it an outlier (fall)? To do that, we simply calculate the distance of test object from the center of the sphere. If this distance comes out to be more that the radius of hyper-sphere, we can say that the test objet is outside our target dataset or in other words it is an outlier. Mathematically, if **z** is our test object, then the distance of our test object from center of sphere can be given by, using equation (52) for the center of hyper-sphere:

$$\left|\left|\boldsymbol{z}-\boldsymbol{a}\right|\right| = (\boldsymbol{z}.\boldsymbol{z}) - 2\sum_i \alpha_i\,(\boldsymbol{z}.\boldsymbol{x_i}) + \sum_{i,j}\alpha_i\alpha_j\,(\boldsymbol{x_i}\boldsymbol{x_j}) \leq R^2 \qquad (53)$$

Here $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ represent support vectors only as these are the only objects with corresponding non-zero multipliers $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$. R^2 is the distance from center of hyper-sphere to one of the support vectors on the boundary. This can be given as

$$R^2 = (\boldsymbol{x_k}.\boldsymbol{x_k}) - 2\sum_i \alpha_i\,\boldsymbol{x_i}\boldsymbol{x_k} + \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j\,\boldsymbol{x_i}\boldsymbol{x_j} \qquad (54)$$

For any $x_k \in SV^{bnd}$ here again, equation (52) is used for center of hyper-sphere.

Finally, this leads us to the equation for one class classifier called support vector data description given as:

$$f_{SVDD}(z;\alpha,R) = I\left(\left|\left|\boldsymbol{z}-\boldsymbol{a}\right|\right|^2 \leq R^2\right) \qquad (55)$$

$$= I((\boldsymbol{z}.\boldsymbol{z}) - 2\sum_i \alpha_i\,(\boldsymbol{z}.\boldsymbol{x_i}) + \sum_{i,j}\alpha_i\alpha_j\,(\boldsymbol{x_i}\boldsymbol{x_j}) \leq R^2)$$

Where the Indicator function *I* is defined as

$$I(A) = \begin{cases} 1, & if\ A\ is\ true \\ 0, & otherwisex \end{cases} \quad (56)$$

Which is equivalent to saying that a test object $\boldsymbol{z}$ is assigned to non fall class if the distance of that object from the center of the sphere we obtained is less than radius otherwise it is called a fall.

Flexible Data Description: In two class classification using support vector machine section 2.3.2.1 the idea of kernel function was introduced. It was described that how if data was *not* linearly separable we could map our input data to a higher dimension space in which it was, in fact linearly separable. The kernel function is nothing but the inner product of two objects in this new higher dimension space. Similar to two class case we can define a flexible and less rigid data description for one class classifier as well. Here again using *Kernel* function we would define non-hyper-spherical boundary around our data. Assuming we are given a mapping for the input data as follows:

$$\boldsymbol{x}^* = \varphi(\boldsymbol{x})$$

We can apply this mapping to equation (51) to get the following equation:

$$L = \sum_i \alpha_i \varphi(\boldsymbol{x_i})\varphi(\boldsymbol{x_i}) - \sum_{ij} \alpha_i \alpha_j \varphi(\boldsymbol{x_i})\varphi(\boldsymbol{x_j}) \quad (57)$$

The above function is minimized with the constraints and we get the final indicator function:

$$f_{SVDD}(z; \alpha, R) = I(\varphi(\boldsymbol{z}).\varphi(\boldsymbol{z}) - 2\sum_i \alpha_i\, \varphi(\boldsymbol{z}).\varphi(\boldsymbol{x_i}) + \sum_{i,j} \alpha_i \alpha_j \varphi(\boldsymbol{x_i}).\varphi(\boldsymbol{x_j}) \leq R^2) \quad (58)$$

As can be seen, that above equation has only inner product in mapped space, which we can replace by a *Kernel* function.

$$\text{Kernel function}=K(\boldsymbol{x}, \boldsymbol{x}) = \varphi(\boldsymbol{x}) * \varphi(\boldsymbol{x})$$

Putting this Kernel function in equation (58)

$$f_{SVDD}(z;\alpha,R) = I(K(\boldsymbol{z},\boldsymbol{z}) - 2\sum_i \alpha_i K(\boldsymbol{z},\boldsymbol{x_i}) + \sum_{i,j} \alpha_i\alpha_j K(\boldsymbol{x_i},\boldsymbol{x_j}) \leq R^2) \quad (59)$$

In practice, we do not use mapping but the kernel function directly. Mapping can be obtained by kernel function in some cases but it is not possible to obtain kernel function implicitly in general. Concept of Kernel function was introduced by (Vapnik 1998). The advantage of the '*kernel trick*' is that the introduction of the kernels does not introduce much extra computational cost. The optimization problem remains identical in the number of free parameters. The only extra cost is in the computation of the kernel function $K(\boldsymbol{x_i},\boldsymbol{x_j})$. The most common kernel functions used are Polynomial (equation 41) and Gaussian (equation 42). In case of SVDD Gaussian Kernel should be used because by using the Gaussian kernel we can control the boundary around our target dataset as well as number of support vectors. The decision rule for one class can be summarized as follows:

Assign test sound frame feature vector $\boldsymbol{z}$ to no-fall (fall):

$$K(\boldsymbol{z},\boldsymbol{z}) - 2\sum_i \alpha_i K(\boldsymbol{z},\boldsymbol{x_i}) + \sum_{i,j} \alpha_i\alpha_j K(\boldsymbol{x_i},\boldsymbol{x_j}) \leq (\geq)\ R^2 + thr \quad (60)$$

In case of one class Parzen density estimation a threshold was put on density, here a threshold should be put on distance measure of radius R^2 to obtain the whole ROC curve. The threshold is actually a negative number so that we can successively reject more and more target (no fall) to see how many falls (outlier) we are able to detect hence giving us the whole ROC curve. The Gaussian Kernel used in Support Vector Data Description also uses same distance measure as Parzen density estimation method. The distance measure used for Support vector is described in equation (23). Another advantage of using Gaussian Kernel is that the equation (57) changes to maximization of a simpler function give as:

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j(\boldsymbol{x_i}^{\boldsymbol{T}}\boldsymbol{x_j})$$

And also the first term in equation (58) becomes 1. Hence we can re-write equation (58) after using Gaussian Kernel as follows:

$$1 - 2\sum_i \alpha_i \exp(-\frac{||\boldsymbol{x_i}-\boldsymbol{z}||}{\sigma^2}) + \sum_{i,j}\alpha_i\alpha_j \exp(-\frac{||\boldsymbol{x_i}-\boldsymbol{x_j}||}{\sigma^2}) \leq (\geq) R^2 + thr \quad (58.1)$$

2.4.2.3 K-Nearest Neighbors Two Class Classification

The Nearest Neighbor algorithm (Cover et al. 1967) is a very popular algorithm for doing the Pattern Recognition. Its popularity lies in its simplicity. In simplest terms a nearest neighbor algorithm assigns a sample vector or a test vector to the class of its nearest neighbor. Nearest neighbor can be found by any of the available similarity or dissimilarity measures. In this thesis we have exclusively used Distance measure or dissimilarity measure given by equation (27). K-nearest neighbor is an extension of nearest neighbor algorithm where we use K number of nearest neighbors. The test vector is assigned to the class represented by majority of K-Nearest neighbor vectors. Of course there can be tie among the classes if majority class is represented by equal number of vectors. For two class case this problem can be avoided by picking an odd K. The complete K-Nearest Neighbor algorithm is given below :

K-Nearest Neighbor:

Let $W = \{x_1, x_2, \ldots \ldots x_n\}$ be a set of labeled samples.

Begin:

1. Input test vector *y*. Set 1<= k<=n and initialize i=1

2. Do until (k-nearest neighbors are found)

3. Compute distance d (x_i, y)

If (i $\leq$ k) then include x_i in the set of K-nearest neighbors

Else if (x_i is closer to y than previous nearest neighbors)

Then delete farthest nearest neighbor in set and include x_i in the set of nearest neighbors.

End if

Increment i. End Do until

To classify we follow the following steps:

Determine majority class in set of K-nearest neighbors.

If (a tie exists) then compute sum of distances of nearest neighbors in classes.

If (no tie in sum) then classify y in class of min sum

Else classify y in class of last min found

End if

Else (no tie)

Classify y in majority class.

End if

End (of algorithm)

2.4.2.4 K-Nearest Neighbor Data Description

For one class classification K-Nearest Neighbor data description uses similar method but not identical. In this method we have only one class called the target class. It has to be determined whether a test object **z** belongs to this target class or not. The target class is represented by a set of training objects. We will start with nearest neighbor data description and then extend it to K-nearest neighbor data description.

In the one-class classifier or KNN-dd, a test object z is accepted when its local density is larger or equal to the local density of its (first) nearest neighbor in the training set. To estimate the density, a volume of hyper-sphere in d-dimension is assumed around the test point z and it is grown until a predefined number of points are included in this hyper-sphere. Local density is then estimated as follows:

$$pNN(\boldsymbol{z}) = \frac{k/N}{V_k(||\boldsymbol{z} - NN_k^{tr}(\boldsymbol{Z}))} \qquad (61)$$

Here $NN_k^{tr}(\boldsymbol{Z})$ is the k-nearest neighbor of **z** in the training set and V_k is the volume of the cell containing this object which has been grown. For the *local* density estimation, k = 1 is used. This gives us the following indicator function for NN-dd

$$f_{NN^{tr}}(\boldsymbol{z}) = I\left(\frac{1/N}{V(||\boldsymbol{z} - NN^{tr}(\boldsymbol{z})||} \geq \frac{1/N}{V(|\left|NN^{tr}(\boldsymbol{z}) - NN^{tr}\left(NN^{tr}(\boldsymbol{z})\right)\right||}\right) \qquad (62)$$

Here I is the indicator function as used in the case of SVDD. This is equivalent to:

$$f_{NN^{tr}}(\boldsymbol{z}) = I\left(\frac{V(|\left|z - NN^{tr}(\boldsymbol{z})\right||}{V(|\left|NN^{tr}(\boldsymbol{z}) - NN^{tr}\left(NN^{tr}(\boldsymbol{z})\right)\right||} \geq 1\right) \qquad (63)$$

The volume is represented as

$$V(r) = V_d r^d$$

Where d is the dimension and V_d is the volume of unit hyper-sphere. This can be substituted in equation (63) to get the final result as:

$$f_{NN^{tr}}(\boldsymbol{z}) = I\left(\frac{V_d\left||\boldsymbol{z}-NN^{tr}(\boldsymbol{z})|\right|^d}{V_d\left||NN^{tr}(\boldsymbol{z})-NN^{tr}\left(NN^{tr}(\boldsymbol{z})\right)|\right|^d} \geq 1(thr)\right)$$

$$= I\left(\frac{\left||\boldsymbol{z}-NN^{tr}(\boldsymbol{z})|\right|}{\left||NN^{tr}(\boldsymbol{z})-NN^{tr}\left(NN^{tr}(\boldsymbol{z})\right)|\right|} \geq 1(thr)\right) \qquad (64)$$

This means that the distance from object **z** to its nearest neighbor in the training set $NN^{tr}(\boldsymbol{z})$ is compared with the distance from this nearest neighbor $NN^{tr}(\boldsymbol{z})$ to its nearest neighbor $NN^{tr}\left(NN^{tr}(\boldsymbol{z})\right)$. If the ratio between these distances is less than 1 (*a threshold which can be varied*) the test object is accepted in the training target set otherwise it is rejected as outlier. This means that boundary around the target set would be much tighter where objects are close to each other whereas for a uniformly sparsely distributed data, large regions around the training data are accepted. The technique can be visualized with the figure given below:

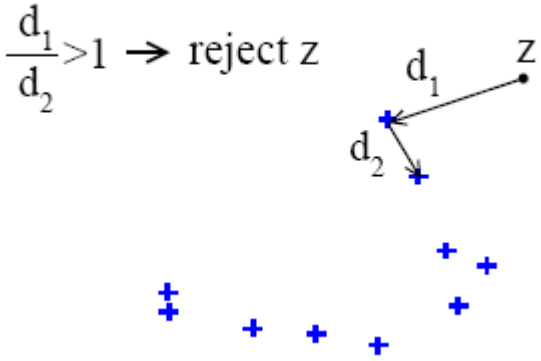

Figure 13: Nearest Neighbor Data Description

The method can easily be extended to a larger number of neighbors 'k'. Instead of taking the first nearest neighbor into account, the k th neighbor should be considered. This

increases the acceptance rate. In case of K-nearest neighbor ratio of average of distances of k-nearest neighbors to the distance of nearest neighbor of nearest neighbor is taken and compared with 1 (or threshold)

2.5 Previous Results Using One Class Classifier

In (C. M. Bishop 1994) Bishop investigated novelty detection and neural network validation for novelty detection. Author explains the quantitative details of how to do novelty detection and finally uses novelty detection as the basis of a practical system for network validation. The specific industrial application of neural networks is concerned with the determination of the oil fraction in a multiphase oil pipeline. In the paper, author considers the problem of validating the outputs generated by a trained neural network, once it has been installed in an application. One of the most important sources of high errors in network predictions arises from novel input data, i.e. data which differ significantly from those used to train the network. In (C. M. Bishop 1994) it is explained that novelty can be quantified by modeling the unconditional probability density of the input data used during training, and by subsequently evaluating this density for all new data points. Further, this can be used to assign error bars to the network outputs, or it can be used to classify input vectors on the basis of a novelty threshold. The paper illustrated the technique of novelty detection using a simple kernel-based estimator of the unconditional input density.

Another example of one class classification is described in (Cohen et al. 2008) which explains the use of Parzen density estimation. Nosocomial infections (NIs) - those acquired in health care settings - represent one of the major causes of increased mortality

in hospitalized patients. They are a real problem for both patients and health authorities. The development of an effective surveillance system to monitor and detect them is of paramount importance. It is very important to detect these kinds of infections for a patient's safety. A positive case would be if a patient does get an infection (nonsocomial) whereas negative would be if a patient does not get this type of infection. In this classification task, the main difficulty lies in the significant imbalance between positive and negative cases. In this application it was shown in [par] that one-class Parzen density estimator attains a very high level (88.6 %) of accuracy in detecting the infection. However, the price paid in terms of loss in specificity is quite exhorbitant. Still, it shows that one-class Parzen density estimator is a promising approach to the detection of non-socomial infections and can become a reliable component of an infection control system.

Finally, (Tax et al. 1999) explains the Support Vector Data Description which is inspired from Support vector machine to detect the outlier objects. The technique is then applied to both real and artificial data. The data was related to detecting faulty situation in a water pump. Different kernels (explained in section 2.3.2.1) were tested on the data. The results obtained using svdd were comparable to the results obtained using other one class classification methods like Parzen density estimation and K-nearest neighbor data description method.

Chapter 3: Experimental Procedures and Methods

In this chapter, we will look at the experimental procedures employed for fall detection in which we applied the techniques (one class classifier) described in chapter 2.We will show how the experiment were set up, how the gathered data looks like, how we retrieve a significant sound event from the noisy background and how we detect and locate a sound event adaptively by real-time signal processing.

3.1 Experimental Set Up

The experiment was conducted in a small lab room where several computers were running all the time. All the windows in the room were closed and the central air-condition was kept running when the experiments were performed. This environment is somewhat similar to the actual living environment of an elderly person's home. If there is no one in the room, the background noise is mainly from the working computers and air-conditioner. We call this kind of living environment as 'silence' because we assume that it lasts forever and no significant or abnormal sound events occur, like clapping, knocking, speaking and falls, etc. The experimental environment is shown in the photo below.

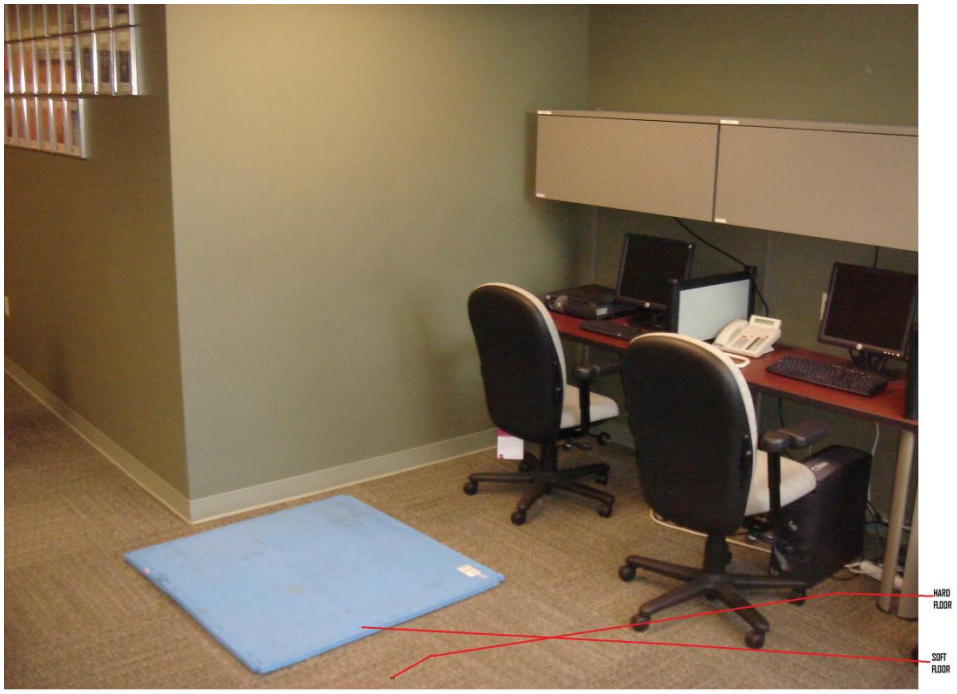

Figure 14: Picture of experimental environment

We also assume that in this experimental setting the silence is kept at a low noise level. In General, the power strength of silence is strongly dependent of the environment outside the room. Specially, to avoid adding unnecessary complexity to the signals extraction process, we assume the noise level is constant if the experimental setup is fixed.

The fall detector (figure 7) consists of a linear array of electric condenser acoustic sensors, also called microphones (two shown: Mic1 and Mic2) mounted on pre-amplifier boards Cana Kit CK495 (about $20 each). More acoustic sensors might be considered in the future to help improve beam forming and source separation. The sensor array is fixed on a stick facing the center of the room space so that the sensors are able to capture the significant sound events occurring in the room.

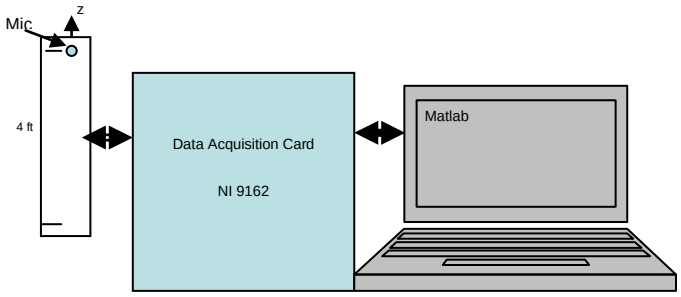

Figure 15: Experimental Setup for Sound Recording

3.2 Data Acquisition

The parameters for recording and the data category are listed in the table below.

Table 1: Experimental data structure

Fs (Sampling Frequency): 20000KHz Channels: Mic1 (High) and Mic2 (Low)		
Training data (1-second length of each file)		
Categories	Scenarios	# of files
Training Fall (30 files)	Falling 1 meter away on the carpet	5
	Falling 3 meters away on the carpet	5
	Falling 5 meters away on the carpet	5
	Falling 1 meter away on the floor	5
	Falling 3 meters away on the floor	5
	Falling 5 meters away on the floor	5
Training Non Fall (60 files)	Ringing the cell phone (music, etc.)	3
	Clapping	5
	Knocking at the desk	3
	Knocking at the wall	4
	Knocking at the metal cabinet	3
	Playing the keys	3
	Hanging down the phone	4
	Talking	4
	Typing	4
	Closing the window	3
	Dropping books on the carpet, 1 meter away	4
	Dropping books on the carpet, 3 meters away	4
	Dropping books on the carpet, 5 meters away	4
	Dropping books on the floor, 1 meter away	4
	Dropping books on the floor, 3 meters away	4
	Dropping books on the floor, 5 meters away	4

Testing data		
File	1 hour long, 36 falls and 72 false alarms (For general purpose, this file contains 18 falls on hard floor and 18 falls on soft carpet. It contains all 60 false alarms listed in the training data and another 12 non-train unknown false alarms.)	1

3.3 Adaptive sound detection algorithm

This section describes how we 'pick' the sound events we are interested in and how we generate a hit for that specific sound event at a specific time position.

3.3.1 Retrieving sound events from background noise

The audio recorded in daily life environment contains background noises. Actually, in a small and environmentally stable clinical or residence room there are noises generated from sources like a working computer, a working refrigerator or a working air-condition. The signals we are concerned with, like clapping, dropping objects, speaking and human falls must be retrieved from the noisy signal without processing the noisy frame. In this case, we define a feature called volume using the short time energy information. Specifically, the volume of the *nth* frame is calculated by

$$v(n) = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} x_n^2(i)} \qquad (65)$$

Here $x_n(i)$ is the ith sample in the *nth* frame audio signal and N is the frame length. The idea is if $v(n)$ is less than a prefixed threshold *Vthr*, it indicates a noise only or silent frame; otherwise, a significant signal frame is detected. This assumption is based on that the background noise level is low enough to retrieve the interesting signals. However, in case of the background noise level is so high that some of the sound events are 'hidden',

a noise reducing algorithm like Wiener filter or Spectrum Subtraction may be applied before volume detection. Thus, a SNR (Signal-to-Noise Ratio) threshold *Ethr*, which makes a decision whether or not use the noise reducing algorithm, has to be prefixed to avoid high computation cost on noise reducing. The computation of energy for each N-point frame is defined as

$$E_N = \sum_{k=0}^{N-1} |X_N(k)|^2 \qquad (66)$$

Where $X_N(k)$ the DFT (Discrete Fourier Transform) of $x(n)$ is defined as

$$X_N(k) = \sum_{n=0}^{N-1} x(n) e^{-j\,2\pi kn\,/N} \qquad k = 0,1 \ldots . N-1$$

Then the SNR is determined by:

$$SNR_N = \frac{E_N x - E_N n}{E_N n} \qquad (67)$$

$E_N x$ And $E_N n$ are the energy of sound event and silence (noise only) respectively. Before $E_N n$ is computed, the average EDS must be taken over an initial set of silent frames. The strategy is if the SNR is larger than *Ethr,* we do not apply any noise reducing algorithm to the file and just go ahead to do the work; otherwise, we do Wiener filtering or Spectrum Subtraction before doing anything. Specifically, the algorithm for retrieving sound events from background noise is stated as

Table 2: Signal retrieving algorithm

```
Take initial M silent frames to calculate E_N n ;
(we assume the first M frames of the whole file are known to be silent)
Then calculate E_N x and SNR_N for each training frame, pick the minimum min SNR_N ;
If min SNR_N >Ethr
   For n=M+1,…#of frames
        Compute v(n) on the current frame;
        If v(n)> Vthr
       Current frame is detected and do processing on it;
   End
 End
   Else
     For n=M+1,…#of frames
         Apply Wiener Filter to the current frame;
         Compute v(n) on the current frame;
         If v(n)> Vthr
              Current frame is detected and do processing on it;
         End
   End
End
```

In the case of our experiment, $\min SNR_N$ (signal of stepping on the carpet) is much larger than *Eth* ($\frac{\min(SNR_n)}{Ethr} > 1$) so that the noise reducing phase can be waived.

3.3.2 Finding the Hit Location of a Sound Event

The fall detection experiment should give us the sound event identity and its exact location. It is necessary to know the exact location of falls so that the ground truth (a vector of time point of where the actual fall has occurred) could be used to access the algorithm's performance. The impulse with an exact position and some degree of confidence, which indicates a sound event has been 'more or less' detected somewhere around the position, is called a hit. Intuitively, the position should be around the centre of

a complete section of a sound event regardless of how long it lasts. That means we allow only one hit for a specific sound event.

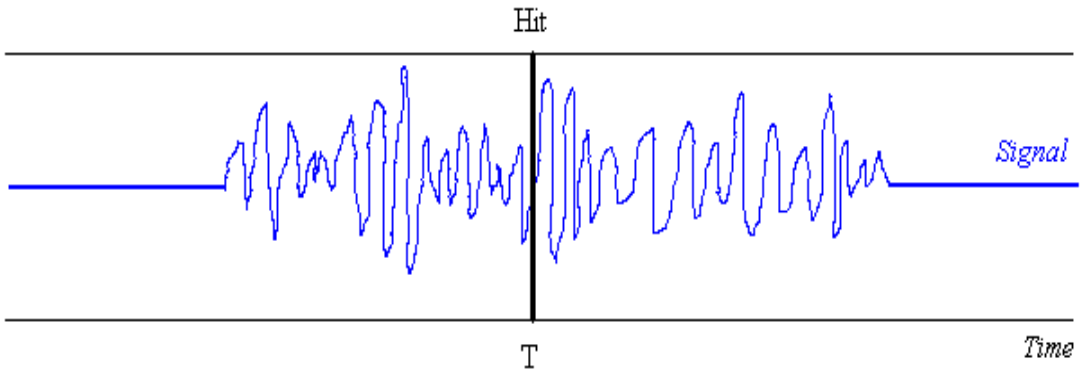

Figure 16: A detected hit

In the figure 11 above, the thick vertical line denotes a hit located at time point T. In this case no confidence is assigned to that hit.

The problem now focuses on how we adaptively locate the hit at a proper place

Table 3: Hits location updating algorithm

```
Initializing count c to 1 and Posit(c);
For each frame;
   If a fall is detected in current frame
        Return the point of the frame, Posit(c+1);
   If Posit(c+1)-Posit(c)≤K (K is a constant integer)
          Update Posit, Posit(c)=[(Posit(c+1)+Posit(c))/2];
          c=c-1;
        End
      c=c+1;
   End
End
```

Since the signal stream is processed frame by frame. When the frame point proceeds, the hit location of that frame is updated properly if a fall is detected in the next frame. Thus, the hit location is iteratively updated until the proceeding of the frame point

ends. The final updated location represents exactly when the event occurs. The adaptive algorithm for exact location of the hit location is given in table 3. The algorithm is basically making sure that we do not count same fall twice. Generally, K should be equal to 1 so that no error detected is allowed in detecting the complete event. However in our case, we choose K=2 to allow at most one error detected frame that could be ignored in detecting the complete event. The algorithm returns the final updated hit location near (a little right to) the centre of the sound event, which makes more sense to tell where it is.

Chapter 4: Results of Fall Detection

As explained in chapter 2 boundary methods can be much more economical as compared to density method because only a boundary is estimated around the target data set (non falls in our case). To determine if our test object is a part of target data or can be considered as an outlier we only have to compute its position relative to some data set. Now, in the case of fall detection outliers are fall events and targets are considered Non Fall events. In this chapter we will discuss fall detection results using One Class Classifiers and compare them with the conventional approach of using two class classifiers. The results are in the form of Receiver Operator Characteristic (ROC) curve which are explained in the next section.

4.1 Support Vector Machine

Support vector data description (one class support vector machine) has several parameters which need to be set on the training data before we can apply our method to actual 1 hour data. The best parameters can be optimized using several methods like 10-fold cross validation, leave-one-out cross validation, reserving a part of training data for testing and a part for training see (Theodorodis et al. 2007) for details. In this thesis we employed leave-one-out cross validation and also taking 50% of training data from one class for training and 50% of that data for testing to search for best parameter settings. The falls are used only in testing.

4.1.1 Best Kernel Function for SVDD

As described in chapter 2, for support vector machine (one class and two classes) we can replace the inner product of two vectors with a Kernel function using which we might be able to map our data to a new feature space where we can get better classification. The introduction of kernel function increases the computation cost but classification results can improve significantly. In this thesis since our focus is on one class classification, kernel function is first determined for support vector data description. Although from literature (Tax. 2004) we know that Gaussian Kernel function should give best results, still for determining the best Kernel function leave-one-out cross validation method was performed on training data and ROC curves were plotted with their area under the curve calculated to determine the best Kernel function. For leave-one-out cross validation one object is tested while all the other objects (only non-fall objects) are used for training. Value of C (in equation 51) was kept 1 for all the Kernel functions whereas n=3 for polynomial kernel and s=22 (average distance between the objects) for Gaussian kernel was used. The following figure and table gives the result:

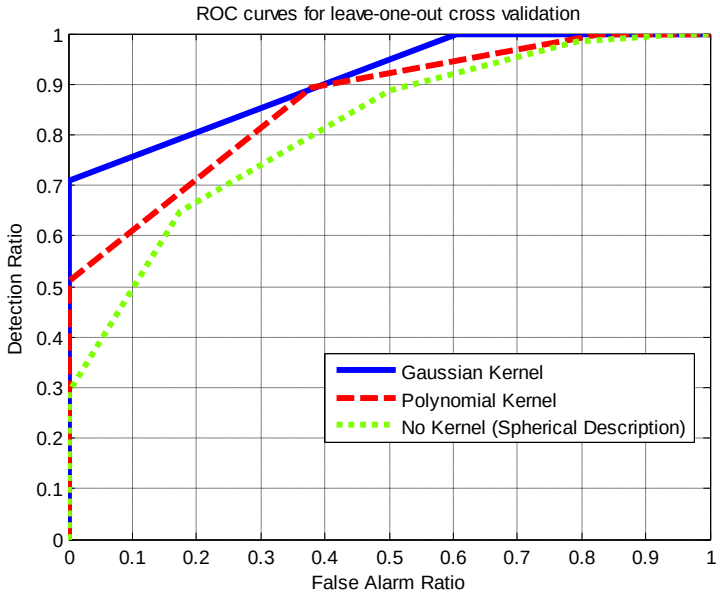

Figure 17: ROC for different kernel functions for SVDD

Table 4: Value of AROC for different kernel functions

KERNEL Function	AROC	PARAMETERS	Number of support vectors out of 59
No Kernel	0.8108	C =1	45
Polynomial	0.8628	C =1, n=3	49
Gaussian	0.9116	C =1, s=22	55

As expected it can be seen clearly the Gaussian kernel gives the best results. Hence from now on we will always use Gaussian kernel for one class support vector data description or SVDD.

4.1.2 Target Error Estimate

After determining the kernel function we move on to the determination of the best setting for the target error. Here target error estimate refers to the estimation of fraction of target objects which will be rejected as outlier by our description when we use them as test objects. A specific target error rate can be achieved using Gaussian kernel in training so that we get the optimum decision boundary for our one class dataset. This error rate is given as:

$$\tilde{\varepsilon} = f_{SV}^{bnd} + f_{SV}^{out} = f_{SV} \qquad (68)$$

Here f_{SV}^{bnd} is the fraction of data on the boundary and f_{SV}^{out} is the fraction of data outside our data description. The latter variable i.e. f_{SV}^{out} can be controlled by C equation (51) whereas the value of f_{SV}^{bnd} can be controlled by the value of s (width parameter) in our Gaussian kernel. This can be explained as follows:

From equation (51) we know that one of the constraints on quadratic optimization is $0 \leq \alpha_i \leq C$. Now, the objects for which Lagrange's multiplier $\alpha_i = C$ correspond to the objects which are completely outside our description whereas objects for which $0 < \alpha_i < C$ are actually the support vectors i.e. objects on the boundary. Also we know that $\sum_{i=1}^{N} \alpha_i = 1$ this tells us that $n_{SV}^{out} C \leq 1$, where n_{SV}^{out} is the number of objects outside the description. This gives maximum value of $C \leq \frac{1}{n_{SV}^{out}}$. Therefore, by choosing the appropriate value of C we can control the number of objects that are outside our description. Next we vary C to get different ROC curves on our training dataset. For these experiments 50% of training data was kept for testing and 50% for training, this portion of 50% was changed 5 times and an average was calculated. Since the dimension of our

data is very large (984) whereas maximum number of training objects (60) are very small, varying the value of C does not make much difference and we end up with pretty much same ROC curves for all target rejection fraction f_{SV}^{out}. Below we give a ROC curve for $f_{SV}^{out} = 0.5$ which means C =0.033. Here we keep value of s=22.

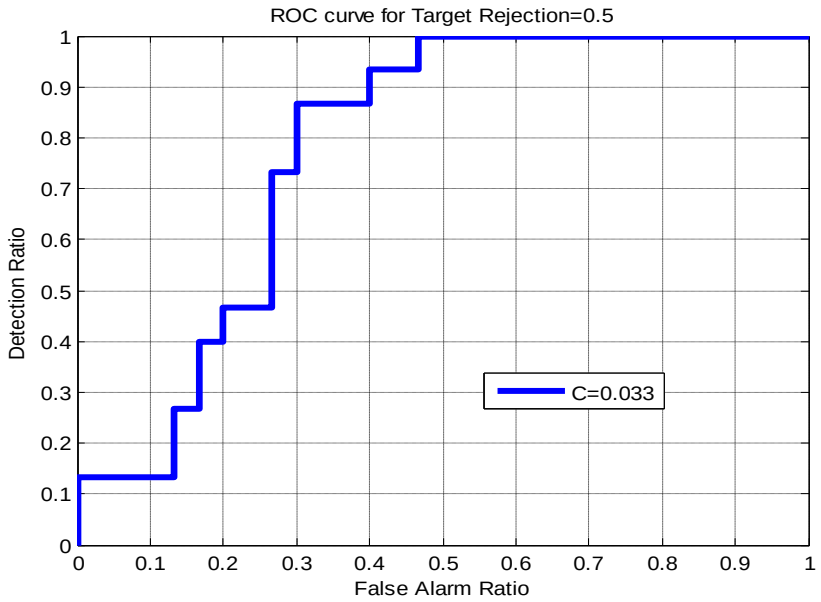

Figure 18: ROC curve for when 50% of objects are rejected

The area under the ROC curve for above figure came out to be 0.7778.

Next we vary the value of width parameter s in our Gaussian Kernel (equation 42) to see what fraction of data should be kept on boundary, in other words how many objects out of the total available one class objects should be made support vectors. The value of s is varied from the highest value of s which is kept as more than the maximum distance (25.5) between any two objects and minimum value of s which is kept as less than the minimum distance (21.1) between any two objects. The results are again compared by

reserving 50% of data for testing and 50% for training and averaging the results of doing it 5 times. The ROC curves for different values of width parameter are shown below:

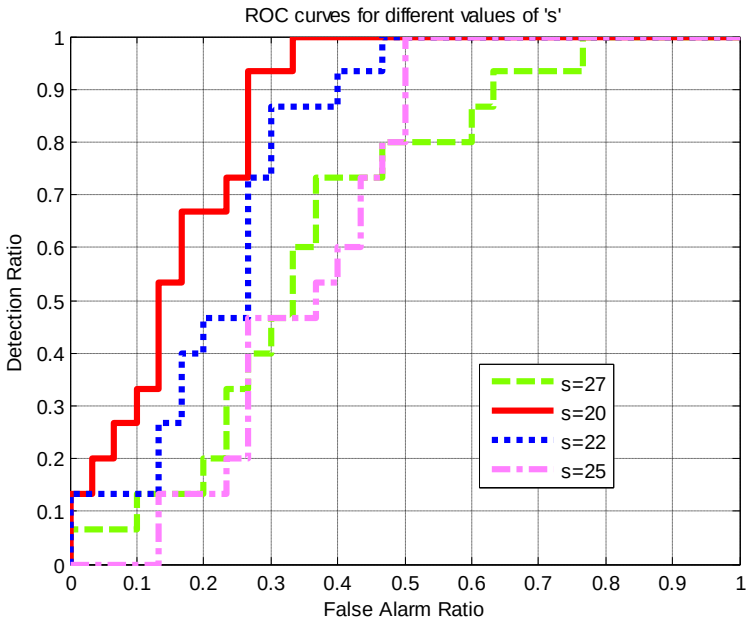

Figure 19: ROC curves for varying width parameter

Table 5: AROC for different width parameter in Gaussian Kernel

Width Parameter Value (s)	Area Under the ROC curve	Number Of Support Vectors out of 30
s=27	0.6533	9
s=25	0.6556	15
s=22	0.7778	25
s=20	0.8467	30

As can be seen from the above table that as the value of *s* is decreased the number of support vectors increase which means that our boundary around one of the classes become more and more tight. High values of *s* correspond to almost spherical description and only a part of the objects become support vectors. Since in our case d>N i.e. dimensionality of objects is much more than number of objects it is necessary that all objects become support vectors as it is also shown by our area under the ROC curve which is maximum for a minimum value of *s*. However, the point to be noted here is that it is actually increase in number of objects which become support vectors which gives better results. We can conclude from above results that for best results with high dimensional data we have to make all training objects as our support vector objects.

4.1.3 Final Results Using One Class and Two Class Support Vector Method

After finding the best parameters for our problem we use the one hour of data as our test data and the whole of training data as our training dataset (see Table 1). The figure below shows the classification results of using one class support vector data description and two class support vector classifier.

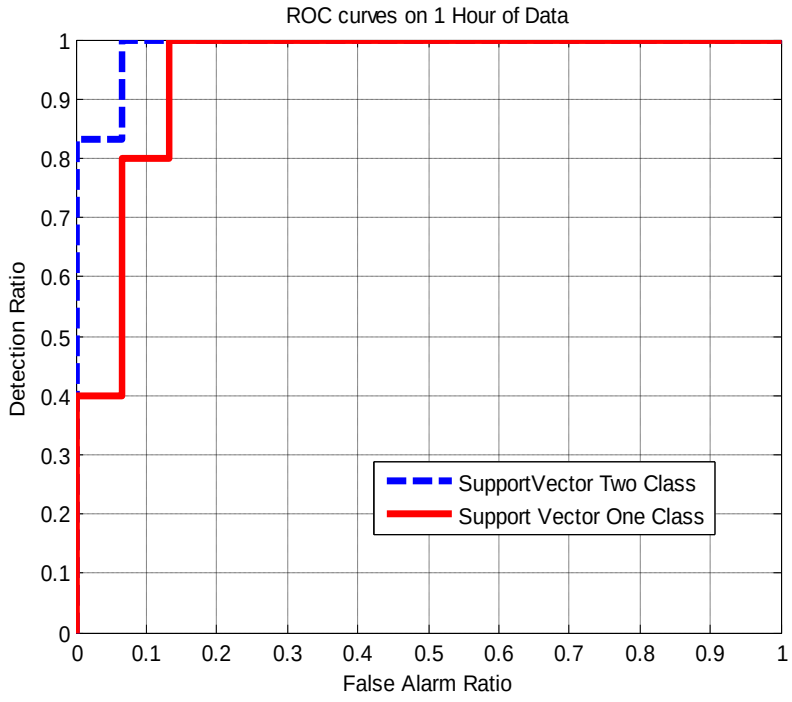

Figure 20: ROC curves for 1 Hour of Data

1. Support Vector Classifier gives an area under the curve of 0.9889.

2. One class classifier or support vector data description gives a little less accuracy as it used examples from only one class to do the classification and the AROC was 0.9467

3. The difference in accuracy between the one class and two class support vector classifier is about 4%.

4. In two class support vector classifier the value of C was varied and finally chosen to be C =10. No kernel function was used as the results using polynomial and Gaussian kernel were no better or worse than without using kernel function.

5. In case of one class classifier the results with Gaussian Kernel were much better that without any kernel or polynomial kernel. Hence we have used Gaussian kernel function

for support vector throughout this thesis. The value of C (equation 51) is very important in case of one class classifier as by reducing this value we can increase the number of target objects rejected during training on one class only. When large fractions of the target set should be rejected, i.e. more than what is expected to lie on the boundary of the dataset C should be adapted usually kept to less than 1. C =1 was used in our case as the numbers of training objects are limited and feature dimension is high and we do not want to reject too many objects.

6. The value of width parameter *'s'* is also important. We iteratively change the value of width parameter from a very small value (less than minimum distance between any two objects) to a very high value (more than the maximum distance between any two objects). As we increase the value of'*s*' the number of support vectors decrease. In our case this value was varied from 20 to 30 (Table 5) to see the results. The minimum value of s gave the best results. On 1 hour of data, we again tried different values of s and a value of 10 was selected as the value of width parameter s.

7. Below the two ROC curves are drawn when the width parameter was 25 and 20 and our test data was 1 hour of data (see Table 1). When the width parameter was 25 the number of objects which became the support vector were 44 out of 60 training objects and the AROC=0.8152 but when we reduce the width parameter to *s*=20 all objects become support vector and AROC=0.8457. The best results were obtained for even lower value, though which is *s*=10. The result of using *s*=10 has been compared with the conventional two class classifier in the above figure 19. If we go below the value of 10 we do not get any result improvement.

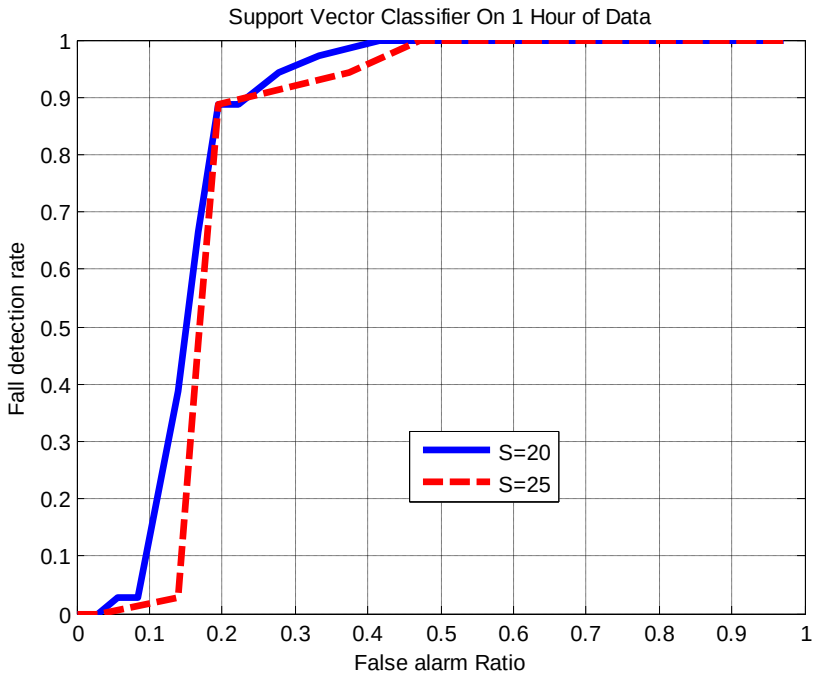

Figure 21: ROC curve for 1 Hour test Data for different values of s

4.2 Results Using Two Classes and One Class K-nearest neighbor Method

As explained in chapter 2 the K-nearest neighbor is one of the easiest and most effective techniques for pattern recognition. For complete explanation of one class and two classes nearest neighbor algorithm see chapter 2. As with support vector data description in this boundary technique too we can reject some part of data as outlier before testing the actual test objects.

4.2.1 Fraction of Data Rejected

The way to reject some training objects is that each object distance is measured from the remaining one class objects we have and the farthest object are successively rejected. The ROC curves are drawn from the remaining one class (target or non-fall) objects using

leave one out cross validation. To draw the ROC curves we use data from both classes but the training data was only from one class (non-falls).

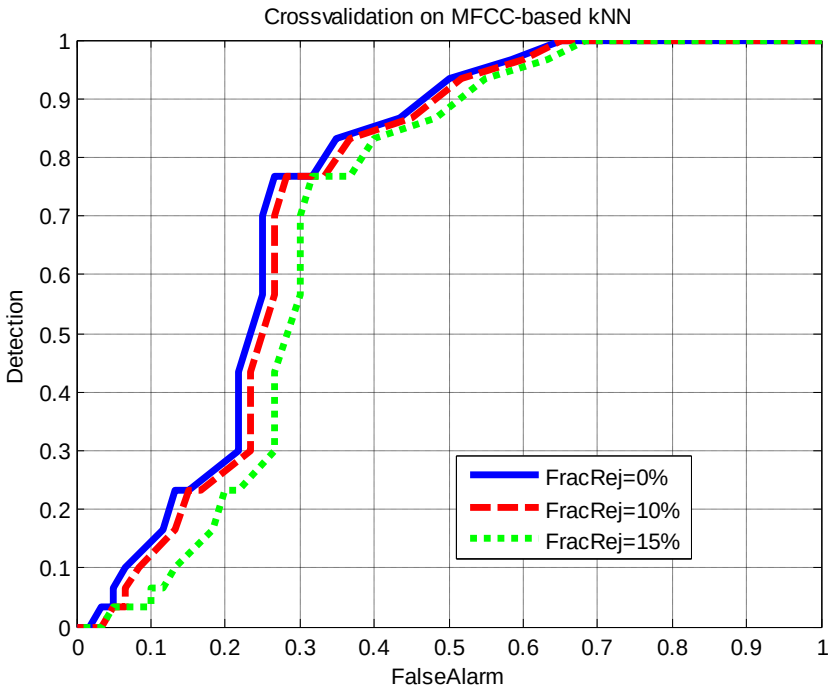

Figure 22: ROC curves for different rejection rates

Table 6: AROC for different rejection rates

Data Rejected (error)	Area Under The ROC Curve	Value of K
0%	0.7533	1
5%	0.7533	1
10%	0.7369	1
15%	0.7014	1

As it can be seen from the above table, that the results get worse when we reject more and more data. The value of k was kept constant (K=1) to compare the effects of number

of objects rejected only. Hence from these experiments we conclude that to get best results for a very high dimensional data (Similar to results with one class support vector where all objects had to be made support vectors) all training objects have to be included in training.

4.2.2 Value of K

Next we vary the value of 'K' to see the effects of changing the number of nearest neighbor we pick. As explained in chapter 2 we take the ratio of distances to accept or reject a particular test object. In case of value of 'K' more than 1 we take the average distance of all the neighbors and compare it with the distance between test object and its nearest neighbor (equation 64). The effect of increasing K can be observed by looking at the ROC curves below:

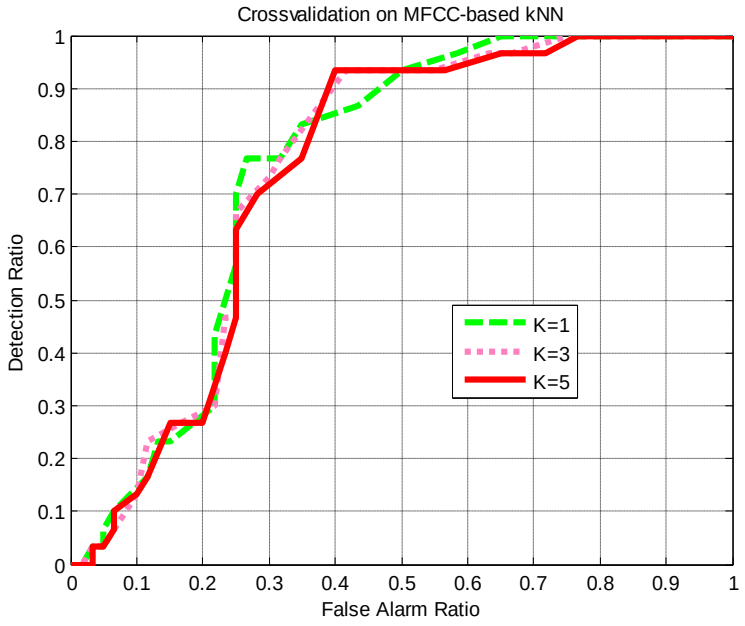

Figure 23: ROC curves for different values of k in k-nearest neighbor

Table 7: AROC for different values of K

Value Of K	Area Under The ROC curve
1	0.7533
3	0.7492
5	0.7419

Hence, the best results are for K=1. The comparison of average distance is done with the nearest neighbor of the nearest neighbor (equation 64) in the one class dataset for all values of K.

As explained in chapter 2 the K-nearest neighbor is one of the easiest most effective techniques for pattern recognition. As shown above we consider only nearest neighbor i.e. K-nearest neighbor with k=1 as it gives best results for leave-one-out cross-validation. Next the ROC curve was plotted for 1 hour of data as our test data; the ROC curves are shown below:

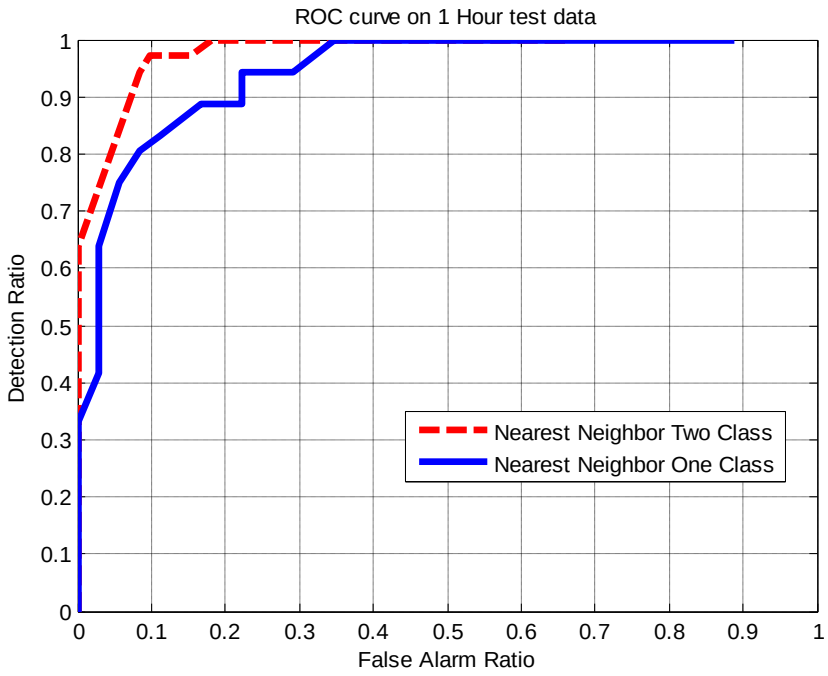

Figure 24: ROC curves for Nearest Neighbor

The area under the ROC curve for two class classifier is 0.9801 and for one class it is 0.9437. The difference between the two classes nearest neighbor classifier and one class nearest neighbor is about 4%. From the ROC curves plotted on 1 hour of data we can conclude that for k=1 the classification performance of one class and two class classifier is pretty close. The threshold in case of one class classifier (equation 64) is always between 1 and 2 signifying very close features of fall and non-fall.

4.3 Results Using Density Methods

Density methods differ from boundary methods in that they require much more data objects to estimate the whole density for each class. For two class case a boundary is set between two class densities. The boundary is then moved to obtain the whole ROC curve by estimating the error on each class. For one class case density of only target class can

be estimated and a threshold can be put on it. A new test object is accepted as target if its density is almost same as the density of target (non-fall) objects. If the density of test object is below the threshold then test object is called an outlier else it is called target.

The most popular method for one class classifier is the Parzen density estimation method. Please refer to chapter 2 for details of this method like object size and classification rules.

4.3.1 Fraction of data rejected

A threshold needs to be set according to a pre-defined fraction of data that has to be accepted from training data. This threshold θ_{fT+} can be defined as:

$$\theta_{fT+}: \frac{1}{N}\sum_{i=1}^{N}(p(\boldsymbol{x_i}) > \theta_{fT+}) = fT+ \qquad (69)$$

Here $p(\boldsymbol{x_i})$ is the density estimate for object $\boldsymbol{x_i}$ and N is our total number of objects.

Now we keep 50% of our no-fall objects for training and 50% for testing. All the falls are kept for testing only. The threshold is set as defined above to reject a pre-defined amount of training data (fraction of data rejected can be obtained from above set $fT+$ as: $fT-= 1-fT+$). For testing we calculate the density of each test object and compare it with only the objects included i.e. $fT+ multiplied\ by\ N$ objects. If the ratio of densities of test object and training object is below 1(a different threshold from above equation (69) the test object is regarded as fall object otherwise it's a classified as non-fall. The fraction of data rejected is then varied using equation (69) and different ROC curves are drawn to test the best fraction rejected. The 50% of data used for training and testing is then successively changed and average classification results are calculated and reported below

for different $fT+$ in the form of ROC curves. Value of width parameter was kept to be 24 which was calculated using Maximum Likelihood Estimate (equation 27)

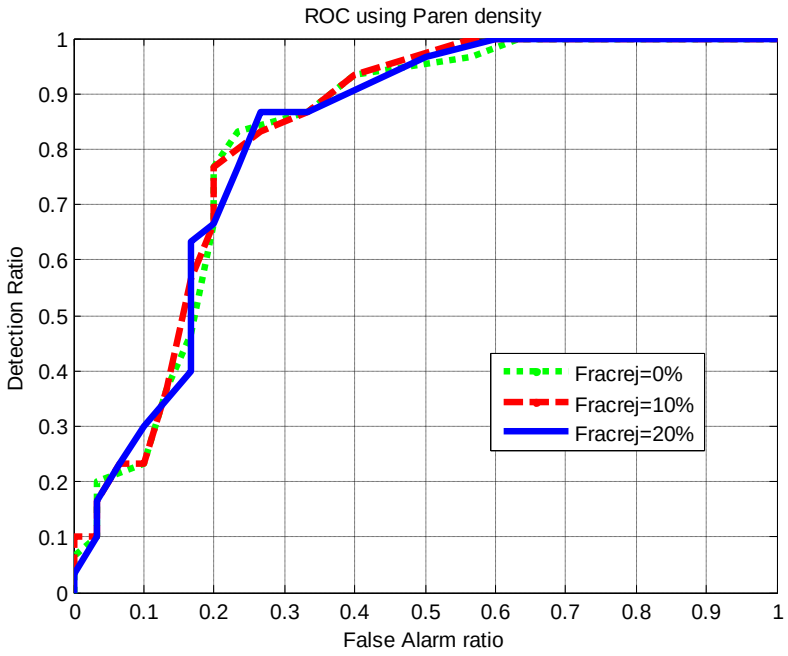

Figure 25: ROC curves for different fractions of data rejected

Table 8: AROC for different rejection rates

Fraction of Data Rejected $fT-$	Area Under the ROC curve	Value of Width Parameter h
0%	0.8228	24
10%	0.8222	24
20%	0.8222	24

It can be seen from the figure and the table above that the difference in the classification results is very little with the change in fraction of data rejected. Hence, the rejection

fraction does not play an important role. The width parameter used above was calculated using leave-one-out maximum likelihood estimate defined in equation (27).

4.3.2 Variation in width parameter h

Next we vary the width parameter '*h*' used in Parzen density estimate. This width parameter is varied and classification results are obtained in the form of ROC curves. The training data is again divided into training and testing data and results are averaged as before.

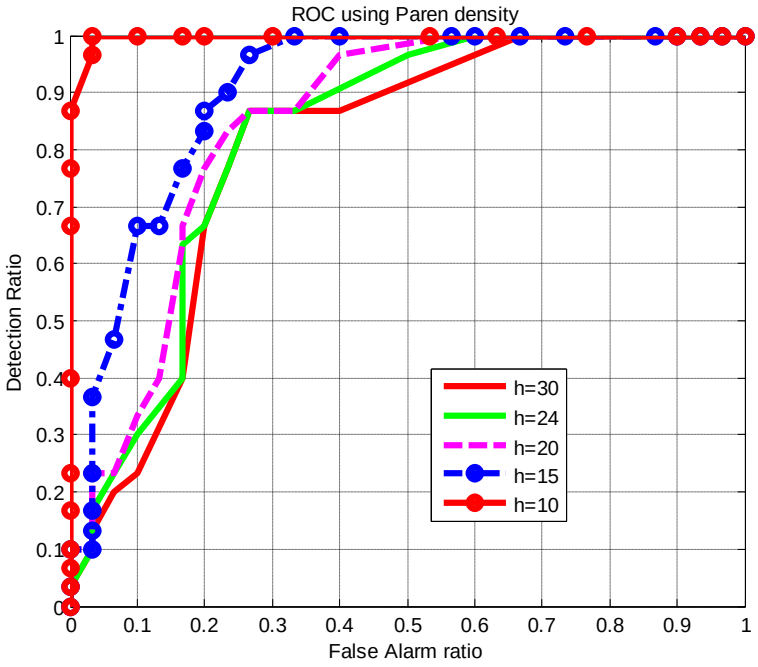

Figure 26: ROC curves for different values of *h*

Table 9: AROC for different values of width parameter

Value of *h* (width parameter)	Area Under The ROC curve	Fraction rejection
30	0.8028	0%
24	0.8222	0%
20	0.8450	0%
15	0.9017	0%
10	0.9972	0%

Hence we can conclude that the minimum value of width parameter gives the best results, Again similar to the case of support vector data description with Gaussian Kernel a lower than a value of 10 for width parameter *h* will give a value of Gaussian Kernel which is very low and does not give any classification results therefore we cannot go below the value of 10 to improve our results. For a value of *'h'* above 24 we again get worse results as can be seen in the Table 9 above.

4.3.3 Final Results Using One Class and Two Class Parzen Density Estimation

Next we use our parameters for testing one hour of data and for training and we use the whole training data for training (table 1).

The value of width parameter for both types of curves was chosen to be h=10, as shown above. The ROC curves plotted when Parzen density estimate was applied to 1 hour of data are given below:

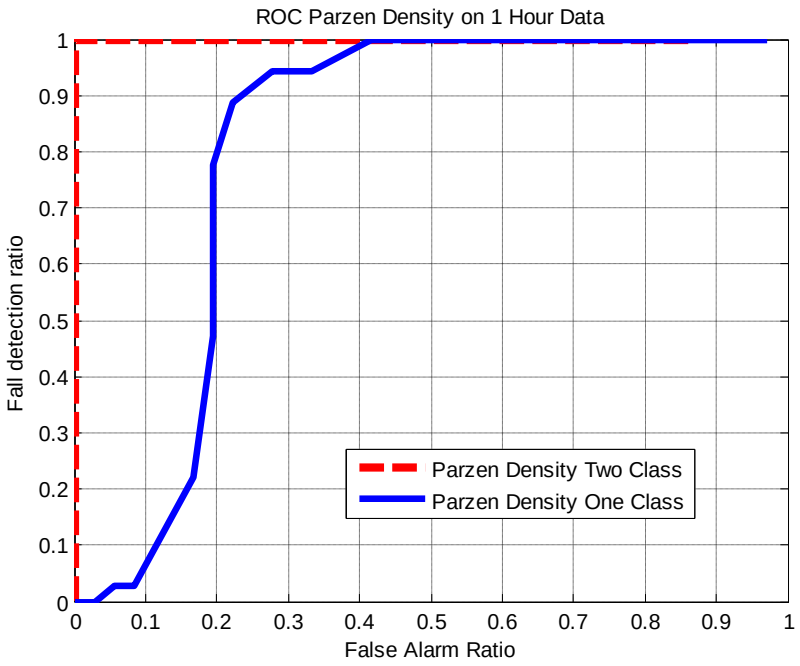

Figure 27: ROC curves for 1 hour of data as Test Data

The area under the ROC curve for two class Parzen Density Estimate was 1.0 but using one class Parzen density estimate it was 0.7843 hence a difference of almost 21%. In one class parzen density estimate we get some false alarms for no detection but the detection rate rises rapidly afterwards. The two class Parzen estimation method seems to work best as it gives no False Alarms for a detection rate of 1.0. i.e an AROC of 1.0. The value of width parameter was kept to be 10 as the best results were obtained using that. The fraction of data rejected was 48 out of 60 i.e. 80% of training data, as that gave us the best results in cross-validation.

4.4 Testing Other Spectrum Based Features

Until now we have used MFCC exclusively as our feature set to do the classification. Now we use other spectrum based features (explained in chapter 2) to see if those features can give us a better classification results. For that we make our feature vector extracted from one frame comprising of Centroid Frequency, Band-Width and ERSB1, ERSB2 and ERSB3. The number of ERSB selected in our feature vector depends on the leave-one-out cross-validation done on our training data set. The ROC curves are drawn using leave one out cross validation as shown below:

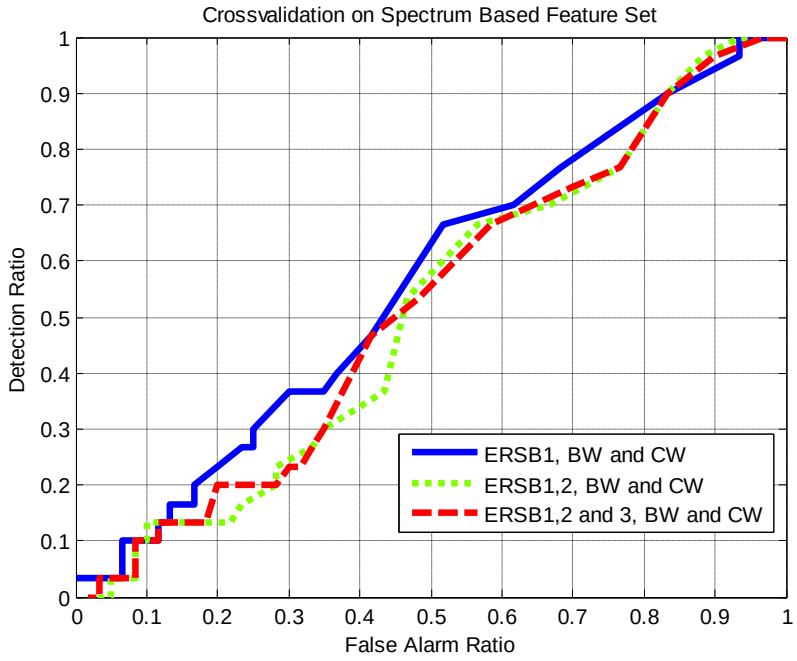

Figure 28: ROC curves for nn-dd with Spectrum Based Features

Table 10: AROC for different Feature set used

Feature Set	Area under the ROC curve
ERSB1, ERSB2, ERSB3, Band-Width (BW) and Centroid-Frequency (CF)	0.5200
ERSB1, ERSB2, BW and CF	0.5100
ERSB1,BW and CF	0.5607

From table 10 above we can conclude that using just ERSB1 combined with Band-Width and Centroid-Frequency for each sound frame gives us the best results for one class nearest neighbor data description. Next, we use ERSB1, CW and BW as our feature set and 1 hour of data as our test data. The ROC curve was drawn on 1 hour of data using Nearest Neighbor Data Description. The ROC curve is compared with the graph of nearest neighbor data description when MFCC were used as sound frame features

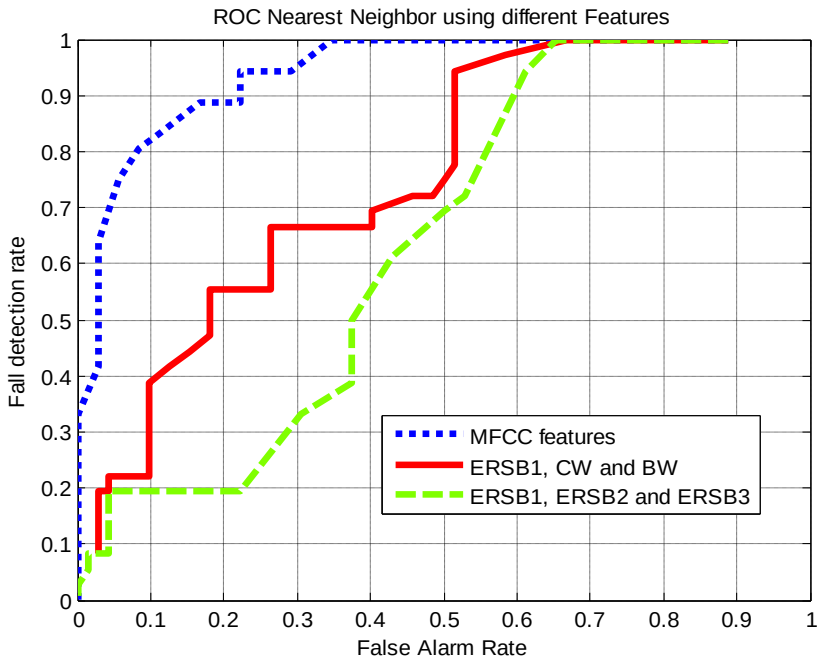

Figure 29: ROC on 1 hour of data as test data

Area under the ROC curve when MFCC were used as sound features was 0.9437 whereas for ERSB1, BW and CF the AROC was 0.7544. Finally when we use ERSB1, ERSB2 and ERSB3 as the features we get an AROC of 0.6400. This clearly shows that MFCC are the best features as far as one class classifiers are concerned for detecting falls from sound signal only.

Chapter 5 Conclusions

In this thesis we have applied One Class Classifiers for doing Fall Detection. The reason for trying to apply these was the fact that it is difficult to obtain samples of one of the classes which is the fall class. Hence there would be no training data available for a conventional classifier. After many experiments with one class classifiers we draw the following conclusions:

1. Results with one class classifier are no worse than 25% (worst case) when compared to conventional two class classifier.

2. In general boundary methods work better than density methods for one class classifiers where the accuracy from conventional classifiers is no worse than 4%.

3. Support Vector Data Description get affected greatly by the choice of C. For our application of fall detection Gaussian Kernel gives the best results which is in consistent with literature review. The value of width parameter was iteratively varied between the more than the maximum and less than minimum distance between any two objects in target class.

4. Support Vector Classifier gives good classification result even without using any kernel function.

Future Research Possibilities

Although fall detection using one class classifier give good results it is far from perfect. The big problem is that both types of algorithms (one class and two class) fail to detect any fall which overlaps with other one or more sound events. In the future work, we can

focus on the separation of overlapped signals by using the microphone array processing technique. Also we used MFCC features for all types of algorithms. We will try to combine these with advanced signal features like Wavelet Transform (WT) to get better classification results with especially one class classification algorithms. Feature selection is another aspect where we plan to work more as the major problem with any classification algorithm is curse of dimensionality which means as we increase the number of features the amount of data required grows exponentially. Even methods like Support Vector Machine cannot escape the curse of dimensionality. Feature selection methods like Principal Component Analysis (PCA) can therefore be an important field to research on in future fall detection projects.

References

Bashshur R. L., (2002) "State-of-the-art telemedicine/telehealth: "L telemedicine and health care," *Telemed. J. e-Health,* Vol. 8, no. l, pp. 5–12, 2002.

Bishop, C. (1995) "Neural Networks for Pattern Recognition" *Oxford University Press*, Walton Street, Oxford OX2 6DP.

Bishop C. M. (1994) "Novelty detection and neural network validation" *Vision, Image and Signal Processing, IEE Proceedings*-, 141(4): pp-217–222, 1994.

Bishop C.M. (1994) "Novelty detection and neural network validation" *IEE, 1994 Paper 1330K (C4, E5),* first received 29th November 1993 and in revised form 19th May 1994.

Bishop C. M. (1994) "Novelty detection and neural network validation" *Vision, Image and Signal Processing, IEE Proceedings*-, 141(4): pp-217–222, 1994.

Cohen G., Sax H., Geissbuhler A. (2008) "Novelty Detection using One-class Parzen Density Estimator. An Application to Surveillance of Nosocomial Infections" *eHealth Beyond the Horizon – Get IT There S.K. Andersen et al. (Eds.)* IOS Press, 2008

Cover T.M. and Hart P.E. (1967) "Nearest Neighbor Pattern Classification," *IEEE Trans Inform Theory* Vol IT-13 pp.21-27 Jan 1967

Istrate D., Castelli E.,Vacher M., Besacier and Serignat J.F. (2006) "Information Extraction From Sound for Medical Telemonitoring" *IEEE Transactions on information technology in Biomedicine,* Vol. 10, No. 2, APRIL 2006.

Japkowicz N., Myers C. and Gluck M. (1995) "A novelty detection approach to classification" *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence,* pages 518–523.

Jennett P. A., Hall L. A., Hailey D., Ohinmaa A., Anderson C., Thomas R., B. Young, D. Lorenzetti, and R. E. Scott, (2003) "The socio-economic impact of telehealth: A systematic review," *J. Telemed. Telecare,* Vol. 9, no. 6, pp. 311– 320, 2003

Kinnunen Tomi, Chernenko E., Tuononen M., Fränti P., Li H., (2007) "Voice Activity Detection Using MFCC Features and Support Vector Machine", *Speech and Dialogue Processing Lab, Institute for Infocomm Research (I^2R),* Singapore
Speech and Image Processing Unit, Department of Computer Science, University of Joensuu, Finland

Li F., Ma J., and Huang D. (2006) "MFCC and SVM Based Recognition of Chinese Vowels" , *Department of Information Science, School of Mathematical Sciences and*

LMAM, Peking University, Beijing 100871, China Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Moya, M., Koch, M., and Hostetler, L. (1993) "One-class classifier networks for target recognition applications" *In Proceedings world congress on neural networks*, pages 797–801, Portland, OR. International Neural Network Society, INNS.

Nelly Christina Laydrus, Eliathamby Ambikairajah and Branko Celler (2007) "Automated Sound Analysis System For Home Telemonitoring using Shifted Delta Cepstral Features" *Proc. of the 2007 15th Intl. Conf. on Digital Signal Processing* (DSP 2007) pp-135-138

Richard, M. and Lippmann, R. (1991) "Neural network classifiers estimate Bayesian a posteriori probabilities" *Neural Computation, 3:* pp-461–483.
[1] Tax D.M.J (2004) "One-class classification Concept-learning in the absence of counter-examples" *ASCI dissertation series number 65.*

Ripley, B. (1996) "Pattern Recognition and Neural Networks" *Cambridge University Press*, Cambridge.

Ritter, G. and Gallegos, M. (1997) "Outliers in statistical pattern recognition and an application to automatic chromosome classification" *Pattern Recognition Letters,* 18: pp 525–539.

Roberts, S. and Penny, W. (1996) "Novelty, confidence and errors in connectionist systems" *Technical report,* Imperial College, London TR-96-1.

Schmandt C. and Vallejo G.,(2003) "LISTENIN" To Domestic Environments From Remote Locations" *Proceedings of the 2003 International Conference on Auditory Display*, Boston, MA, USA, 6-9 July 2003ICAD03-1

Tarassenko, L., Hayton, P., and Brady, M. (1995) "Novelty detection for the identification of masses in mammograms" *In Proc. of the Fourth International IEE Conference on Artificial Neural Networks*, Vol. 409, pp 442–447.

Tax David M.J., Duin Robert P.W. (2004) "Support Vector Data Description**"** *Pattern Recognition Group, Faculty of Applied Sciences,* Delft University of Technology, Lorentzweg 1,2628 CJ Delft, The Netherlands. **Editor:** Douglas Fisher

Theodoridis S. and Koutroumbas K. (2003) "Pattern Recognition" *Academic Press* 84 Theobald's Road, London WClX 8RR, UK pp-711

Vapnik, V. (1998). Statistical Learning Theory John Wiley and Sons Inc, 605 Third Avenue, New York, NY.

Zhu Liu, Yao Wang, Tsuhan Chen, "Audio Feature Extraction and Analysis for Scene Segmentation and Classification", Journal of VLSI Signal Processing 20, 61-79 (1998)