CHAPTER 4

TARGET CLASSIFIER

Features also called as attributes are characteristic signatures of the data that are given as input to the classifier. A highly representative and robust feature extraction technique will certainly improve the classifier performance. The chapter highlights the various state-of-the-art feature extraction and feature selection techniques that lead to the formation of the feature vector for the proposed classifier.

The chapter also gives a detailed description of the Support Vector Machine (SVM) based classifier adopted in this work. The chapter throws light on various machine learning concepts such as over-fitting and bias-variance trade-off, that led to the development of SVM. The performance achieved for the proposed classification task in a multi-class SVM based classifier with various feature extraction and feature selection methods are also discussed.

4.1 Background

Underwater target classification is a highly demanding task owing to the various noise impediments imposed by the underwater environment. This brings out the need for robust classifiers that can efficiently form non-linear decision boundaries. Furthermore, the extraction of target-specific features which carry the characteristic information about the underwater targets inflicted with noise, is also crucial in determining the performance of the classifier. In this work, an underwater target classifier based on Support Vector Machines (SVM) is adopted for classifying eleven classes of acoustic
targets. Suitable feature extraction techniques are employed to extract source-specific target features. Feature selection algorithms working on the extracted features, dynamically select the pertinent features, which gives the best classification performance.

4.2 Feature Extraction

Classification is a pattern recognition problem which identifies the set of categories to which an observation will belong. Different algorithms can be developed to implement the task of classification so that it can predict the class labels of previously unseen observations. The basic block diagram of a classifier is as depicted in Figure 4.1. Classification algorithms typically use features, often referred to as attributes, present in the underlying data as clues for the classification task. Features are the signature patterns that remove redundant information in the signal while representing it best. The classification algorithm operates on the labelled feature set to generate the decision surface in the classification task, and therefore the feature vector should be good predictors of the class membership.

Depending on the classification task, different features may be significant, and hence it is essential to have knowledge about the possible feature set and identify the best among them suited for the underlying classifier. The feature vector, $y$, that is composed of several sets of features should be as discriminative as possible, between the considered classes.

Various feature extraction algorithms working on the input signal can be used to extract the features. The broad classification of acoustic features includes time domain, frequency domain, cepstral domain and higher order spectral features. The choice of the specific features is the result of extensive
experimentation and conclusions that stem from them. The classifier design can be simplified using effectively chosen feature vector.

![Basic block diagram of a classifier](image)

Fig. 4.1 Basic block diagram of a classifier

Most underwater acoustic signals are non–stationary. Short term processing techniques are used to calculate characteristics of an acoustic signal. Let $x(n), n = 1, \ldots, L$ be samples of the acoustic signal and $L$ be length of the signal. The acoustic signal is divided into short-term windows or frames in which the signal is assumed to be ‘quasi–stationary’ and the frames can either be overlapping or non-overlapping. The feature value $f_j$ is calculated for each frame which results in an $M$-element array of feature values $F = f_j$, where $j = 1, \ldots, M$. The length of the feature array is equal to the number of frames; i.e. $M = [(L - S)/N] + 1$, where $N$, is the
window length or say the number of samples, \( S \) is the window step and \( L \) is the total number of acoustic samples of the signal.

### 4.3 Time Domain Features

Time domain features are simple representations of the signal energy changes and can be directly extracted from the time domain. Acoustic signal discrimination based on energy differentiations like energy, energy entropy offer a simple way of acoustic analysis and can be used in combination with elements that contain frequency-related information.

#### 4.3.1 Energy

Most marine acoustic sources are time varying in nature. The standard deviation of the energy sequence can be used to detect signals with large energy variations. Energy can be calculated by the equation 4.1.

\[
E(i) = \frac{1}{N} \sum_{n=1}^{N} |x_i(n)|^2
\]

where \( x_i(n) \) is acoustic sample on \( i \)-th frame of length \( N \).

#### 4.3.2 Energy entropy

Energy entropy is a measure of abrupt changes in the energy level of an acoustic signal. Each frame is further divided into \( k \) sub-frames of fixed duration. For each sub-frame \( j \), the normalised energy \( e_j^2 \) is calculated, i.e., the energy of sub-frame divided by the corresponding short frame energy.

\[
E(i) = \frac{1}{N} \sum_{n=1}^{N} |x_i(n)|^2
\]

\[
e_j^2 = \frac{E_{\text{sub frame} j}}{E_{\text{short frame} i}}
\]
Chapter 4 Target Classifier

The entropy of the sequence is then computed for each frame using the equation 4.4.

\[
H(i) = -\sum_{j=1}^{k} e_j^2 \log_2(e_j^2)
\]  

4.4

Fig. 4.2 Energy and Energy entropy sequence of ship and gunshot noise

Any unexpected variations present in the structure of an acoustic signal can lead to lowering of the energy-entropy. Figure 4.2 shows the magnitude of energy and energy entropy obtained from a ship noise with seemingly lesser energy variations and a continuous series of gunshots which is a typical example of signals having large energy variations.

4.4 Frequency Domain Features

Short-Term Fourier transform or alternatively Short-time Fourier Transform, (STFT), of the acoustic signal computed using Discrete-Time Fourier Transform (DFT) helps in gaining inference on the spectral characteristics of the time-varying signal. The longer time signal is divided
into shorter segments of equal length and DTFT computed separately on each shorter segment.

\[ X(w, n) = \sum_{m=-\infty}^{\infty} x(m) w(n - m) e^{-j\omega m} \]  

where \( w(n) \) is any window function suitable for Short Term Processing. This helps in determining the sinusoidal frequency and phase content of local sections of a signal as it changes over time.

### 4.4.1 Spectral Centroid

The spectral centroid, \( C_i \), of the \( i \)-th frame is defined as the centre of gravity of its spectrum, i.e., the frequency at which the magnitude spectrum can be divided into two portions of approximately equal mass. The centroid of a spectral frame, which is the measure of the spectral shape, can be defined as the average frequency weighted by amplitudes, divided by the sum of the amplitudes.

\[ C_i = \frac{\sum_{k=1}^{N} k x_i(k)}{\sum_{k=1}^{N} x_i(k)} \]

Centroid models the sharpness of sound and textures with high frequencies will possess higher centroid.

### 4.4.2 Spectral Roll–off

Another spectral feature, which gives a measure of the spectral shape, is the spectral roll off, \( RO \), is measure of the amount of the right-skewedness of the power spectrum and is defined as the frequency below which 85% of the magnitude distribution of the signal is concentrated.

\[ i.e. \ RO = \text{Minimum}(R), \text{ such that} \]

\[ \sum_{k=0}^{R} S_k \geq 0.85 \sum_{k=0}^{N-1} S_k \]
4.4.3 Spectral Flux

Spectral flux is a measure of the local spectral change between any two successive frames and a high value indicates rapid change of the power spectrum of signal. It is defined as the difference between squares of normalised magnitudes of the spectra of any two successive frames.

\[ F_{l(i,i-1)} = \sum_{k=1}^{N} (EN_i(k) - EN_{i-1}(k))^2 \]

where \( EN_i(k) = \frac{X_i(k)}{\sum_{i=1}^{N} X_i(k)} \).

Fig.4.3 Frequency-domain features and spectrogram of ship noise

4.4.4 Spectral Entropy

Spectral entropy gives an estimate of the abruptness in the spectrum of a signal. It is computed as

\[ H = -\sum_{f=1}^{N} n_f \log_2 (n_f) \]
where \( n_f \) is the normalized spectral energy computed as

\[
n_f = \frac{E_f}{\sum_{f=0}^{F-1} E_f}
\]

where \( E_f \) is the energy of the \( f \)th sub-band.

Figures 4.3 and 4.4 respectively shows the plots of frequency domain features of a ship noise and a continuous series of gunshots.

Fig.4.4 Frequency-domain features and spectrogram of gunshot noise

Fig.4.5 Plot of energy of different targets
Chapter 4 Target Classifier

The time and frequency features are often grouped into a feature vector which forms the signature pattern for classification. Fig.4.5 to 4.10 shows the plot of various time and frequency features computed for different types of target noises.

Fig.4.6 Plot of energy entropy of different targets

Fig.4.7 Plot of spectral centroid of different targets
Underwater Target Classifier with Improved Success Rate using Meta-Optimal Support Vector Machine

Fig. 4.8 Plot of spectral roll-off of different targets

Fig. 4.9 Plot of spectral flux of different targets

Fig. 4.10 Plot of spectral entropy of different targets
4.5 Cepstral Features

Time domain and frequency domain features cannot always guarantee reliable classification in the presence of composite ambient noise and varying oceanic parameters. Nonlinear techniques such as cepstral analysis are capable of yielding potential features that can better aid in the process of classification of patterns heavily laden with noise. Cepstral analysis refers to a group of homomorphic signal processing methods that is frequently useful in decomposing non-linearly combined signals. The concept of homomorphic analysis, as a technique for non-linear signal processing was proposed by Alan V. Oppenheim [137]. The basic idea of homomorphic analysis is to use non-linearity to transform convolved or non-linearly related signals to additive signals which may then be processed by linear techniques.

The canonic representation of homomorphic systems consists of a cascade of three systems consisting of a system of forward and inverse operations with a linear operation sandwiched in between as shown in Fig.4.11.

Fig.4.11 Canonic representation of a Homomorphic system

The first system consists of an invertible non-linear operation that maps a non-additive combination such as convolution into a simple additive combination. The second system is a linear system obeying additive superposition, and the third system is the inverse of the first nonlinear system. Thus, for signals combined by convolution, a homomorphic
deconvolution system maps convolution into addition, then addition into addition, and finally addition into convolution [138].

The spectrum of a signal can be decomposed into two components, the slowly varying part, referred to as the filter or spectral envelope and the rapidly varying part, referred to as the source or harmonic structure. Separation of these two components can be achieved by taking the Cepstrum, an anagram of the word spectrum, which is a homomorphic transform. Cepstrum is defined as the inverse Fourier transform of the log magnitude of the Fourier transform. The methodology of computation of Cepstrum is depicted in Fig.4.12. Cepstral analysis is capable of separating the filter and source components of the spectrum in a new domain called quefrency.

![Fig.4.12 Methodology for computing Cepstrum](image)

The convolution of any two signals in the time domain can be transformed into multiplication of the signals in the frequency domain through Fourier analysis. Further, on applying logarithm to the Fourier transform, convolution in the time domain can be transformed into sum of log-magnitude components in the frequency domain. Now applying an inverse Fourier transform to the log spectrum takes the function back into the time domain, and gives a measure of the rate of change of the spectral magnitudes.

Consider a signal \( s(t) \) as a convolution of the two components, \( x(t) \) and \( y(t) \), so that

\[
s(t) = x(t) * y(t)
\]

Then, taking Fourier transforms of both sides,
Chapter 4 Target Classifier

\[ s(\omega) = X(\omega)Y(\omega) \]  \hspace{1cm} 4.11

The magnitude spectrum of the signal can be written as

\[ |S(\omega)| = |X(\omega)||Y(\omega)| \]  \hspace{1cm} 4.12

and taking the logarithms of both sides gives,

\[ \ln|S(\omega)| = \ln|X(\omega)| + \ln|Y(\omega)| \]  \hspace{1cm} 4.13

Thus, convolution in time has been transformed into a sum of log-magnitude components in the frequency domain. The individual components may be separated from each other using a suitable cepstral filter referred to as lifter (anagram of filter). They may then be transformed back by applying Inverse Fourier Transform (IFT). The IFT takes the signal to a domain, similar to the frequency domain, called the quefreny (anagram of frequency) domain. However, the phase information will be lost as a result of applying the magnitude operation.

Applying an inverse Fourier transform to the log spectrum gives

\[ F^{-1}\{\ln|S(\omega)|\} = F^{-1}\{\ln|X(\omega)|\} + F^{-1}\{\ln|Y(\omega)|\} \]  \hspace{1cm} 4.14

For the signal \( s(t) = x(t) * y(t) \) the cepstra is given by

\[ c_s(n) = c_x(n) + c_y(n) \]  \hspace{1cm} 4.15

\( c_x(n) \) and \( c_y(n) \) are the cepstra of the signals \( x(t) \) and \( y(t) \) respectively.

As the Cepstrum is derived from the power spectrum of the signal, it is always a real function of frequency. Because the log-magnitude spectrum is real and symmetrical for real signals, the final IFT can also be replaced with a Discrete Cosine Transform (DCT).
Cepstral feature extraction schemes have been developed for speech recognition application. However, they have also proved to be successful in other acoustic recognition applications such as audio forensics [139], audio watermark detection [140], acoustic environment identification [141] and underwater target recognition [142].

4.6 Linear Prediction Coefficients

Linear Prediction Coefficients (LPC) are conventional features used in speech processing. According to the speech synthesis model, speech can be modelled as the output of a linear time-invariant system. LP analysis provides a robust, reliable, and accurate method for estimating the parameters that characterise the linear time-varying system representing the vocal tract [143]. The basic idea of LP analysis is that, at a particular time $k$, the signal sample is represented as a linear sum of $n$ previous samples

$$S_n = \sum_{k=1}^{p} a_k S_{n-k}$$  \hspace{1cm} 4.16

in which $a_k$ is known as the predictor coefficients and $p$ is the prediction order. The predictor coefficients provide a good estimate of the spectral properties of the speech signal and are often used as features. The two widely used methods for estimating the LP coefficients are the autocorrelation method and the covariance method described in section 4.6.1 and 4.6.2 respectively. Figure 4.13 shows the plot of log magnitude of frequency response of LPC coefficients obtained with autocorrelation method and covariance method.

Both methods assume that neither the vocal tract shape nor the glottal waveform changes, and determines the predictor coefficients in such a way that, the prediction error $E$, is minimised in the least squared sense.
Chapter 4 Target Classifier

\[ E = \sum_{n=-\infty}^{\infty} e^{2} (n) = \sum_{n=-\infty}^{\infty} (s(n) - \sum_{k=1}^{p} a_{k} s(n - k))^{2} \quad 4.17 \]

![Log magnitude plot of STFT of a windowed signal of a humpback whale](image)

![Log magnitude plot of frequency response of filter coefficients obtained with LPC (autocorrelation method) analysis](image)

![Log magnitude plot of frequency response of filter coefficients obtained with LPC (covariance method) analysis](image)

Fig. 4.13 Plot of log magnitude of frequency response of filter coefficients obtained with LPC by autocorrelation method and covariance method

In the linear prediction model, the filtering action of the vocal tract, the radiation, and the glottal flow is represented by a discrete linear filter with \( p \) poles [144]. The transfer function of this filter in the complex \( z \) domain is related to the predictor coefficients by equation 4.18.

\[ H(z) = \frac{1}{[1 - \sum_{k=1}^{p} a_{k} z^{-k}]} \quad 4.18 \]

The transfer function \( H(z) \) is related to the samples of the impulse response of the filter by equation 4.19

\[ H(z) = \sum_{n=0}^{\infty} h_{n} z^{-n} \quad 4.19 \]

On substituting equation 4.19 in equation 4.18, the relationship between predictor coefficients and the samples of the impulse response can be derived as in equation 4.20

\[ h_{n} = \begin{cases} \sum_{k=1}^{p} a_{k} h_{n-k} & n > 0 \\ 1 & n = 0 \\ 0 & n < 0 \end{cases} \quad 4.20 \]
The first $p$ samples $h_1, h_2, \ldots, h_p$ are sufficient to determine the $p$ predictor coefficients uniquely [144]. Thus the linear prediction characteristics of the speech wave can be represented by $p$ numbers, $h_1, h_2, \ldots, h_p$.

### 4.6.1 Autocorrelation Method

In this method the values of the predictor coefficients $a_k$ that minimise $E$ are found by assigning the partial derivatives of $E$ of the windowed speech signal with respect to $a_k$ to zeros.

$$\frac{\partial E}{\partial a_k} = 0 \text{ for } k = 1, \ldots, p$$ \hspace{1cm} 4.21

which yields $p$ equations with $p$ unknown variables as

$$\sum_{k=1}^{p} a_k \sum_{n=-\infty}^{\infty} s(n-i) s(n-k) = \sum_{n=-\infty}^{\infty} s(n-i) s(n),$$ \hspace{1cm} 1 \leq i \leq p 4.22

which can be expressed in terms of autocorrelation function as

$$\frac{\partial E}{\partial a_k} = 0 \text{ for } k = 1, \ldots, p$$ \hspace{1cm} 4.23

$$\sum_{k=1}^{p} R(|i-k|) a_k = R(i), \quad 1 \leq i \leq p$$ \hspace{1cm} 4.24

The set of linear equations given by equation 4.24 can be represented in the matrix form as

\[
\begin{bmatrix}
R(0) & R(1) & \ldots & R(p-1) \\
R(1) & R(2) & \ldots & R(p-2) \\
\vdots & \vdots & \ddots & \vdots \\
R(p-1) & R(p-2) & \ldots & R(0)
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
\vdots \\
a_p
\end{bmatrix} =
\begin{bmatrix}
R(1) \\
R(2) \\
\vdots \\
R(p)
\end{bmatrix}
\] \hspace{1cm} 4.25

The equation 4.25 can be represented as

$$Ra = r$$ \hspace{1cm} 4.26

The resulting matrix is a Toeplitz matrix where all elements along a given diagonal are equal. This allows the linear equations to be solved by the Levinson-Durbin algorithm.
Chapter 4 Target Classifier

4.6.2 Covariance Method

The covariance method is very similar to the autocorrelation method, with the difference that the covariance method windows the error signal instead of the original speech signal. The energy $E$ of the windowed error signal is

$$E = \sum_{n=-\infty}^{\infty} e^2(n) w(n) \quad 4.27$$

The $p$ equations with $p$ unknown variables in equation 4.22, obtained by setting the partial derivative of the error signal with respect to $a_k$ to zero is expressed in terms of the covariance function as

$$\sum_{k=1}^{p} \varphi(i, k) a_k = \varphi(i, 0), \quad 1 \leq i \leq p \quad 4.28$$

are coefficients of a linear predictive filter in which value of next sample is determined by a linear combination of previous samples.

The set of linear equations given by equation 4.28 can be represented in the matrix form as

$$\begin{bmatrix} \varphi(1,1) & \varphi(1,2) & \ldots & \varphi(1,p) \\ \varphi(2,1) & \varphi(2,2) & \ldots & \varphi(2,p) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi(p,1) & \varphi(p,2) & \ldots & \varphi(p,p) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} \varphi(1,0) \\ \varphi(2,0) \\ \vdots \\ \varphi(p,0) \end{bmatrix} \quad 4.29$$

The equation 4.29 can be represented as

$$\Phi a = \Psi \quad 4.30$$

Since $\Phi$ is symmetric, this system of equations can be solved efficiently using Cholesky decomposition in $(p^3)$.  

Figures 4.14 and 4.15 shows the plots of LPCs of five records different target types, namely humpback whale, ship, boat, sealion and snapping shrimp.
Underwater Target Classifier with Improved Success Rate using Meta-Optimal Support Vector Machine

Fig. 4.14 Plot of LPCs of Humpback whale noise

Fig. 4.15 Plots of LPCs of different target types - ship, boat, sealion and snapping shrimp
4.7 Linear Prediction Cepstral Coefficients

Linear Prediction Cepstral Coefficients (LPCC) are an important representation of speech derived from linear prediction model and is obtained by considering the power series expansion of the logarithmic transfer function \( \ln H(z) \) in powers of \( z^{-1} \). If all the poles of \( H(z) \) are inside the unit circle, \( \ln H(z) \) can be expressed by equation 4.31.

\[
\ln H(z) = C(z) = \sum_{n=1}^{\infty} c_n z^{-n}
\]  

(4.31)

A simple and unique relationship exist between the parameters \( c_n \) and \( \alpha_n \), which is obtained by substituting equation 4.18 into equation 4.31 and taking derivatives on both sides with respect to \( z^{-1} \).

\[
\frac{d}{dz^{-1}} \ln \left[ \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}} \right] = \frac{d}{dz^{-1}} \sum_{n=1}^{\infty} c_n z^{-n}
\]

(4.32)

which can be simplified to

\[
\frac{\{\sum_{k=1}^{p} k a_k z^{-k+1}\}}{\{1 - \sum_{k=1}^{p} a_k z^{-k}\}} = \sum_{n=1}^{\infty} n c_n z^{-n+1}
\]

(4.33)

and rearranged as

\[
\sum_{k=1}^{p} k a_k z^{-k+1} = \left(1 - \sum_{k=1}^{p} a_k z^{-k}\right) \sum_{n=1}^{\infty} n c_n z^{-n+1}
\]

(4.34)

Equating the constant terms and the powers of \( z^{-1} \) on the LHS and RHS, we get

\[
c_1 = a_1
\]

(4.35)

\[
c_n = \sum_{k=1}^{n-1} \left(1 - k/n\right) a_k c_{n-k} + a_n, \quad 1 < n < p
\]

\[
c_n = \sum_{k=1}^{n-1} \left(1 - k/n\right) a_k c_{n-k, n > p}
\]

The coefficients \( c_n \)'s can be computed from the predictor coefficients and vice versa by equation 4.35. \( c_n \)'s are the samples of the Cepstrum and are popularly known as Linear Predictive Cepstral Coefficients (LPCC). The Cepstrum is defined as the inverse Fourier transform of the log magnitude of the Fourier transform. However, for a transfer function with poles only, the
Cepstrum can be obtained directly from the impulse response samples $h_n$ by equation 4.36 or by the predictor coefficients $a_k$ by equation 4.35.

$$c_1 = h_1$$
$$c_n = \sum_{k=1}^{n-1} \left( 1 - \frac{k}{n} \right) h_k c_{n-k} + h_n, \quad 1 < n$$

Equation 4.36

LPC and LPCC were developed for speech recognition applications. However, LPCs and LPCCs in conjunction with other features have been used for other acoustic recognition applications, such as audio based event detection[145] and underwater target recognition [86].

![Fig.4.16 Plots of LPCCs of noise of different target types - ship, boat, sealion and snapping shrimp](image-url)
Fig. 4.17 Plot of LPCCs of Humpback whale noise

Figures 4.16 and 4.17 shows the plots of LPCCs of five noise records of different target noises, namely ship, boat, sealion and snapping shrimp and humpback whale.

4.8 Perceptual Linear Prediction (PLP) Cepstral Coefficients

In Perceptual Linear Prediction (PLP) technique, the properties of human hearing are simulated by practical approximations, and the auditory like spectrum of speech is approximated by an autoregressive pole model. The steps for computing PLP cepstral coefficients is depicted in Figure 4.18.

The first stage in the PLP feature extraction process is to frame the signal and apply a suitable windowing technique. Typically, hamming window, which avoids discontinuities by shrinking the values of signal towards zero at the window boundaries, is applied. Hamming window can be represented by equation 4.37.

\[
w(n) = \begin{cases} 
-0.54 - 0.46 \cos \frac{2\pi n}{L}, & 0 \leq n \leq L \\
0, & \text{Otherwise}
\end{cases} 
\]

4.37
The spectral information of the windowed output is computed by taking Fourier transform (discrete Fourier transform or fast Fourier transform).

The next step in computing PLP coefficients is the critical band analysis. Critical band is a concept introduced by Harvey Fletcher [146] and describes the frequency bandwidth of the auditory filter created by the cochlea. Critical band models the change in hearing threshold around a sound as a filter. Psycho-acoustically, the critical bandwidth can be measured through the concept of masking. The critical band is the band of audio frequencies in which the narrow band of noise surrounding the tone causes auditory masking of the tone, when the power of the noise in this band (the critical band) is equal to the power in the tone. Critical bands are of great importance in understanding many auditory phenomena such as perception of loudness, pitch, and timbre.
Bark scale is a non-linear frequency scale which models the resolution of the human hearing system. The bark scale is defined so that the critical bands of human hearing each have a width of one Bark. The Bark scale ranges from 1 to 24 Barks, corresponding to the first 24 critical bands of hearing. The Bark-hertz transformation is given by

$$
\Omega(\omega) = 6\ln\left\{ \frac{\omega}{1200\pi} + \left[ \left( \frac{\omega}{1200\pi} \right)^2 + 1 \right]^{0.5} \right\}
$$

where $\omega$ is the angular frequency in rad/s. The first step in the critical band analysis, is to warp the power spectrum calculated along the frequency axis $\omega$ into the Bark frequency $\Omega$ by equation 4.38. The relation between frequency in Hz, and Bark frequency is shown in Fig.4.19. The warped power spectrum is then convolved with the simulated critical band masking curve $\psi(\Omega)$ given by

$$
\psi(\Omega) = \begin{cases} 
0 & \text{for } \Omega < -1.3 \\
10^{2.5(\Omega+0.5)} & \text{for } -1.3 \leq \Omega \leq -0.5 \\
1 & \text{for } -0.5 < \Omega < 0.5 \\
10^{-(\Omega-0.5)} & \text{for } 0.5 \leq \Omega \leq 2.5 \\
0 & \text{for } \Omega > 2.5
\end{cases}
$$

Fig.4.19 The Bark scale
In order to compensate the unequal sensitivity of human hearing at different frequencies, the next processing stage in PLP analysis pre-emphasizes the power spectrum processed by the critical band analysis, by the simulated equal loudness curve. The approximation is given by equation 4.40 and represents a transfer function of a filter with asymptotes of 12 dB/oct between 0 and 400 Hz, 0 dB/oct between 400 and 1200 Hz, 6 dB/oct between 1200 and 3100 Hz, and 0 dB/oct between 3100 Hz and the Nyquist frequency. For moderate sound levels, this approximation is reasonably good up to 5000 Hz.

\[
E(\omega) = \frac{[(\omega^2 + 56.8 \times 10^6)\omega^4]}{[(\omega^2 + 6.3 \times 10^6)^2(\omega^2 + 0.38 \times 10^9)]} \quad 4.40
\]

For applications requiring a higher Nyquist frequency, an additional term representing a steep (about -18 dB/oct), decrease of the sensitivity to hearing for frequencies higher than 5000 Hz is incorporated. Equation 4.40 would then become

\[
E(\omega) = \frac{[(\omega^2 + 56.8 \times 10^6)\omega^4]}{[(\omega^2 + 6.3 \times 10^6)^2(\omega^2 + 0.38 \times 10^9)(\omega^6 + 9.58 \times 10^{36})]} \quad 4.41
\]

To simulate the non-linear relation between the intensity of sound and the human perception of loudness, the pre-emphasized signal is subjected to cubic root amplitude compression.

\[
\Phi(\Omega) = \Xi(\Omega)^{0.33} \quad 4.42
\]

This operation is an approximation to the power law of hearing and also helps reducing the spectral amplitude variation of the critical-band spectrum so that the all-pole modelling can be done by a relatively low model order.

The next step in PLP analysis, is to approximate \( \Phi(\Omega) \) by the spectrum of an all-pole model using the autocorrelation method of all-pole spectral
modelling. The auto regressive coefficients of the all-pole model gives the PLP coefficients. PLP coefficients, like linear prediction coefficients, themselves can be used as acoustic features. However, transforming the PLP coefficients into the cepstral domain yields more robust features. The PLP coefficients can be converted into cepstral coefficients by equation 4.43.

\[ c_1 = a_1, \]
\[ c_n = \sum_{k=1}^{n-1} \left(1 - \frac{k}{n}\right) a_k c_{n-k} + a_n, \quad 1 < n < p \]
\[ c_n = \sum_{k=1}^{n-1} \left(1 - \frac{k}{n}\right) a_k c_{n-k}, \quad n > p \]

Fig.4.21 and Fig. 4.21 shows the plots PLP cepstral coefficients of five records of different target noises, namely ship, boat, sealion, snapping shrimp and humpback whale.

Fig.4.20 Plots of PLP cepstral coefficients of different target types - ship, boat, sealion and snapping shrimp
Mel Frequency Cepstral Coefficients

Mel Frequency Cepstral Coefficients (MFCCs) are acoustic features widely used in automatic speech recognition systems. MFCCs are based on the human auditory system in which the perception sensitivity varies with frequency. Extensive studies on human inner ear have shown that basilar membrane, a portion of inner ear, stimulation of different areas of which are perceived as different pitches or tones, can be simulated using a bank of filters [147].

Mel, short for melody, is a psychoacoustic perceptual scale that provides the relation between pitch, perceived frequency, as a function of frequency [148]. The Mel scale, first formulated by Stevens et al., is a heuristically derived scale, and attempts to represent the psychological sensation of pitch of the human ear on a linear scale. Stevens et al. (1937) [149] organized experiments in which subjects were required to adjust the frequency of a stimulus tone to be half as high as that of a comparison tone. Based on the experimental results, the Mel scale was developed, but with the drawback of being subject dependent. Later in 1940, the original Mel scale
was revised in which differences among the test subjects were resolved. Equal increments in the Mel scale correspond to equal increments of perceived pitch of pure tone stimuli. The Hz to Mel transformation can be achieved by the formula as in equation 4.44 and is shown in Figure 4.22.

\[ m = 2595 \log_{10} \left( 1 + \frac{f}{700} \right) \]  \hspace{1cm} 4.44

![Mapping between Hz and Mel scale](image)

**Fig.4.22 The Mel scale**

MFCCs are systematically computed by taking the real Cepstrum of a windowed signal derived from the Fast Fourier Transform of Mel scaled signal. The discrete cosine transform of the real logarithm of the energy spectrum expressed on Mel scale gives the Mel Frequency Cepstral Coefficients. The steps for computing MFCCs are depicted in Figure 4.23.

The first stage in the MFCC feature extraction process is pre–emphasis filtering which boosts the energy in the higher frequencies. Typically, in case of speech signals for which MFCC were derived and is normally applied, a spectral tilt exists since more energy is concentrated in the lower frequencies than at the higher frequencies, which is caused by the
nature of excitation of glottal pulse. Therefore, boosting the high frequencies makes information from these higher formants more available to the acoustic model.

![Diagram showing steps in computing MFCCs](image)

**Fig. 4.23 Steps in computing MFCCs**

The next step in the computation of MFCC is to frame the signal and apply a suitable windowing technique. The spectral information of the windowed output is computed by taking Fourier Transform (discrete Fourier transform or fast Fourier transform) which is then passed through a Mel scaled filter bank. The filter bank is implemented as a set of triangular band-pass filters as shown in Figure 4.24, with spacing and bandwidth determined by a constant Mel-frequency interval. In the spectral domain, it corresponds to a set of non-uniformly spaced filters with more and narrow filters in the low frequency region and less and wide filters in the high frequency region.
to account for more discriminative lower frequencies and less discriminative higher frequencies.

![Mel filter bank](image)

**Fig. 4.24 Mel filter bank**

The next step in computing MFCC is to compute the logarithm of the square magnitude of the output of Mel-filter bank, which leads to compression of the dynamic range. Taking logarithm models the logarithmic sensitivity of human ear to sound amplitudes, as humans are less sensitive to slight differences in amplitude at high amplitudes than at low amplitudes. Therefore taking log of filter energies makes frequency estimates less sensitive to slight variations in input such as power variations.

The final stage in computation of MFCC is to take the Discrete Cosine Transform (DCT) of the log filter bank energies which transforms the log Mel spectrum back into the spatial domain. Even though DFT can also be used for the same purpose, DCT is preferred for its property to concentrate the information to a relatively fewer number of coefficients [150].

Figures 4.25 and 4.26 shows the plots MFCCs of five records of different target noises, namely humpback whale, ship, boat, sealion and snapping shrimp.
**Fig.4.25** Plot of MFCCs of Humpback whale noise

**Fig.4.26** Plots of MFCCs of different target types - ship, boat, sealion and snapping shrimp
4.10 Gammatone Cepstral Coefficients

The Gammatone cepstral coefficients (GTCCs) are biologically inspired modification to MFCCs in which Gammatone filters equally spaced in the Equivalent Rectangular Bandwidth (ERB) scale is employed. ERB is a psychoacoustic measure, which approximates the bandwidth of the auditory filter at each point along the cochlea as the bandwidth of a rectangular filter, having the same peak transmission as the auditory filter and which passes the same total power for a white noise input [151]. The equation describing the value of ERB as a function of centre frequency, \( F \) (in hertz), is

\[
\text{ERB} = 24.7(0.00437F + 1)
\]

As per the above equation, the ERB value at a centre frequency of 1 kHz is approximately 132 Hz, which corresponds to one step of ERB number in the ERB scale. The Hz to ERB transformation can be achieved by the formula as in equation 4.46 and is shown in Figure 4.27.

\[
\text{ERB}_{\text{scale,1step}} = 21.4 \log(0.00437F + 1)
\]

![Fig. 4.27 The ERB scale](image)
The Gammatone filter bank like the Mel filter bank is a physiologically inspired modelling of the human auditory system in which, the response of the basilar membrane is modelled as a gammatone filter bank with impulse response being the product of a gamma distribution function and a sinusoidal tone centred at frequency $f_c$ as represented in equation 4.47, and is illustrated in Fig.4.28.

$$g(t) = K t^{n-1} e^{-2\pi B t} \cos(2\pi f_c t + \varphi) \quad t > 0$$  \hspace{1cm} 4.47

where $K$ is the amplitude factor, $n$ is the filter order, $f_c$ is the centre frequency in Hertz, $\varphi$ is the phase shift and $B$ is the equivalent rectangular bandwidth which represents the duration of the impulse response.

Fig.4.28 Typical time domain response of Gammatone filter bank

The extraction of GTCC is similar to that of MFCC extraction scheme and is depicted as flowchart in Fig.4.29, except that Gammatone filter bank equally spaced in the ERB scale is used instead of triangular filter bank equally spaced in the Mel scale.

Figures 4.30 and 4.31 shows the plots GTCCs of five records of different target noises, namely humpback whale, ship, boat, sealion and snapping shrimp.
Chapter 4 Target Classifier

Fig. 4.29 Steps in computing GTCCs

Fig. 4.30 Plot of GTCCs of Humpback whale noise
Fig.4.31 Plots of GTCCs of different target types - ship, boat, sealion and snapping shrimp

4.11 Feature Selection

The features extracted through the different feature extraction techniques may contain redundant and irrelevant features, which do not, or negatively impact, the performance of the classifier. Such features are identified and removed with the help of feature selection algorithms. Feature selection is a process to identify irrelevant and redundant features that do not contribute to the accuracy of the predictive model, which may be removed to reduce the complexity of the model. The best subset of features contains the least number of dimensions that most contribute to prediction accuracy.

In classifier design, feature selection is an important pre-processing step to avoid noisy, irrelevant and misleading features [152]. Irrelevant,
redundant and noisy features may mislead learning algorithms or cause them to overfit the data. Hence, the obtained classifier in general is less accurate than the one learned from the relevant data. In addition, with the presence of redundant or irrelevant data, it is more likely that the classifier obtained is more complex. A complex classifier tends to be less accurate and un-generalistic compared to a simple classifier. Thus the objective of feature selection is three-fold: to improve the prediction performance of the classifier, to reduce the curse of dimensionality thus reducing the computational burden, and to minimise the chances of overfitting caused by irrelevant data.

Feature selection methods can be broadly categorized into exhaustive search, filter methods, wrapper methods and embedded methods. A brute force feature selection method following an exhaustive search, evaluates all possible combinations of the input features to find the best subset. The computational cost of exhaustive search methods is prohibitively high as its space is $O(2^N)$ and also imposes the danger of overfitting. Hence, exhaustive search techniques are seldom used for feature selection. Filter methods rely on the characteristics of data by applying a statistical measure to assign scoring to each feature. Feature selection is done based on the score obtained by the features. These methods do not consider the effects of the selected feature subset on the performance of the underlying classifier [153]. Wrapper techniques utilize the prediction performance of the underlying classifier to assess the relative usefulness of a feature subset [154]. Embedded methods embed the feature selection with classifier construction, and have the advantages of wrapper models and filter models [153]. They are usually specific to the learning algorithm (classifier).
4.11.1 Filter Methods

Filter methods work on the general characteristics of the data and employ statistical methods such as correlation between the features, and assign scoring to each feature. The features are ranked by their individual scores and are accordingly kept or removed from the feature set. The block diagram for filter method is shown in Figure 4.32.

Correlation based feature selection (CFS) is an effective and efficient filter based approach that uses a correlation based heuristic to evaluate the worthiness of features. The heuristic that forms the core of CFS algorithm to evaluate the merit of a subset of features. Good feature subsets contain features highly correlated within the class, yet uncorrelated with other classes. The feature subsets are then ranked depending on their correlation with members of the same class to denote their usefulness in classification. The $m$ top ranked features are selected to be retained or removed from the feature set. Eigen vector Centrality based feature Selection (ECS) is a filter method which maps the feature selection problem on an affinity graph where features are the nodes and the importance of nodes are assessed and ranked through Eigen Vector Centrality. The central premise of ECS method is to estimate the importance of a feature as a function of the importance of its
neighbours. Ranking central nodes individuates candidate features, which turn out to be effective from a classification point of view [155]. Filter based approaches are not dependent on classifiers and are usually faster and more scalable than wrapper based methods. In addition, they have low computational complexity since measuring information gains, distance, dependence, or consistency is less complex by time measure, than measuring the performance of a classifier [156]. Also, since filter based approaches rely only on the intrinsic properties of data, the selected features can be used to learn different classifiers.

4.11.2 Wrapper Methods

Wrapper methods are so called because; the feature subset selection algorithm exists as a wrapper around the classification algorithm[157]. For selecting a good feature subset, the wrapper methods use the classification algorithm to evaluate different possible feature subsets. The idea behind the wrapper approach is shown in Figure 4.33.

The classification algorithm is considered as a black box and is run on the dataset partitioned into training and validation sets. The validation set is adjudged through an evaluation metric and the feature set with the highest evaluation metric is chosen.
The wrapper approaches conducts a search in the space of possible features. The search can be greedy or heuristic in nature. Greedy search techniques include sequential forward selection (SFS), sequential backward selection (SBS), and bi–directional search (BDS). Sequential feature selection methods, learn the usefulness of each feature at each time–step depending on its performance score. Sequential forward selection (SFS) starts with an empty set, incrementally adding features in each step. As each feature is added in, the classifier is evaluated with the feature set and a new feature is retained, only if the performance metric on which the classifier is evaluated has improved \[158\]. Sequential backward selection (SBS) starts with the full set of features, and removes one of the redundant or irrelevant features at each step. Another method is bi-directional search (BDS), which is a parallel implementation of both SFS and SBS and features are both added and deleted simultaneously until convergence. Sequential techniques have the advantage of picking out features which together work well for classification, as the performance of the classifier is evaluated at each step of the algorithm with different possible combination of features.

Heuristic search based wrapper approaches employ heuristics to search the feature space. These methods do not guarantee the optimal solution but generally arrives at a near optimal solution. They have the advantage of converging faster to near optimal solution but bear the disadvantage of having the tendency of being trapped at a local solution. Meta-heuristic algorithms are a variant of heuristic algorithms, and are strategies which guide the heuristic search towards the optimal solution by not getting trapped at a local solution. Exploitation and exploration – two competing design goals are two key components of a meta-heuristic algorithm and are respectively responsible for local intensification and global diversification. Diversification generates diverse solutions so as to explore
the search space on the global scale, while intensification focuses the search to a local region by exploiting the information that a current good solution is found in this region. The competing design considerations must strike the right balance to derive optimal performance from the algorithm. Different meta-heuristic algorithms employ different degree of exploration and exploitation. A variety of meta-heuristic algorithms such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA), Ant Colony Optimization (ACO) can be used for feature selection. However the most widely employed meta-heuristic algorithm for feature selection is genetic algorithm.

Genetic algorithm (GA) is a meta-heuristic optimization algorithm inspired by the procedures of natural evolution. GA is a population based algorithm, and is capable of effectively exploring large search spaces which is usually required in case of feature selection. Each individual of this population represents a candidate solution to the given problem. Each individual is assigned a fitness score on the basis of a fitness function. GA operates in three stages: selection, crossover and mutation. In the selection step, the best solutions with higher fitness score are selected and are given more chances for reproduction. During the crossover, portions of the parent solutions are exchanged in the hope of generating more adapted solutions. Mutation operates by randomly changing one or more components of a selected individual. Mutation operator introduces diversity in the current solutions, and prevents premature convergence of the algorithm. The population is operated upon by the three GA operators and then re-evaluated until the termination criterion is met. Meta-heuristic search based wrapper methods have the advantage of arriving at near optimal feature subset without an exhaustive search of the feature space thus achieving faster convergence. The trade-off is between optimality and speed and is often
worthwhile because of much gained speed with little loss of optimality in arriving at near optimal solutions.

4.12 Classification – A Machine Learning Approach

In machine learning and statistics, classification can be defined as the problem of identifying to which of a set of categories a previously unseen observation belongs to, on the basis of knowledge gained through a training set of data containing observations whose categories are known. Learning is the act of acquiring, modifying or reinforcing existing knowledge or behaviour through experience, study or teaching; which may lead to a potential change in synthesizing information. Learning is built upon and is shaped by previous experience and knowledge. Machine Learning provides computers and related systems the ability to automatically learn and improve from experience without being explicitly programmed.

Designing a machine learning approach involves a number of design choices, including choosing the type of training experience, the target function to be learned, a representation for this target function, and an algorithm for learning the target function from training examples.

The process of machine learning starts with observations or data. Data can be thought of as a collection of instances having an associated label, which is analysed by the learning algorithm to gain knowledge on the process which generated it, and to identify regularities or patterns within. The performance outcome of the learning algorithm on a particular task, measured by a suitable performance metric, generates experience which adds knowledge on the process which created data. Even though the learning algorithm may not completely understand the underlying process, a reasonable approximation of the process could be successfully carried out
which can detect patterns and regularities within data to do a particular task. The niche of machine learning is that even though we may not gain full knowledge about the underlying process or the system, the little know–how we have about the system which is also based on the experience of the learning algorithm at a particular task, helps us to make predictions on the data. According to Mitchell, a learning algorithm learns from expertise concerning some task and performance measure, if its performance at the task improves with expertise [159]. Thus, a well-defined learning necessitates a well-defined task, performance metric, and feedback on training experience.

Machine learning algorithms typically consist of two phases: a learning phase and testing phase. To implement the above two aspects, the algorithm divides the dataset into training data and testing data. In the learning phase depicted in Figure 4.34, the learning algorithm designs a mathematical model of the dependency, which approximates the relationship between the data and outcome, based on the training data given. In the testing phase depicted in Figure 4.35, the models developed by the learning algorithm in the training phase are used to predict the outcome of the data which has not been previously seen by the algorithm.

![Fig.4.34 Learning phase of machine learning](image)

![Fig.4.35 Testing phase of machine learning](image)
Machine learning algorithms can be broadly classified into two categories: Unsupervised and Supervised learning algorithms. Unsupervised learning is solely based on the correlations among the input data and is used to find the significant patterns or features in the input data without the help of a teacher. The goal of unsupervised learning is to draw inferences about the underlying structure or distribution in the data. Unsupervised learning does not have labelled data and hence does not involve any target values at the training phase. In case of unsupervised learning, a stopping criterion is needed to terminate the learning phase. Without a stopping criterion, a learning process continues even when a pattern, which does not belong to the training patterns set, is presented to the network. Figure 4.36 depicts the general block diagram of unsupervised learning algorithms. A typical example of unsupervised learning is clustering which is used in exploratory data analysis to find similar hidden patterns or groupings in data.

![General block diagram of unsupervised learning algorithm](image)

**Fig.4.36 General block diagram of unsupervised learning algorithm**

Supervised learning algorithms learn a model by inferring a mapping, \( y = f(x) \), between labelled training data ‘\( x \)’ with known response \( y \), which makes it capable to make predictions on future unseen data. For a typical classification task, the classifier predicts the output \( y, y \in \{1, \ldots, C\}, \) where
Chapter 4 Target Classifier

\( C \) is the number of classes in the classification task. Supervised learning is so called because, the process of the algorithm learning from the training dataset can be thought of as a teacher supervising the learning process.

Figure 4.37 shows the general block diagram of supervised learning algorithms. In the optimal scenario, a large number of labelled training data will allow the algorithm to infer well the mapping between the input and the output, which will enable it to determine the class labels of unseen test data correctly. A lot many learning algorithms such as decision trees, logistic regression, and support vector machines belong to the domain of supervised algorithms.

![Fig.4.37 General block diagram of supervised learning algorithm](image)

The machine learning field suggests three phases for the design of a supervised learning algorithm: a training phase, validation phase, and testing phase. Hence, it recommends three divisions (or subsets) of the data sets to carry out these tasks. The training dataset is the sample of data used to build the model by inferring the mapping between the input and the output. The validation dataset is used to determine how well the model has been trained,
the estimate of which is used to adjust its parameters. The test dataset is used
to assess the performance of a fully-specified classifier. In machine learning,
it is also essential to define suitable performance evaluation metrics to train,
validate, and test the learning models. The cause of poor performance in
machine learning can be attributed to concepts such as under-fitting, and
over-fitting. Over-fitting refers to a model that models the training data too
well, however, fails to generalise to new data and under-fitting refers to a
model that can neither model the training data nor generalise to new data.
Over-fitting occurs when the model learns the concepts in the training data
along with the noise present in it, to the extent that it negatively impacts the
performance of the model on previously unseen data. Underfitting occurs
when the model fails to follow the trends in the training data.

An important aspect of machine learning is the bias-variance trade-
off. Reducible error incurred by a learning algorithm, is a combination of the
error due to squared bias and the error due to variance, and the goal of the
designer is to reduce both bias and variance simultaneously. The error due to
squared bias is the value by which the expected model prediction differs from
the real value or target, over the training data. Bias conveys the ability of the
learning model to approximate the data. An overly simplistic model often
leads to a solution that is highly biased and does not fit the data. As the
complexity of the learning model increases, its ability to approximate and
follow the patterns in the data increases, thus keeping the bias low.

Variance refers to the error by which the prediction over one training
set, differs from the expected, predicted value, over all the training sets.
Variance attributes to the error due to an overly-complex model, that tries to
fit the training data as closely as possible, leading to over-fitting. It depends
on the extent of training of the classifier, and decreases with rigorously
trained classifiers with more training data. However, a model having high
variance when tested on unseen data during the testing phase will not yield satisfactory results, thus exhibiting poor generalisation capabilities. Generalization ability of an algorithm is defined as the ability of the algorithm to predict the outcome of previously unseen data accurately. An algorithm with high variance becomes highly sensitive to high degrees of variation in the training data, since an overfit model closely follows the trends of the training data, including the noise in the training data and hence fails to generalise in the test phase.

![Bias-variance trade-off](image)

**Figure 4.38 Depiction of Bias-variance trade-off**

A high variance indicates low training error and high validation error, and high bias implies a high training as well as a high validation error. The designer’s goal is to simultaneously reduce bias and variance to obtain the most accurate model feasible. However, a trade-off has to be made when selecting models of differing flexibility or complexity and is referred to as bias-variance trade-off and is depicted in Figure 4.38. Bias is reduced and...
variance is increased in relation to model complexity. When the complexity of the model is low, the learning algorithm is too flexible to fit the data well and hence has a high bias. As the complexity of the model increases, with more and more parameters being added, bias steadily falls, and variance becomes the primary concern. The bias-variance trade-off suggests that, a learning algorithm should have the flexibility to fit the data well and at the same time be optimally complex for improved generalisation.

**4.13 Support Vector Machines**

Support Vector Machines (SVM) is a supervised learning algorithm that has stemmed from Statistical Learning Theory (SLT). It is a kernel-based machine learning algorithm which is used for both classification and regression. SVM is based on the Structural Risk Minimization (SRM) induction principle that minimises an upper bound on the expected risk, thus reducing generalisation error. The issues with machine learning algorithms that steered the development of SVMs are the bias-variance trade-off, capacity control and overfitting.

The formulation and constitution of algorithms to overcome the above disadvantages using concepts from SLT has led to the development of SVMs. SVMs have strong generalisation abilities and have advantages in selecting a model with the optimum complexity, and overcoming problems such as overfitting. A detailed description of SLT which has led to the development of SVM based on SRM induction principle has been provided in Appendix. To implement the SRM induction principle in learning algorithms, one has to minimise the value of empirical risk and the capacity factor, to choose a model with appropriate VC dimension. The modelling is achieved in support vector machine by employing a maximal margin
classifier and for non-linearly separable data, the data is mapped to a high dimensional space where they are linearly separable.

### 4.13.1 Separating Hyperplanes & their Generalization

Consider the problem of minimising empirical risk on the set of linear indicator functions described by equation 4.48.

\[
f(x, w) = \theta \{ \sum_{i=0}^{n} w_i x_i^t \}, \quad w \in W
\]

Let the training set be described as \( Zp, o \{ , o \{ , ..., Zp, o \{ \} \} \) where each \( x_j = (x_j^1, ......., x_j^l) \) is a vector belonging to the class \( y_j \in \{0,1\}, \ j = 1, ..., ..., n \). To minimise the empirical risk, one has to find the parameters \( w = (w^1, ......., w^n) \) (weights) which minimise the empirical risk functional, given by equation 4.49.

\[
R_{emp}(w) = \frac{1}{l} \sum_{j=1}^{l} \left(y_j - f(x_j, w)\right)^2
\]

There are several methods for minimising this risk functional. An exact solution can be ensured if the minimum of the empirical risk is zero. A non zero value for the minimum of empirical risk functional will yield only an approximate solution. Therefore, by controlling the weights of the hyperplane, one can control the value of empirical risk. Unfortunately, the set of hyperplanes defined by different weights may not be flexible enough to provide low empirical risk for many real-world problems. However, the flexibility can be increased in one of the following ways [160],

i. selecting a hyperplane with the largest margin
ii. map the input vectors to a higher dimensional feature space which transforms a nonlinear classification problem to a linear one

The above idea has led to the development of Support Vector Machines.
4.13.2 Optimal Separating Hyperplane

Consider a linearly separable training data
\[(x_1, y_1), \ldots, (x_n, y_n), \ x \in \mathbb{R}^n, \ y \in \{+1, -1\}\]
which can be separated by the set of hyperplanes described by equation 4.51
\[(w^T x) + b = 0\]
where \(b\) is the bias and \(w = [w_1, w_2, \ldots, w_l]\) is the weight vector normal to the hyperplane.

The choice of the hyperplane from the set of hyperplanes described by equation 4.52 should be done in such a way that the resulting classifier can generalise well. The most optimal hyperplane would be the one with the most significant margin and is also known as the maximal margin hyperplane. In other words, maximal margin hyperplane is the hyperplane that has the farthest minimum distance to the closest training vector. A hyperplane with the most substantial margin on the training data can be expected to have a significant margin on the test data, and hence will be able to generalise well on the test data.

4.13.3 Linear SVM

For a linearly separable binary classification problem, the linear separating hyperplane can be described as follows
\[(w^T x_i) + b > 0 \text{ if } y_i = 1\]
\[(w^T x_i) + b < 0 \text{ if } y_i = 0\]
Applying scale transformation on \(w\) and \(b\), equations 4.53 and 4.54 are equivalent to
\[(w^T x_i) + b \geq 1 \text{ if } y_i = 1\]
Chapter 4 Target Classifier

\[(w^T.x_i) + b \leq -1 \text{ if } y_i = -1 \quad 4.56\]

which can be alternatively written as

\[y_i[(w^T.x_i) + b] \geq 1, \quad i = 1, \ldots, n \quad 4.57\]

The margin \(M\), which is the distance from the hyperplane to the closest data point can be derived as

\[M = \frac{2}{||w||} \quad 4.58\]

The support vectors can be defined as those data points the margin pushes against and which satisfies the equality in equation 4.57 [92]. Only, these data points influence in determining the position of the hyperplane. This is illustrated in Figure 4.39.

Fig.4.39 Maximal margin linear classifier

For obtaining a maximal margin classifier, the margin \(M\) described by equation 4.58, has to be maximised which is equivalent to minimising \(\frac{||w||^2}{2}\), subject to the constraint expressed by equation 4.57. Thus we have a quadratic optimization problem subject to constraints specified by a linear
inequality. The solution to this optimization problem is given by the saddle point of the Lagrange functional (Langrangian). Introducing Lagrange multipliers, $\alpha_i \geq 0, i = 1, \ldots, n$, the Lagrangian for the above optimization problem can be formulated as

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i (w^T x_i + b) - 1) \quad 4.59$$

The objective is to find the saddle point of the above Lagrangian which is achieved through its dual formulation. The Karush-Kuhn-Tucker (KKT) conditions specify the requirements to be satisfied by an optimal solution to a general optimisation problem [161]. Given, a primal problem defined by equation 4.59, KKT conditions require, $\mathcal{L}(w, b, \alpha)$ to be minimised with respect to $w$, $b$ and maximized over the Lagrangian multipliers, $\alpha_i$.

$$\frac{\partial}{\partial b} \mathcal{L}(w, b, \alpha) = 0 \quad 4.60$$

$$\frac{\partial}{\partial w} \mathcal{L}(w, b, \alpha) = 0 \quad 4.61$$

Equations 4.60 and 4.61 gives,

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad 4.62$$

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \quad 4.63$$

substituting equations 4.62 and 4.63 into equation 4.59, the dual Lagrangian can be formulated as

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \quad 4.64$$

The dual optimisation problem is to maximise the dual Lagrangian $L_D$ subject to
\[ \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0 \quad 4.65 \]

The dual optimisation problem can be solved using standard programming techniques such as sequential minimal optimisation (SMO) by which we get the coefficients \( \alpha_i, i = 1, \ldots, n \) from which we can calculate

\[ w = \sum_{i=1}^{n} \alpha_i y_i x_i \quad 4.66 \]

and the classifier can be expressed as

\[ f(x_{\text{new}}) = \text{sgn}(\sum_{i=1}^{n} \alpha_i y_i (x, x_i) + b) \quad 4.67 \]

### 4.13.4 Soft Margin Classifier

In most practical applications, training data would be linearly inseparable. Considering the case when the data is almost linearly separable except for some outliers as shown in Figure 4.40.

![Soft Margin Classifier](image)

Fig.4.40 Soft Margin Classifier when data is almost linearly separable except for outliers

In such cases, opting a hard-margin classifier as in Figure 4.41 will lead to poor generalisation and hence the hard-margin SVM needs to be modified so that it can cater for the misclassification.
A soft-margin classifier which introduces slack variables in the objective function, that relax the constraint in equation 4.57 is employed. The slack variables define the cost at which each outlier can be moved to its original position. The overall goal of the optimisation is now to find the hyperplane, such that the number of misclassified instances is minimised. The relaxed constraints with slack variables now becomes

\[
w x_i + b \geq 1 - \xi_i, \quad y_i = +1 \quad 4.68
\]
\[
w x_i + b \leq -1 - \xi_i, \quad y_i = -1 \quad 4.69
\]

and the objective function takes the form

\[
\min \frac{1}{2} w^T w + C \sum_{i=1}^n \xi_i \quad 4.70
\]

such that

\[
y_i (w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n \quad 4.71
\]

A penalty parameter C is included in the objective function described by equation 4.69, which controls the width of the soft margin and also determines the trade-off between the training error and the VC dimension of the model [92]. Introducing Lagrange multipliers, \( \alpha_i, \beta_i \geq 0, \quad i = 1, \ldots, n \), the Lagrangian for the above optimization problem can be formulated as
Chapter 4 Target Classifier

\[ L(w, b, \xi, \alpha) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i (y_i (w^T x_i + b) - 1 + \xi_i) - \sum_{i=1}^{n} \beta_i \xi_i \]  

4.72

The objective is to find the saddle point of the above Lagrangian which is achieved through its dual formulation. For the primal problem specified by equation 4.72, KKT conditions require, \( L(w, b, \alpha) \) to be minimised with respect to \( w, b, \xi \) and maximized over the Lagrangian multipliers, \( \alpha_i \).

\[ \frac{\partial}{\partial b} L(w, b, \xi, \alpha) = 0 \]  

4.73

\[ \frac{\partial}{\partial w} L(w, b, \xi, \alpha) = 0 \]  

4.74

\[ \frac{\partial}{\partial \xi} L(w, b, \xi, \alpha) = 0 \]  

4.75

Equations 4.73, 4.74 and 4.75 gives,

\[ \sum_{i=1}^{n} \alpha y_i = 0 \]  

4.76

\[ w = \sum_{i=1}^{n} \alpha_i y_i x_i \]  

4.77

\[ \alpha_i = C - \beta_i \]  

4.78

substituting equations 4.76, 4.77, and 4.78 into equation 4.72 the dual Lagrangian can be formulated as

\[ L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i x_j \rangle \]  

4.79

The dual optimisation problem is to maximise the dual Lagrangian \( L_D \) subject to

\[ \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad 0 \leq \alpha_i \leq C \]  

4.80
4.13.5 Non-Linear SVM

The soft margin approach works well for data which is close to linearly separable. However, when the data is not linearly separable, and a non-linear decision surface becomes mandatory, SVM’s map the input data $x_i \in \mathbb{R}^m$ into vectors $\phi(x_i) \in \mathbb{R}^s$ of a higher dimensional space where it can be linearly separated.

$$x \in \mathbb{R}^m \rightarrow \phi(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_s(x)]^T \in \mathbb{R}^s$$  

where $\phi$ represents the mapping: $\mathbb{R}^m \rightarrow \mathbb{R}^s$

Fig. 4.42 Kernel trick by which non-linearly separable data is transformed to linearly separable data in a higher dimensional space
SVM’s, employ an easy and efficient way of mapping the data to a higher dimensional space, which is often referred to as the ‘kernel trick’. By equation 4.64 the linear SVM classifier depends on the inner product \((x_i, x_j)\) between the data point vectors. When the data points are transferred to the high dimensional feature space where a linear classifier can be employed, via a mapping function \(\phi\) the inner product becomes \((\phi(x_i)^T \phi(x_j))\). This is as depicted in Figure 4.42.

To define the non-linear SVM classifier in the dual formulation, it is sufficient to know the inner product \((\phi(x_i)^T \phi(x_j))\) and the explicit mapping function need not be known. The inner product in equation 4.64 has to be replaced by the inner product \((\phi(x_i)^T \phi(x_j))\) to define the non-linear SVM. The inner product \((\phi(x_i)^T \phi(x_j))\) is called the kernel function and is denoted as

\[
K(x_i, x_j) = (\phi(x_i)^T \phi(x_j))
\]

4.82

The non-linear SVM classifier can be expressed as

\[
f(x_{new}) = \text{sgn}(\sum_{i=1}^{m} \alpha y_i K(x_i, x_j) + b)
\]

4.83

The kernel function can be defined as a function that corresponds to a dot product of two vectors in some expanded feature space. In SVM formulation, using the kernel function, data in the original input space can be easily carried to a higher dimensional feature space. For a given function \(K(x_i, x_j)\) to be a kernel (i.e. a dot product in an expanded feature space), the function must satisfy Mercer’s conditions [93]. Mercer’s theorem states that for \(K\) to be a valid kernel function, the kernel matrix \(K\) must be symmetric and positive semi-definite. By Mercer’s theorem, a symmetric function \(K(x_i, x_j)\) can be expressed as a kernel

\[
K(x_i, x_j) = (\phi(x_i)^T \phi(x_j))
\]

4.84
for some $\phi$ if and only if $K(x_i, x_j)$ is positive semidefinite i.e.

$$\int K(x, y)g(x)g(y)dxdy \geq 0 \quad \forall g$$

or, equivalently

$$\begin{bmatrix}
K(x_1, x_1) & K(x_1, x_2) & \cdots & \cdots \\
K(x_2, x_1) & \ddots & \ddots & \ddots \\
\vdots & \ddots & \ddots & \ddots \\
K(x_n, x_1) & \cdots & \cdots & \ddots \\
\end{bmatrix}$$

is positive semidefinite for any $\{x_1, \ldots, \ldots, x_n\}$

The kernels often investigated for the pattern recognition problem that are suitable for most common settings are listed in Table 4.1.

The linear kernel results in a classifier which has a linear decision surface and the polynomial kernel results in a polynomial decision surface. The Gaussian kernel results in a Gaussian RBF classifier and the multilayer perceptron kernel results in a sigmoidal network.

**Table 4-1 Commonly used Kernels**

<table>
<thead>
<tr>
<th>Type of Kernel</th>
<th>Inner Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Kernel</td>
<td>$K(x_i, x_j) = x_i^T x_j$</td>
</tr>
<tr>
<td>Polynomial Kernel</td>
<td>$K(x_i, x_j) = (1 + x_i^T x_j)^p$</td>
</tr>
<tr>
<td>Gaussian (Radial Basis Function (RBF)) Kernel</td>
<td>$K(x_i, x_j) = \exp\left(-\frac{|x_i - x_j|^2}{2\sigma^2}\right)$</td>
</tr>
<tr>
<td>Multi Layer Perceptron (MLP)</td>
<td>$K(x_i, x_j) = \tanh(p_1 x_i x_j + p_2)$</td>
</tr>
</tbody>
</table>

**4.14 Multiclass SVM**

SVM’s are originally defined as binary classifiers. The binary SVM classifiers are extended by different algorithms to solve the multiclass problem. Algorithms for solving multiclass problems are built upon the
Chapter 4 Target Classifier

binary classifier by reducing the multiclass problem to multiple binary classification problems. Two conventional approaches adopted for multiclass SVM classification are one-against-one (1-a-1) and one-against-all (1-a-a) approach.

Knerr et al. proposed the one-against-one classifier [94] as a stepwise building procedure with single layer training, as an alternative to multilayer neural networks. The algorithm was later adopted to solve multiclass SVM problems. The algorithm, also referred to as pairwise classification algorithm, works by creating SVM classifiers for all possible pair of classes as depicted in Figure 4.43. Each unseen example is classified to the class that ‘wins’ most binary classifications, in the sense that one which gets the highest number of votes. A voting scheme is employed in which each class gets a vote when the classifier assigns a particular instance to the class.

Fig 4.43 One-against-one approach for multiclass problems

For a $K$-class problem, the one-against-one algorithm creates $K(K-1)/2$ SVM classifiers. When an unseen example is classified, all SVM’s are evaluated, and the unseen example is classified to the class that wins, i.e. the one with the most votes. The advantage of the one-against-one classifier is that very complex decision boundaries can be realised and an unseen example previously misclassified by one binary SVM, still has a chance of
being correctly classified as there are $K-1$ binary models per class. The distinct disadvantage with this approach is that when the number of classes, $K$, is considerable, the number of binary SVMs required, $K(K-1)/2$, will become exorbitantly large which results in a slower system.

![Diagram of one-against-one approach for multiclass problems](image)

**Fig. 4.44** One-against-one approach for multiclass problems

The one-against-all classifier proposed by Vapnik [95] constructs $K$ separate binary classifiers for $K$-class classification as in Figure 4.44. Each classifier attempts to build a decision boundary between itself and the rest. The $n$-th binary classifier is trained using the data from the $n$-th class as positive instance and the remaining $K-1$ classes as negative instances. During the testing phase, a particular example is assigned to a particular class in which the performance metric of the binary classifier gives the maximum output value. The one-against-all approach is faster than the one-against-one approach as only $K$ classifier models need to be built. However, complex decision boundaries cannot be realised with one-against-all as with one-against-one approach.
4.15 Results and Discussions

A non-linear multiclass SVM classifier using 1-a-a approach was developed to classify 11 classes of underwater acoustic targets. The targets include the noises of a humpback whale, noises of 4 ships (ship\(_1\), ship\(_2\), ship\(_3\), ship\(_4\)), sounds of 4 boats (boat\(_1\), boat\(_2\), boat\(_3\), boat\(_4\)), sound of sea lion and noise of snapping shrimps. To select the best features for classification, the performance of the classifier was tested with different elements and is discussed in the following sections. The range of parameters of different kernels, such as polynomial order, MLP parameter values, RBF sigma value, chosen for the experiments, was by trial and error method, in which the parameter values which exhibited better performance over others are tabulated.

4.15.1 Performance with time domain and frequency domain features

The time domain and frequency domain features were combined into a feature vector with which the classifier was evaluated. The feature vector is formed by combining the average values of energy, energy entropy and spectral entropy as well as ten frame values, each, of the spectral centroid, spectral roll-off and spectral flux leading to a total length of 33. The rationale behind incorporating the average values of energy, energy entropy and spectral entropy is that, they do not change much over time, as can be observed in Figures 4.5, 4.6 and 4.10.

The performance of a non-linear multiclass SVM based classifier to classify eleven classes of acoustic targets with the feature vector formed from time and frequency domain features are measured. The different kernels used are polynomial, multilayer perceptron (MLP) and radial basis function (RBF). The average value of F-scores of 11-classes and the overall accuracy,
obtained with time and frequency features for kernel parameters such as, for different values of polynomial order \( p \) in case of the polynomial kernel, different values of MLP parameters \( (p_1, p_2) \) in case of MLP kernel and different values of sigma \( (\sigma) \) in case of RBF kernel are listed in Table 4-2, Table 4-3 and Table 4-4.

Table 4-2 Performance obtained with time domain and frequency domain features with polynomial kernel for different polynomial orders

<table>
<thead>
<tr>
<th>Polynomial order</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.57</td>
<td>50.9</td>
</tr>
<tr>
<td>3</td>
<td>0.58</td>
<td>52.7</td>
</tr>
<tr>
<td>4</td>
<td>0.62</td>
<td>54.9</td>
</tr>
<tr>
<td>5</td>
<td>0.60</td>
<td>53.4</td>
</tr>
<tr>
<td>6</td>
<td>0.59</td>
<td>52.3</td>
</tr>
<tr>
<td>7</td>
<td>0.57</td>
<td>51.2</td>
</tr>
</tbody>
</table>

Table 4-3 Performance obtained with time domain and frequency domain features with MLP kernel for different values of MLP parameters

<table>
<thead>
<tr>
<th>[( p_1, p_2 )] values</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0.001,-0.001]</td>
<td>0.42</td>
<td>38.0</td>
</tr>
<tr>
<td>[0.002,-0.002]</td>
<td>0.47</td>
<td>40.7</td>
</tr>
<tr>
<td>[0.003,-0.003]</td>
<td>0.48</td>
<td>42.9</td>
</tr>
<tr>
<td>[0.004,-0.004]</td>
<td>0.46</td>
<td>41.1</td>
</tr>
<tr>
<td>[0.005,-0.005]</td>
<td>0.46</td>
<td>40.3</td>
</tr>
</tbody>
</table>
Chapter 4 Target Classifier

Table 4-4 Performance obtained with time domain and frequency domain features with RBF kernel for different values of RBF sigma

<table>
<thead>
<tr>
<th>RBF Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>σ value</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

4.15.2 Performance with LPCs

The performance of a non-linear multiclass SVM based classifier to classify eleven classes of acoustic targets with the feature vector formed from LPCs are evaluated. The appropriate feature vector length was found out by trial and error. The performance results, the average F-score of eleven classes and accuracy, with different feature vector length are tabulated in Table 4-5.

Table 4-5 Classification performance obtained with LPC features of different length

<table>
<thead>
<tr>
<th>Vector Size</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.48</td>
<td>42.9</td>
</tr>
<tr>
<td>20</td>
<td>0.49</td>
<td>49.1</td>
</tr>
<tr>
<td>30</td>
<td>0.49</td>
<td>50.9</td>
</tr>
<tr>
<td>40</td>
<td>0.69</td>
<td>63.3</td>
</tr>
<tr>
<td>50</td>
<td>0.59</td>
<td>58.9</td>
</tr>
<tr>
<td>60</td>
<td>0.61</td>
<td>56.0</td>
</tr>
</tbody>
</table>

Kernel – RBF, RBF Sigma Value - 4
It can be inferred that, the best feature vector length of LPC for the given classification task is 40. Hence, a feature vector of length 40 was used for evaluating the classification performance. The different kernels used are polynomial, multilayer perceptron (MLP) and radial basis function (RBF).

The average value of F-scores of 11-classes and the overall accuracy, obtained with LPC for kernel parameters such as, different values of polynomial order ($p$) in case of polynomial kernel Table 4-6, different values of MLP parameters ($p_1, p_2$) in case of MLP kernel Table 4-7 and different values of sigma ($\sigma$) in case of RBF kernel are listed in Table 4-6.

<table>
<thead>
<tr>
<th>Polynomial Kernel</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial order</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.62</td>
<td>55.6</td>
</tr>
<tr>
<td>3</td>
<td>0.62</td>
<td>56.0</td>
</tr>
<tr>
<td>4</td>
<td>0.63</td>
<td>57.1</td>
</tr>
<tr>
<td>5</td>
<td>0.64</td>
<td>58.2</td>
</tr>
<tr>
<td>6</td>
<td>0.63</td>
<td>57.4</td>
</tr>
<tr>
<td>7</td>
<td>0.61</td>
<td>56.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MLP Kernel</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[p_1, p_2] values</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[0.001, 0.001]</td>
<td>0.46</td>
<td>41.4</td>
</tr>
<tr>
<td>[0.002, 0.002]</td>
<td>0.51</td>
<td>44.7</td>
</tr>
<tr>
<td>[0.003, 0.003]</td>
<td>0.47</td>
<td>43.6</td>
</tr>
</tbody>
</table>
Table 4-8 Performance obtained with LPCs with RBF kernel for different values of RBF sigma

<table>
<thead>
<tr>
<th>RBF Kernel</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.65</td>
<td>60.5</td>
</tr>
<tr>
<td>2</td>
<td>0.67</td>
<td>61.8</td>
</tr>
<tr>
<td>3</td>
<td>0.68</td>
<td>62.6</td>
</tr>
<tr>
<td>4</td>
<td>0.69</td>
<td>63.3</td>
</tr>
<tr>
<td>5</td>
<td>0.67</td>
<td>62.5</td>
</tr>
<tr>
<td>6</td>
<td>0.67</td>
<td>61.0</td>
</tr>
<tr>
<td>7</td>
<td>0.66</td>
<td>59.0</td>
</tr>
</tbody>
</table>

4.15.3 Performance with LPCCs

The performance of a non-linear multiclass SVM based classifier to classify eleven classes of acoustic targets with feature vector formed from LPCCs, is evaluated. The appropriate feature vector length was found out by trial and error. The performance results, the average F-score of eleven classes and accuracy, with different feature vector length are tabulated in Table 4-9. From Table 4-9, it can be inferred that, the best feature vector length of LPCCs for the given classification task is 40. Hence, a feature vector of length 40 was used for evaluating the performance of the proposed SVM based target classifier. The different kernels used are polynomial, multilayer perceptron (MLP) and radial basis function (RBF).
Table 4-9 Classification performance obtained with LPCCs features of different length

<table>
<thead>
<tr>
<th>Vector Size</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.52</td>
<td>47.6</td>
</tr>
<tr>
<td>20</td>
<td>0.59</td>
<td>52.7</td>
</tr>
<tr>
<td>30</td>
<td>0.66</td>
<td>60.7</td>
</tr>
<tr>
<td>40</td>
<td>0.71</td>
<td>65.4</td>
</tr>
<tr>
<td>50</td>
<td>0.63</td>
<td>57.4</td>
</tr>
<tr>
<td>60</td>
<td>0.58</td>
<td>51.6</td>
</tr>
</tbody>
</table>

Kernel – RBF, RBF Sigma Value - 3

The average value of F-scores of 11-classes and the overall accuracy, obtained with LPCC for kernel parameters such as, different values of polynomial order \( p \) in case of polynomial kernel, different values of MLP parameters \( p_1, p_2 \) in case of MLP kernel and different values of sigma \( \sigma \) in case of RBF kernel are listed in Table 4-10, Table 4-11 and Table 4-12.

Table 4-10 Performance obtained with LPCCs with polynomial kernel for different polynomial orders

<table>
<thead>
<tr>
<th>Polynomial order</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.64</td>
<td>58.5</td>
</tr>
<tr>
<td>3</td>
<td>0.65</td>
<td>59.6</td>
</tr>
<tr>
<td>4</td>
<td>0.66</td>
<td>60.3</td>
</tr>
<tr>
<td>5</td>
<td>0.64</td>
<td>59.2</td>
</tr>
<tr>
<td>6</td>
<td>0.64</td>
<td>58.2</td>
</tr>
<tr>
<td>7</td>
<td>0.64</td>
<td>57.4</td>
</tr>
</tbody>
</table>
Table 4-11 Performance obtained with LPCCs with MLP kernel for different values of MLP parameters

<table>
<thead>
<tr>
<th>MLP Kernel</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{0.001}, p_{-0.001}$</td>
<td>0.48</td>
<td>42.5</td>
</tr>
<tr>
<td>$w_{0.002}, p_{-0.002}$</td>
<td>0.49</td>
<td>44.0</td>
</tr>
<tr>
<td>$w_{0.003}, p_{-0.003}$</td>
<td>0.55</td>
<td>49.4</td>
</tr>
<tr>
<td>$w_{0.004}, p_{-0.004}$</td>
<td>0.51</td>
<td>45.4</td>
</tr>
<tr>
<td>$w_{0.005}, p_{-0.005}$</td>
<td>0.50</td>
<td>44.3</td>
</tr>
</tbody>
</table>

Table 4-12 Performance obtained with LPCC with RBF kernel for different values of RBF sigma

<table>
<thead>
<tr>
<th>RBF Kernel</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma = 1$</td>
<td>0.68</td>
<td>61.8</td>
</tr>
<tr>
<td>$\sigma = 2$</td>
<td>0.69</td>
<td>63.2</td>
</tr>
<tr>
<td>$\sigma = 3$</td>
<td>0.71</td>
<td>65.4</td>
</tr>
<tr>
<td>$\sigma = 4$</td>
<td>0.68</td>
<td>62.6</td>
</tr>
<tr>
<td>$\sigma = 5$</td>
<td>0.67</td>
<td>61.2</td>
</tr>
<tr>
<td>$\sigma = 6$</td>
<td>0.66</td>
<td>60.5</td>
</tr>
<tr>
<td>$\sigma = 7$</td>
<td>0.65</td>
<td>59.2</td>
</tr>
</tbody>
</table>

4.15.4 Performance with PLP Cepstral coefficients

The performance of a non-linear multiclass SVM based classifier to classify eleven classes of acoustic targets with the feature vector formed from
PLP cepstral coefficients are evaluated. The appropriate feature vector length was found out by trial and error. The performance results, the average F-score of eleven classes and accuracy, with different feature vector length are tabulated in Table 4-13.

Table 4-13 Classification performance obtained with PLP cepstral coefficients of different length

<table>
<thead>
<tr>
<th>Vector Size</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.66</td>
<td>61.0</td>
</tr>
<tr>
<td>15</td>
<td>0.69</td>
<td>63.6</td>
</tr>
<tr>
<td>20</td>
<td>0.71</td>
<td>65.8</td>
</tr>
<tr>
<td>25</td>
<td>0.68</td>
<td>62.0</td>
</tr>
<tr>
<td>30</td>
<td>0.65</td>
<td>59.0</td>
</tr>
<tr>
<td>35</td>
<td>0.62</td>
<td>56.0</td>
</tr>
</tbody>
</table>

Kernel – RBF, RBF Sigma Value - 4

From Table 4-13, it can be inferred that, the best feature vector length of PLP cepstral coefficients for the given classification task is 20. Hence, a feature vector of length 20 was used for evaluating the performance of the proposed SVM based target classifier. The different kernels used are polynomial, multilayer perceptron (MLP) and radial basis function (RBF). The average value of F-scores of 11-classes and the overall accuracy, obtained with PLP cepstral coefficients for kernel parameters such as, different values of polynomial order \(p\) in case of polynomial kernel, different values of MLP parameters \(p_1, p_2\) in case of MLP kernel and different values of sigma \(\sigma\) in case of RBF kernel are listed in Table 4-14, Table 4-15 and Table 4-16.
### Table 4-14 Performance obtained with PLP cepstral coefficients with polynomial kernel for different polynomial orders

<table>
<thead>
<tr>
<th>Polynomial order</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.65</td>
<td>59.2</td>
</tr>
<tr>
<td>3</td>
<td>0.67</td>
<td>61.1</td>
</tr>
<tr>
<td>4</td>
<td>0.69</td>
<td>62.9</td>
</tr>
<tr>
<td>5</td>
<td>0.67</td>
<td>63.2</td>
</tr>
<tr>
<td>6</td>
<td>0.67</td>
<td>62.2</td>
</tr>
<tr>
<td>7</td>
<td>0.66</td>
<td>61.4</td>
</tr>
</tbody>
</table>

### Table 4-15 Performance obtained with PLP cepstral coefficients with MLP kernel for different values of MLP parameters

<table>
<thead>
<tr>
<th>[p1,p2] values</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0.001,-0.001]</td>
<td>0.49</td>
<td>42.9</td>
</tr>
<tr>
<td>[0.002,-0.002]</td>
<td>0.50</td>
<td>44.3</td>
</tr>
<tr>
<td>[0.003,-0.003]</td>
<td>0.53</td>
<td>48.7</td>
</tr>
<tr>
<td>[0.004,-0.004]</td>
<td>0.58</td>
<td>52.7</td>
</tr>
<tr>
<td>[0.005,-0.005]</td>
<td>0.56</td>
<td>50.2</td>
</tr>
</tbody>
</table>

### Table 4-16 Performance obtained with PLP cepstral coefficients with RBF kernel for different values of RBF sigma

<table>
<thead>
<tr>
<th>σ value</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.67</td>
<td>62.5</td>
</tr>
<tr>
<td>2</td>
<td>0.69</td>
<td>63.9</td>
</tr>
<tr>
<td>3</td>
<td>0.68</td>
<td>64.0</td>
</tr>
</tbody>
</table>
The performance of a non-linear multiclass SVM based classifier to classify eleven classes of acoustic targets with the feature vector formed from MFCCs are evaluated. The appropriate feature vector length was found out by trial and error. The performance results, the average F-score of eleven classes and accuracy, with different feature vector length are tabulated in Table 4-17.

**Table 4-17 Classification performance obtained with MFCCs of different length**

<table>
<thead>
<tr>
<th>Vector Size</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.65</td>
<td>59.2</td>
</tr>
<tr>
<td>15</td>
<td>0.67</td>
<td>62.1</td>
</tr>
<tr>
<td>20</td>
<td>0.69</td>
<td>64.0</td>
</tr>
<tr>
<td>25</td>
<td>0.73</td>
<td>68.7</td>
</tr>
<tr>
<td>30</td>
<td>0.70</td>
<td>65.0</td>
</tr>
<tr>
<td>35</td>
<td>0.68</td>
<td>62.0</td>
</tr>
</tbody>
</table>

From Table 4-17, it can be inferred that, the best feature vector length of MFCCs for the given classification task is 25. Hence, a feature vector of length 25 was used for evaluating the performance of the proposed SVM based target classifier. The different kernels used are polynomial, multilayer
Chapter 4 Target Classifier

perceptron (MLP) and radial basis function (RBF). The average value of F-scores of 11-classes and the overall accuracy, obtained with MFCCs for kernel parameters such as, different values of polynomial order \( p \) in case of polynomial kernel, different values of MLP parameters \( p_1, p_2 \) in case of MLP kernel and different values of sigma \( \sigma \) in case of RBF kernel are listed in Table 4-18, Table 4-19 and Table 4-20.

Table 4-18 Performance obtained with MFCCs with polynomial kernel for different polynomial orders

<table>
<thead>
<tr>
<th>Polynomial order</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.68</td>
<td>63.6</td>
</tr>
<tr>
<td>3</td>
<td>0.69</td>
<td>64.3</td>
</tr>
<tr>
<td>4</td>
<td>0.71</td>
<td>66.2</td>
</tr>
<tr>
<td>5</td>
<td>0.69</td>
<td>63.6</td>
</tr>
<tr>
<td>6</td>
<td>0.68</td>
<td>62.5</td>
</tr>
<tr>
<td>7</td>
<td>0.67</td>
<td>61.8</td>
</tr>
</tbody>
</table>

Table 4-19 Performance obtained with MFCCs with MLP kernel for different values of MLP parameters

<table>
<thead>
<tr>
<th>([p_1, p_2]) values</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>([0.001,-0.001])</td>
<td>0.61</td>
<td>55.6</td>
</tr>
<tr>
<td>([0.002,-0.002])</td>
<td>0.62</td>
<td>56.7</td>
</tr>
<tr>
<td>([0.003,-0.003])</td>
<td>0.59</td>
<td>52.7</td>
</tr>
<tr>
<td>([0.004,-0.004])</td>
<td>0.55</td>
<td>49.1</td>
</tr>
<tr>
<td>([0.005,-0.005])</td>
<td>0.53</td>
<td>47.2</td>
</tr>
</tbody>
</table>
Table 4-20 Performance obtained with MFCCs with RBF kernel for different values of RBF sigma

<table>
<thead>
<tr>
<th>σ value</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.71</td>
<td>63.2</td>
</tr>
<tr>
<td>2</td>
<td>0.72</td>
<td>65.5</td>
</tr>
<tr>
<td>3</td>
<td>0.73</td>
<td>66.8</td>
</tr>
<tr>
<td>4</td>
<td>0.74</td>
<td>67.8</td>
</tr>
<tr>
<td>5</td>
<td>0.68</td>
<td>64.1</td>
</tr>
<tr>
<td>6</td>
<td>0.68</td>
<td>63.2</td>
</tr>
<tr>
<td>7</td>
<td>0.67</td>
<td>60.3</td>
</tr>
</tbody>
</table>

4.15.6 Performance with GTCCs

The performance of a non-linear multiclass SVM based classifier to classify eleven classes of acoustic targets with the feature vector formed from GTCCs are evaluated. The appropriate feature vector length was found out by trial and error. The performance results, the average F-score of eleven classes and accuracy, with different feature vector length are tabulated in Table 4-21.

From Table 4-21, it can be inferred that, the best feature vector length of GTCCs for the given classification task is 25. Hence, a feature vector of length 25 was used for evaluating the performance of the proposed SVM based target classifier. The different kernels used are polynomial, multilayer perceptron (MLP) and radial basis function (RBF). The average value of F-scores of 11-classes and the overall accuracy, obtained with GTCCs for
kernel parameters such as, different values of polynomial order \( (p) \) in case of polynomial kernel, different values of MLP parameters \( (p_1, p_2) \) in case of MLP kernel and different values of sigma \( (\sigma) \) in case of RBF kernel are listed in Table 4-22, Table 4-23 and Table 4-24.

### Table 4-21 Classification performance obtained with GTCCs of different length

<table>
<thead>
<tr>
<th>Vector Size</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.64</td>
<td>59.2</td>
</tr>
<tr>
<td>15</td>
<td>0.67</td>
<td>62.5</td>
</tr>
<tr>
<td>20</td>
<td>0.69</td>
<td>64.0</td>
</tr>
<tr>
<td>25</td>
<td>0.73</td>
<td>67.8</td>
</tr>
<tr>
<td>30</td>
<td>0.69</td>
<td>65.1</td>
</tr>
<tr>
<td>35</td>
<td>0.68</td>
<td>62.9</td>
</tr>
</tbody>
</table>

*Kernel – RBF, RBF Sigma Value - 4*

### Table 4-22 Performance obtained with GTCCs with polynomial kernel for different polynomial orders

<table>
<thead>
<tr>
<th>Polynomial Kernel</th>
<th>Polynomial order</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial order</td>
<td>2</td>
<td>0.66</td>
<td>60.3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.68</td>
<td>62.9</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.69</td>
<td>64.7</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.68</td>
<td>63.2</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.67</td>
<td>62.2</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.67</td>
<td>61.4</td>
</tr>
</tbody>
</table>
Table 4-23 Performance obtained with GTCCs with MLP kernel for different values of MLP parameters

<table>
<thead>
<tr>
<th>[ρ₁, ρ₂] values</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0.001, -0.001]</td>
<td>0.55</td>
<td>49.8</td>
</tr>
<tr>
<td>[0.002, -0.002]</td>
<td>0.58</td>
<td>52.0</td>
</tr>
<tr>
<td>[0.003, -0.003]</td>
<td>0.62</td>
<td>55.6</td>
</tr>
<tr>
<td>[0.004, -0.004]</td>
<td>0.59</td>
<td>53.1</td>
</tr>
<tr>
<td>[0.005, -0.005]</td>
<td>0.55</td>
<td>48.7</td>
</tr>
</tbody>
</table>

Table 4-24 Performance obtained with GTCCs with RBF kernel for different values of RBF sigma

<table>
<thead>
<tr>
<th>σ value</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.68</td>
<td>63.2</td>
</tr>
<tr>
<td>2</td>
<td>0.70</td>
<td>65.4</td>
</tr>
<tr>
<td>3</td>
<td>0.71</td>
<td>66.8</td>
</tr>
<tr>
<td>4</td>
<td>0.73</td>
<td>67.8</td>
</tr>
<tr>
<td>5</td>
<td>0.68</td>
<td>64.1</td>
</tr>
<tr>
<td>6</td>
<td>0.69</td>
<td>63.2</td>
</tr>
<tr>
<td>7</td>
<td>0.66</td>
<td>60.3</td>
</tr>
</tbody>
</table>

4.15.7 Analysis of results of different Feature Extraction techniques

Different feature extraction techniques have been evaluated for the proposed classifier with different kernels and varying kernel parameter values. For all the feature extraction methods evaluated, an appropriate feature vector length was found by trial and error with different evaluations. It is inferred from results that the appropriate feature vector length for LPC
& LPCC are 40, PLP is 20, MFCC and GTCC is 25. A lower feature vector length indicates the ability of the feature extraction technique to contain the characteristic information of the data in fewer coefficients, and is therefore advantageous in classifier design as fewer coefficients will result in lower complexity.

The kernels employed for the classifier evaluation are polynomial, MLP and RBF. The experiments were done on trial and error basis. The range of parameter values are heuristically arrived at, in which the values that exhibited better performance over others are tabulated. From the results obtained, RBF kernel has shown to outperform polynomial kernel and MLP kernel for all feature extraction techniques evaluated.

A comparison of the performance results of different feature extraction techniques suggest that cepstral features exhibit better classification results compared to others. This substantiated the ability of cepstral based techniques to yield features that perform well with data which is non-linearly corrupted with noise.

The results and inferences obtained from feature extraction is used for the formation of the final feature vector of the classifier through feature selection technique. The best results, the appropriate feature vector length and the appropriate kernel parameter, obtained from all the described feature extraction techniques is used to form the feature vector for the feature selection stage.

4.15.8 Performance with Feature Selection

Feature selection techniques have been employed in the proposed classifier to select the most relevant features suitable for the classification problem at hand. The feature selection algorithms operate on a feature vector
composed of average value of energy and energy entropy, 10 frame values each of spectral centroid, spectral roll-off, spectral flux and the average value of spectral entropy followed by 40 coefficients of LPC, 40 coefficients of LPCC, 20 coefficients of PLP, 25 coefficients of MFCC and 25 coefficients of GTCC. The total length of the feature vector would be 183 and is depicted in Figure 4.45. The classifier is evaluated by filter based approaches such as CFS and ECS, and Wrapper based approaches such as SFS, SBS and GA.

The experimental results with filter based techniques such as CFS and ECS for varying size of feature subset is tabulated in Table 4-25 and Table 4-26. The feature ranking obtained with CFS and ECS methods are tabulated in Table 4.27.

Table 4-25 Performance results obtained with CFS based feature selection for varying feature subset size

<table>
<thead>
<tr>
<th>Vector Size</th>
<th>F-score</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.61</td>
<td>56.7</td>
</tr>
<tr>
<td>30</td>
<td>0.66</td>
<td>61.1</td>
</tr>
<tr>
<td>40</td>
<td>0.71</td>
<td>67.6</td>
</tr>
<tr>
<td>50</td>
<td>0.70</td>
<td>65.0</td>
</tr>
<tr>
<td>60</td>
<td>0.67</td>
<td>61.8</td>
</tr>
<tr>
<td>70</td>
<td>0.64</td>
<td>58.9</td>
</tr>
</tbody>
</table>
### Chapter 4 Target Classifier

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Average value of energy across all frames</td>
</tr>
<tr>
<td>2</td>
<td>Average value of energy entropy across all frames</td>
</tr>
<tr>
<td>3</td>
<td>Spectral centroid 1&lt;sup&gt;st&lt;/sup&gt; frame value</td>
</tr>
<tr>
<td>12</td>
<td>Spectral centroid 10&lt;sup&gt;th&lt;/sup&gt; frame value</td>
</tr>
<tr>
<td>13</td>
<td>Spectral roll-off 1&lt;sup&gt;st&lt;/sup&gt; frame value</td>
</tr>
<tr>
<td>22</td>
<td>Spectral roll-off 10&lt;sup&gt;th&lt;/sup&gt; frame value</td>
</tr>
<tr>
<td>23</td>
<td>Spectral flux 1&lt;sup&gt;st&lt;/sup&gt; frame value</td>
</tr>
<tr>
<td>32</td>
<td>Spectral flux 10&lt;sup&gt;th&lt;/sup&gt; frame value</td>
</tr>
<tr>
<td>33</td>
<td>Average value of spectral entropy across all frames</td>
</tr>
<tr>
<td>34</td>
<td>LPC 1&lt;sup&gt;st&lt;/sup&gt; coefficient</td>
</tr>
<tr>
<td>73</td>
<td>LPC 40&lt;sup&gt;th&lt;/sup&gt; coefficient</td>
</tr>
<tr>
<td>74</td>
<td>LPCC 1&lt;sup&gt;st&lt;/sup&gt; coefficient</td>
</tr>
<tr>
<td>113</td>
<td>LPCC 40&lt;sup&gt;th&lt;/sup&gt; coefficient</td>
</tr>
<tr>
<td>114</td>
<td>PLP 1&lt;sup&gt;st&lt;/sup&gt; coefficient</td>
</tr>
<tr>
<td>133</td>
<td>PLP 20&lt;sup&gt;th&lt;/sup&gt; coefficient</td>
</tr>
<tr>
<td>134</td>
<td>MFCC 1&lt;sup&gt;st&lt;/sup&gt; coefficient</td>
</tr>
<tr>
<td>158</td>
<td>MFCC 25&lt;sup&gt;th&lt;/sup&gt; coefficient</td>
</tr>
<tr>
<td>159</td>
<td>GTCC 1&lt;sup&gt;st&lt;/sup&gt; coefficient</td>
</tr>
<tr>
<td>183</td>
<td>GTCC 25&lt;sup&gt;th&lt;/sup&gt; coefficient</td>
</tr>
</tbody>
</table>

Fig.4.45 Depiction of feature vector which forms the input to feature selection
### Table 4-26 Performance obtained with ECS based feature selection for varying feature subset size

<table>
<thead>
<tr>
<th>Vector Size</th>
<th>F-score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.63</td>
<td>57.0</td>
</tr>
<tr>
<td>30</td>
<td>0.66</td>
<td>60.7</td>
</tr>
<tr>
<td>40</td>
<td>0.71</td>
<td>66.9</td>
</tr>
<tr>
<td>50</td>
<td>0.69</td>
<td>64.3</td>
</tr>
<tr>
<td>60</td>
<td>0.65</td>
<td>60.0</td>
</tr>
<tr>
<td>70</td>
<td>0.63</td>
<td>58.1</td>
</tr>
</tbody>
</table>

### Table 4-27 Feature rankings obtained with CFS and ECS methods

<table>
<thead>
<tr>
<th>Rank</th>
<th>Features (CFS)</th>
<th>Features (ECS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MFCC 6\textsuperscript{th} coefficient</td>
<td>MFCC 5\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>2</td>
<td>MFCC 3\textsuperscript{rd} coefficient</td>
<td>MFCC 11\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>3</td>
<td>GTCC 5\textsuperscript{th} coefficient</td>
<td>GTCC 5\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>4</td>
<td>MFCC 11\textsuperscript{th} coefficient</td>
<td>GTCC 4\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>5</td>
<td>MFCC 2\textsuperscript{nd} coefficient</td>
<td>GTCC 15\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>6</td>
<td>Average of energy</td>
<td>MFCC 22\textsuperscript{nd} coefficient</td>
</tr>
<tr>
<td>7</td>
<td>LPC 2\textsuperscript{nd} coefficient</td>
<td>MFCC 3\textsuperscript{rd} coefficient</td>
</tr>
<tr>
<td>8</td>
<td>GTCC 6\textsuperscript{th} coefficient</td>
<td>PLPC 7\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>9</td>
<td>PLPC 7\textsuperscript{th} coefficient</td>
<td>GTCC 3\textsuperscript{rd} coefficient</td>
</tr>
<tr>
<td>10</td>
<td>LPCC 17\textsuperscript{th} coefficient</td>
<td>Spectral centroid 3\textsuperscript{rd} frame value</td>
</tr>
<tr>
<td>11</td>
<td>MFCC 16\textsuperscript{th} coefficient</td>
<td>GTCC 10\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>12</td>
<td>GTCC 15\textsuperscript{th} coefficient</td>
<td>Spectral flux 9\textsuperscript{th} frame value</td>
</tr>
</tbody>
</table>
### Chapter 4 Target Classifier

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature Type</th>
<th>Coefficient Req.</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>PLPC 2nd coefficient</td>
<td>Spectral roll-off 4th frame value</td>
</tr>
<tr>
<td>14</td>
<td>GTCC 10th coefficient</td>
<td>PLPC 2nd coefficient</td>
</tr>
<tr>
<td>15</td>
<td>Spectral centroid 3rd frame value</td>
<td>LPCC 23rd coefficient</td>
</tr>
<tr>
<td>16</td>
<td>PLPC 17th coefficient</td>
<td>LPCC 27th coefficient</td>
</tr>
<tr>
<td>17</td>
<td>MFCC 20th coefficient</td>
<td>LPCC 2nd coefficient</td>
</tr>
<tr>
<td>18</td>
<td>LPCC 27th coefficient</td>
<td>LPC 39th coefficient</td>
</tr>
<tr>
<td>19</td>
<td>LPC 39th coefficient</td>
<td>Average of spectral entropy</td>
</tr>
<tr>
<td>20</td>
<td>LPC 8th coefficient</td>
<td>LPC 8th coefficient</td>
</tr>
<tr>
<td>21</td>
<td>Spectral roll-off 4th frame value</td>
<td>MFCC 7th coefficient</td>
</tr>
<tr>
<td>22</td>
<td>LPCC 3rd coefficient</td>
<td>MFCC 16th coefficient</td>
</tr>
<tr>
<td>23</td>
<td>LPCC 38th coefficient</td>
<td>MFCC 20th coefficient</td>
</tr>
<tr>
<td>24</td>
<td>Spectral centroid 1st frame value</td>
<td>MFCC 15th coefficient</td>
</tr>
<tr>
<td>25</td>
<td>Average of spectral entropy</td>
<td>LPC 2nd coefficient</td>
</tr>
<tr>
<td>26</td>
<td>Spectral flux 9th frame value</td>
<td>Average of energy</td>
</tr>
<tr>
<td>27</td>
<td>Average of energy entropy</td>
<td>LPCC 38th coefficient</td>
</tr>
<tr>
<td>28</td>
<td>LPC 34th coefficient</td>
<td>GTCC 25th coefficient</td>
</tr>
<tr>
<td>29</td>
<td>Spectral flux 4th frame value</td>
<td>LPCC 17th coefficient</td>
</tr>
<tr>
<td>30</td>
<td>LPC 18th coefficient</td>
<td>Average of energy entropy</td>
</tr>
<tr>
<td>31</td>
<td>PLPC 12th coefficient</td>
<td>GTCC 23rd coefficient</td>
</tr>
<tr>
<td>32</td>
<td>GTCC 25th coefficient</td>
<td>PLPC 15th coefficient</td>
</tr>
<tr>
<td>33</td>
<td>LPCC 37th coefficient</td>
<td>MFCC 24th coefficient</td>
</tr>
<tr>
<td>34</td>
<td>GTCC 4th coefficient</td>
<td>PLPC 12th coefficient</td>
</tr>
</tbody>
</table>
### Underwater Target Classifier with Improved Success Rate using Meta-Optimal Support Vector Machine

<table>
<thead>
<tr>
<th></th>
<th>First Feature</th>
<th>Second Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>LPC 35\textsuperscript{th} coefficient</td>
<td>Spectral centroid 7\textsuperscript{th} frame value</td>
</tr>
<tr>
<td>36</td>
<td>Spectral flux 3\textsuperscript{rd} frame value</td>
<td>GTCC 20\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>37</td>
<td>LPC 5\textsuperscript{th} coefficient</td>
<td>Spectral flux 3\textsuperscript{rd} frame value</td>
</tr>
<tr>
<td>38</td>
<td>MFCC 22\textsuperscript{nd} coefficient</td>
<td>Spectral flux 4\textsuperscript{th} frame value</td>
</tr>
<tr>
<td>39</td>
<td>PLPC 14\textsuperscript{th} coefficient</td>
<td>LPC 18\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>40</td>
<td>MFCC 23\textsuperscript{rd} coefficient</td>
<td>Spectral centroid 10\textsuperscript{th} frame value</td>
</tr>
<tr>
<td>41</td>
<td>PLPC 15\textsuperscript{th} coefficient</td>
<td>Spectral roll-off 1\textsuperscript{st} frame value</td>
</tr>
<tr>
<td>42</td>
<td>Spectral centroid 7\textsuperscript{th} frame value</td>
<td>LPC 27\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>43</td>
<td>Spectral roll-off 9\textsuperscript{th} frame value</td>
<td>LPC 33\textsuperscript{rd} coefficient</td>
</tr>
<tr>
<td>44</td>
<td>LPCC 33\textsuperscript{rd} coefficient</td>
<td>LPC 34\textsuperscript{rd} coefficient</td>
</tr>
<tr>
<td>45</td>
<td>LPC 23\textsuperscript{rd} coefficient</td>
<td>LPC 23\textsuperscript{rd} coefficient</td>
</tr>
<tr>
<td>46</td>
<td>LPC 40\textsuperscript{th} coefficient</td>
<td>LPCC 7\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>47</td>
<td>MFCC 5\textsuperscript{th} coefficient</td>
<td>GTCC 6\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>48</td>
<td>LPC 3\textsuperscript{rd} coefficient</td>
<td>MFCC 18\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>49</td>
<td>MFCC 8\textsuperscript{th} coefficient</td>
<td>LPCC 30\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>50</td>
<td>LPCC 7\textsuperscript{th} coefficient</td>
<td>MFCC 5\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>51</td>
<td>PLPC 5\textsuperscript{th} coefficient</td>
<td>PLPC 5\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>52</td>
<td>LPCC 11\textsuperscript{th} coefficient</td>
<td>GTCC 12\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>53</td>
<td>LPCC 4\textsuperscript{th} coefficient</td>
<td>LPCC 39\textsuperscript{th} coefficient</td>
</tr>
<tr>
<td>54</td>
<td>LPC 31\textsuperscript{st} coefficient</td>
<td>Spectral roll-off 9\textsuperscript{th} frame value</td>
</tr>
<tr>
<td>55</td>
<td>LPC 13\textsuperscript{th} coefficient</td>
<td>LPC 14\textsuperscript{th} coefficient</td>
</tr>
</tbody>
</table>
**Chapter 4 Target Classifier**

<table>
<thead>
<tr>
<th></th>
<th>Spectral flux 6\textsuperscript{th} frame value</th>
<th>LPC 13\textsuperscript{th} coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>GTCC 12\textsuperscript{th} coefficient</th>
<th>LPC 40\textsuperscript{th} coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>57</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Spectral centroid 10\textsuperscript{th} frame value</th>
<th>LPCC 12\textsuperscript{th} coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>58</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Spectral roll-off 1\textsuperscript{st} frame value</th>
<th>LPCC 37\textsuperscript{th} coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>LPC 27\textsuperscript{th} coefficient</th>
<th>LPCC 21\textsuperscript{st} coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>LPCC 35\textsuperscript{th} coefficient</th>
<th>Spectral centroid 5\textsuperscript{th} frame value</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>LPCC 23\textsuperscript{rd} coefficient</th>
<th>Spectral flux 6\textsuperscript{th} frame value</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>LPCC 39\textsuperscript{th} coefficient</th>
<th>LPCC 36\textsuperscript{th} coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>LPCC 20\textsuperscript{th} coefficient</th>
<th>LPCC 20\textsuperscript{th} coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>MFCC 13\textsuperscript{th} coefficient</th>
<th>PLPC 10\textsuperscript{th} coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>LPC 14\textsuperscript{th} coefficient</th>
<th>GTCC 18\textsuperscript{th} coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>66</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>GTCC 8\textsuperscript{th} coefficient</th>
<th>LPC 5\textsuperscript{th} coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>67</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>MFCC 18\textsuperscript{th} coefficient</th>
<th>LPC 6\textsuperscript{th} coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>68</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>LPC 36\textsuperscript{th} coefficient</th>
<th>LPCC 11\textsuperscript{th} coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>69</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>LPCC 21\textsuperscript{st} coefficient</th>
<th>LPCC 35\textsuperscript{th} coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The experimental results with wrapper based techniques such as SFS, SBS and GA based feature selection for the varying size of feature subset is tabulated in Table 4-298, Table 4-29 and Table 4-30.

<table>
<thead>
<tr>
<th>Vector Size</th>
<th>F-score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.63</td>
<td>56.3</td>
</tr>
</tbody>
</table>

152
Underwater Target Classifier with Improved Success Rate using Meta-Optimal Support Vector Machine

Table 4-28 Performance results of SFS based feature selection.

<table>
<thead>
<tr>
<th>Vector Size</th>
<th>F-score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.63</td>
<td>56.3</td>
</tr>
<tr>
<td>30</td>
<td>0.66</td>
<td>60.3</td>
</tr>
<tr>
<td>40</td>
<td>0.70</td>
<td>65.4</td>
</tr>
<tr>
<td>50</td>
<td>0.75</td>
<td>71.2</td>
</tr>
<tr>
<td>60</td>
<td>0.73</td>
<td>68.3</td>
</tr>
<tr>
<td>70</td>
<td>0.68</td>
<td>64.3</td>
</tr>
</tbody>
</table>

Table 4-29 Performance results of SBS based feature selection

<table>
<thead>
<tr>
<th>Vector Size</th>
<th>F-score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.65</td>
<td>58.5</td>
</tr>
<tr>
<td>30</td>
<td>0.66</td>
<td>60.3</td>
</tr>
<tr>
<td>40</td>
<td>0.73</td>
<td>68.3</td>
</tr>
<tr>
<td>50</td>
<td>0.70</td>
<td>72.3</td>
</tr>
<tr>
<td>60</td>
<td>0.69</td>
<td>70.9</td>
</tr>
<tr>
<td>70</td>
<td>0.63</td>
<td>66.9</td>
</tr>
</tbody>
</table>

Table 4-30 Performance results of GA based feature selection

<table>
<thead>
<tr>
<th>Vector Size</th>
<th>F-score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.55</td>
<td>57.4</td>
</tr>
<tr>
<td>30</td>
<td>0.65</td>
<td>60.0</td>
</tr>
<tr>
<td>40</td>
<td>0.71</td>
<td>66.1</td>
</tr>
</tbody>
</table>
From Table 4-28, Table 4-29 and Table 4-30, it can be inferred that the best feature vector size for the proposed classifier is 50. Features selected by SBS method outperforms the features selected by SFS and GA, in terms of performance. However, the run time required for executing sequential feature selection techniques has been found to be approximately 7 times than that of GA, at the compromise of slightly lower performance. For selecting a feature vector of size 50, the runtime required by sequential algorithms 174 minutes as compared to 27 minutes in case of GA, when simulated in MATLAB 2013b platform. Since, time is an important constraint in underwater target classification applications, the faster approach, GA, is selected compromising slightly in performance. The features selected, for feature vector size 50, with SFS, SBS and GA are listed in Table 4-31.

<table>
<thead>
<tr>
<th>Method</th>
<th>Selected features</th>
</tr>
</thead>
</table>
| SFS    | average value of energy, average value of energy entropy, spectral roll-off 4th frame value, spectral roll-off 8th frame value, average value of spectral entropy, LPC 3rd coefficient, LPC 8th coefficient, LPC 15th coefficient, LPC 18th coefficient, LPC 31st coefficient, LPC 33rd coefficient, LPC 36th coefficient, LPCC 2nd coefficient, LPCC 9th coefficient, LPCC 17th coefficient, LPCC 20th coefficient, LPCC 26th coefficient, LPCC 31st coefficient, LPCC 36th coefficient, PLP 2nd coefficient, PLP 5th coefficient, PLP 7th coefficient, PLP13th coefficient, PLP 14th coefficient, PLP 17th coefficient, MFCC 2nd coefficient, MFCC 3rd coefficient, MFCC 5th coefficient, MFCC 6th coefficient, MFCC 11th coefficient, MFCC 12th coefficient, MFCC 15th coefficient, MFCC 20th coefficient,
SBS

- MFCC 22\textsuperscript{nd} coefficient, MFCC 23\textsuperscript{rd} coefficient, MFCC 24\textsuperscript{th} coefficient, MFCC 25\textsuperscript{th} coefficient, GTCC 3\textsuperscript{rd} coefficient, GTCC 5\textsuperscript{th} coefficient, GTCC 6\textsuperscript{th} coefficient, GTCC 7\textsuperscript{th} coefficient, GTCC 15\textsuperscript{th} coefficient, GTCC 16\textsuperscript{th} coefficient, GTCC 18\textsuperscript{th} coefficient, GTCC 19\textsuperscript{th} coefficient, GTCC 20\textsuperscript{th} coefficient, GTCC 22\textsuperscript{nd} coefficient, GTCC 23\textsuperscript{rd} coefficient, GTCC 25\textsuperscript{th} coefficient
- Average value of energy, average value of energy entropy, spectral roll-off 4\textsuperscript{th} frame value, spectral flux 3\textsuperscript{rd} frame value, spectral flux 8\textsuperscript{th} frame value, average value of spectral entropy, LPC 3\textsuperscript{rd} coefficient, LPC 8\textsuperscript{th} coefficient, LPC 15\textsuperscript{th} coefficient, LPC 18\textsuperscript{th} coefficient, LPC 31\textsuperscript{st} coefficient, LPC 33\textsuperscript{rd} coefficient, LPC 36\textsuperscript{th} coefficient, LPCC 2\textsuperscript{nd} coefficient, LPCC 9\textsuperscript{th} coefficient, LPCC 17\textsuperscript{th} coefficient, LPCC 20\textsuperscript{th} coefficient, LPCC 26\textsuperscript{th} coefficient, LPCC 31\textsuperscript{st} coefficient, LPCC 33\textsuperscript{rd} coefficient, LPCC 36\textsuperscript{th} coefficient, PLP 2\textsuperscript{nd} coefficient, PLP 5\textsuperscript{th} coefficient, PLP 7\textsuperscript{th} coefficient, PLP 11\textsuperscript{th} coefficient, PLP 14\textsuperscript{th} coefficient, PLP 17\textsuperscript{th} coefficient, MFCC 2\textsuperscript{nd} coefficient, MFCC 3\textsuperscript{rd} coefficient, MFCC 5\textsuperscript{th} coefficient, MFCC 6\textsuperscript{th} coefficient, MFCC 11\textsuperscript{th} coefficient, MFCC 12\textsuperscript{th} coefficient, MFCC 15\textsuperscript{th} coefficient, MFCC 20\textsuperscript{th} coefficient, MFCC 22\textsuperscript{nd} coefficient, MFCC 23\textsuperscript{rd} coefficient, MFCC 24\textsuperscript{th} coefficient, MFCC 25\textsuperscript{th} coefficient, GTCC 3\textsuperscript{rd} coefficient, GTCC 4\textsuperscript{th} coefficient, GTCC 6\textsuperscript{th} coefficient, GTCC 7\textsuperscript{th} coefficient, GTCC 15\textsuperscript{th} coefficient, GTCC 16\textsuperscript{th} coefficient, GTCC 18\textsuperscript{th} coefficient, GTCC 19\textsuperscript{th} coefficient, GTCC 20\textsuperscript{th} coefficient, GTCC 22\textsuperscript{nd} coefficient, GTCC 23\textsuperscript{rd} coefficient

GA

- Average value of energy, average value of energy entropy, spectral roll-off 4\textsuperscript{th} frame value, spectral flux 3\textsuperscript{rd} frame value, spectral flux 8\textsuperscript{th} frame value, average value of spectral entropy, LPC 3\textsuperscript{rd} coefficient, LPC 8\textsuperscript{th} coefficient, LPC 15\textsuperscript{th} coefficient, LPC 18\textsuperscript{th} coefficient, LPC 31\textsuperscript{st} coefficient, LPC 34\textsuperscript{th} coefficient, LPC 35\textsuperscript{th} coefficient, LPC 2\textsuperscript{nd} coefficient, LPCC 3\textsuperscript{rd} coefficient, LPCC 17\textsuperscript{th} coefficient, LPCC 20\textsuperscript{th} coefficient, LPCC 26\textsuperscript{th} coefficient, LPCC 31\textsuperscript{st} coefficient, LPCC 33\textsuperscript{rd} coefficient, LPCC 36\textsuperscript{th} coefficient, PLP 2\textsuperscript{nd} coefficient, PLP 5\textsuperscript{th} coefficient, PLP 7\textsuperscript{th} coefficient, PLP 13\textsuperscript{th} coefficient, PLP 14\textsuperscript{th} coefficient, PLP 17\textsuperscript{th} coefficient, MFCC 2\textsuperscript{nd} coefficient, MFCC 3\textsuperscript{rd} coefficient, MFCC 5\textsuperscript{th} coefficient, MFCC 6\textsuperscript{th} coefficient, MFCC 11\textsuperscript{th} coefficient
4.16 Summary

Features are signature patterns that best represent the signal and are given as classification cues to the classifier. The feature vector given to the classifier is of profound importance in determining the complexity and performance of the classifier. The chapter describes the various acoustic feature extraction techniques as well as feature selection techniques. A properly chosen feature vector enables the classifier to build better generalizable models. The choice of the specific feature vector for a particular classification task is derived from conclusions and inferences drawn from extensive experimentation.

The chapter also gives a brief introduction to machine learning and also the various concepts associated with it such as overfitting, underfitting and the bias-variance trade-off which led to the formulation of SVM. The chapter describes SVM’s and the formulation of multiclass SVM’s.
CHAPTER 5

CLASSIFIER OPTIMISATION

The parameters of classification algorithms have a profound influence on their performance in terms of generalization ability and robustness to noise. For an SVM-based classifier, the kernel function plays a vital role in dealing with non-linear and arbitrarily structured data, whose parameters also have an impact on the classification performance.

Any attempt to improve the performance of an SVM based classifier must invariably include parameter optimisation. Also, underwater target classification with dynamically changing constraints, necessitates dynamic selection of the optimal algorithmic parameters, kernel function and kernel parameters, which can be achieved through various optimisation techniques. The idea is to find out the parameters that maximize the performance of the proposed classifier. This chapter throws light upon the different optimisation strategies that can be adopted, associated concepts, and a brief about their merits and demerits. The chapter mainly focuses on meta-heuristic optimization algorithms, and also the different meta-heuristic optimisation algorithms adopted in this work.

5.1 Background - The Optimisation problem

Solutions to problems may not always have a binary nature, but are often rated in terms of quality concerning a performance metric. Learning algorithms, usually depend on parameters which control the size of the search
space or the way the search is conducted in the search space. Optimisation is performed to select the one which is of the best quality from a set of many candidate solutions that differ in quality, under the given circumstances. Optimisation reflects from the analytical quality of all intelligent beings to pursue the best. The mathematical formulation of the concept of optimisation can be as follows. Consider a set of candidate solutions, \( X \), subjected to the optimisation problem. Typically, \( X \) is n-dimensional over certain domain, referred to as the search space. The optimisation problem is defined by an objective function (also referred to as cost function or fitness function), to estimate the performance of the candidate in \( X \) on the given problem.

\[
f: X \rightarrow \mathbb{R}
\]

The objective function, typically expressed as a function of design variable, defines a criteria to compare possible solutions. The optimisation problem can be formulated with an objective to find the best candidate solution that minimises or maximizes the fitness function \( f \), as follows

\[
\text{find } x \in X, \text{ so that } \forall y \in X : f(x) \leq f(y)
\]

In practical applications, solutions to this problem may be a subset of all possible combinations or permutations of the elements of the vector \( X \). Such problems are characterised by a finite set of solutions and are referred to as combinatorial optimisation problems.

5.2 Classification of Optimisation Problems

Optimisation problem can be classified in several ways as follows

- Constrained and Unconstrained: Constrained optimisation problems are subjected to one or more constraints and hence maintain the
search effort within a feasible region, whereas unconstrained optimisation problems have no constraints at all.

- Single objective and multi-objective: Optimisation problems may be based on a single objective function or multiple objective functions which involves decision making with multiple criteria.

- Nature of objective function and constraints: Optimisation problems can be classified as linear, quadratic, polynomial and non-linear depending upon the nature of the objective functions and the constraints. Determination of the type of solution often depends on the nature of involved functions.

- Deterministic and Stochastic: In deterministic optimisation problem, all the design variables are deterministic. In stochastic optimisation problem some or all the parameters are stochastic (non deterministic or probabilistic).

**5.3 Curse of Dimensionality**

The only way to ensure global optimality in solutions of an optimization problem is to evaluate all the candidate solutions, which is often computationally intractable. The adversity exponentially grows worse with increasing dimensionality, which is termed as the curse of dimensionality. Hence, it is essential to adopt optimisation methods that give acceptable performance in fewer dimensions and avoid exponentially increasing number of fitness evaluations with increasing dimensions. The optimisation methods should preferably have a linear relationship between the dimensionality of the problem and the number of candidate solutions i.e. it should have linear time complexity $O(n)$ in the dimensionality $n$ of the problem to be optimised.
5.4 NP Theory

NP theory encompasses concepts that explain the solvability of an algorithm. An optimisation algorithm can be considered as efficient if it has polynomial run time [162]. The problems solved in polynomial time, by a deterministic machine is referred to as P problem while by non-deterministic machines is referred as NP (non-deterministic polynomial). The solutions to these problems are also verifiable in polynomial time.

NP-hardness (non-deterministic polynomial-time hardness), in computational complexity theory, is the defining property of a class of problems that are, informally, "at least as hard as the hardest problems in NP". More precisely, a problem H is NP-hard when every problem L in NP can be reduced in polynomial time to H; that is, assuming a solution for H takes 1 unit time, we can use H's solution to solve L in polynomial time. As a consequence, finding a polynomial algorithm to solve any NP-hard problem would give polynomial algorithms for all the problems in NP, which is unlikely as many of them are considered hard.

5.5 No Free Lunch Theorem

The No Free Lunch (NFL) theorem proposed by Wolpert and Macready [163] states that 'if an algorithm performs well in a certain class of problems, then it necessarily pays for that with degraded performance on the set of all remaining problems'. Consider \(f : S \rightarrow Y\) which is the pair of all possible mappings within the range of values \(Y_s\) in the solution space \(S\).

Then \(T_m\) is defined as

\[
T_m = \{(s_1, y_1), \ldots, (s_m, y_m)\} \text{ where } y_j = f(s_j) \quad 5.3
\]

NFL theorem states that for any pair of algorithms \(a\) and \(b\),
Underwater Target Classifier with Improved Success Rate using Meta-Optimal Support Vector Machine

\[ \sum_{f \in F} P(T_m^f|f, m, a) = \sum_{f \in F} P(T_m^f|f, m, b) \quad 5.4 \]

where \( P(T_m^f|f, m, a) \) is the probability of obtaining a certain sequence of values \( y_1, y_2, y_3, ..., y_m \) given that the search is for the function \( f \) with algorithm \( a \).

The NFL theorem suggests that any two algorithms are equivalent when their performance is averaged across all problems, and hence, any one optimisation method will be as likely as any other to find a satisfactory solution. Thus for any optimisation algorithm, high performance over one class of problems is offset by performance over another class and a universally best method does not exist as illustrated in Figure 5.1. There may exist an optimisation method specialised for a particular problem. There are also algorithms which deliver good results for many different problem classes, but may be outperformed by a highly specialised method in each of them.

Another manifestation of NFL theorem is related to the objective of an optimisation problem. The objective function of an optimisation problem is typically formulated based on the expected physical outcome as well as economic considerations. Following the NFL theorem, an algorithm which
performs well for a particular objective, say speed, may perform poorly for another metric say, accuracy. Thus, there always exists a fundamental trade-off between performance vs cost or performance vs reliability aspects of an optimisation algorithm and no choice is universally better than any other [115].

5.6 Optimisation Search Strategies

A large number of optimisation algorithms have been developed based on different search strategies. The naïve strategy towards finding the optimal parameters is to conduct an exhaustive search in the parameter space, which guarantees to find an optimal solution at a problem-specific computational cost. Since many practical optimisation problems are NP-hard, no polynomial-time algorithm is known, and the computational cost for exhaustive search is prohibitively high since it would correspond to running the algorithm for every possible value of parameters and hence the ideal linear time complexity $O(n)$ described in section 5.4 cannot be attained.

The traditional way of parameter optimisation is to conduct a grid search, a brute force method which exhaustively searches through the subset of parameters of the learning algorithm, guided by some performance metric. Grid search involves dividing the parameter space into grids of uniform size. The model is trained for the parameter values at each grid locations and the response is evaluated through a suitable performance metric, to obtain the optimal parameter set.

Another method of parameter optimisation is to employ greedy search techniques, which makes a greedy choice at each step for a solution that yields maximum performance, with the hope of finding a global optimum, without assessing its consequences. These techniques have high chances of
getting trapped in local optima. Greedy search techniques work well for monotonous objective function with smooth solution spaces [164]. However, for multimodal objectives, it is likely to get stuck in local optima. This approach is also called hill-climbing approach, referring to a mountaineer who chooses every next step in a manner that yields maximal improvement. Gradient search algorithms are examples of greedy search techniques.

Stochastic, heuristic and meta-heuristic optimisation techniques are resorted to, in applications where exhaustive search techniques seem to be impractical. Stochastic optimisation techniques are applied to solve highly non-linear, high-dimensional data difficult to be solved by classical deterministic methods. Stochastic optimisation techniques generate and use random variables for the formulation of the optimisation problem which involve random objective functions or random constraints. These techniques are faster than exhaustive search techniques but cannot guarantee best solutions or global solutions. Heuristic optimisation techniques are experience based techniques and are employed to solve complex logistics problems of higher dimension [165]. Heuristic derives from the Greek verb *heuriskein* means ‘to find’ by trial and error, and heuristic algorithms start searching the solution space with an initial guess and try to improve the quality of solutions over the course of iterations. Heuristic algorithms are capable of finding satisfactory solutions at a faster rate, but do not always guarantee to find optimal solutions.

Meta-heuristic algorithms are strategies that guide the search process. ‘Meta’ means ‘beyond’ or ‘higher level’, and the meta-heuristic algorithm guides a subordinate heuristic search through a combination of intelligent randomisation and local search, or in other words, exploration and exploitation. “A meta-heuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently
different concepts for exploring and exploiting the search space; learning strategies are used to structure information to efficiently find near-optimal solutions [166].” Meta-heuristics is an algorithmic architecture that customises the algorithmic procedures of a more problem-specific subordinate heuristics to obtain high-quality solutions. The subordinate heuristics may be a high-level or low-level procedure ranging from a simple local search to a complex search procedure.

Exploration (randomisation/diversification) and exploitation (intensification) are the two competing driving forces of a meta-heuristic algorithm, which allows it to effectively and efficiently explore the search space. Exploration and exploitation of the search space, are two competing design goals, in which the algorithm must be ‘clever’ to intensively exploit areas of the search space with high-quality solutions, as well as move to unexplored search spaces when required. Exploration increases the diversity of solutions and prevents them from being trapped at local optima, while exploitation in promising areas of the search space, based on accumulated search experience will increase the chances of the algorithm to find optimal solutions. A good combination of exploration and exploitation usually ensure global optimality [166].

Nature has evolved over millions of years and has found optimal solutions to a variety of natural phenomena. Nature-inspired meta-heuristic algorithms mimic different strategies of nature in finding optimal solutions. They can be grouped in four main categories: evolution-based, physics-based, swarm-based and human-based algorithms as depicted in Fig. 5.2 [167].

Evolution-based algorithms, inspired by the laws of natural evolution, arrive at an optimal solution by optimising a randomly generated
population of possible solutions, over the course of iterations. The most popular evolution-inspired technique is Genetic Algorithms (GA) that simulates the evolutionary laws of nature. Physics-based meta-heuristic algorithms are formulated by imitating the physical laws in the universe. The most popular algorithms in this category are Simulated Annealing (SA) and Gravitational Local Search (GLS).

Swarm-based techniques are formulated by abstracting the social behaviour of groups of animals or birds. The most popular algorithm is Particle Swarm Optimisation, which inspired by the social behaviour of bird flocking or fish schooling. Human based algorithms are formulated by modelling the human behaviour. An example of human based algorithm is the brain storm (BS) optimisation algorithm which is inspired by the human brainstorming process.

5.7 Parameter Optimisation

Machine learning algorithms have behavioural parameters which have profound effects on their performance. These parameters values are heavily problem dependent in the sense that, a set of values might work best on a certain problem, well on certain other problem instances and bad on another class of instances. Hence, setting the right algorithmic parameters is an influential design goal in machine learning.

Attempts to improve the performance of a classifier should invariably consider estimating suitable parameters for the problem at hand. However, finding the best parameters is a consequential task and it is very ambitious to understand the effect of each parameter. Some parameters may even have effects on other parameters which make the problem all the more complicated. Therefore, in order to determine parameters that are adaptable to the classification problem, parameter optimisation algorithms are resorted.
Chapter 5 Classifier Optimisation

Fig. 5.2 Classification of meta-heuristic algorithms

Evolutionary algorithms
- GA – Genetic Algorithm
- ES – Evolution Strategy
- GP – Genetic Programming
- DE – Differential Evolution

Physics based algorithms
- GSA – Gravitational Search Algorithm
- CSS – Charged System Search
- CFO – Central Force Optimisation
- BBBC – Big Bang-Big Crunch

Swarm based algorithms
- AC – Ant Colony Optimisation
- PSO – Particle Swarm Optimisation
- ABS – Abaffy Broyden Spedicato
- CS – Cuckoo Search

Human based algorithms
- TLBO – Teaching – Learning – Based Optimisation
- HS – Hirschberg-Sinclair
- TS – Tabu Search
- BS – Brain Storm Optimisation

GA – Genetic Algorithm
ES – Evolution Strategy
GP – Genetic Programming
DE – Differential Evolution
GSA – Gravitational Search Algorithm
CSS – Charged System Search
CFO – Central Force Optimisation
BBBC – Big Bang-Big Crunch
AC – Ant Colony Optimisation
PSO – Particle Swarm Optimisation
ABS – Abaffy Broyden Spedicato
CS – Cuckoo Search
TLBO – Teaching – Learning – Based Optimisation
HS – Hirschberg-Sinclair
TS – Tabu Search
BS – Brain Storm Optimisation
The search space for most parameter optimisation tasks is large, and hence exhaustive search techniques are normally not feasible in terms of runtime. Furthermore, the interdependency between the parameters and their effect on the algorithm’s performance is largely unknown most of the times. Meta-heuristic algorithms, which strikes a balance between exhaustive search as well as heuristics are adopted in this work for optimising the parameter values of the underlying SVM based target classifier. Five nature inspired meta-heuristics algorithms namely, genetic algorithm, bat algorithm, whale optimisation algorithm, stochastic fractal search and symbiotic organisms search algorithm have been adopted for improving the classifier performance by tuning the algorithmic parameters. We have also proposed a modified symbiotic organisms search algorithm which is found to have better performance in classifying the underwater targets of interest.

### 5.8 Genetic Algorithm (GA)

Genetic algorithm (GA), is a meta-heuristic search and optimisation algorithm formulated by the abstraction of the ‘survival of fittest’ among individuals over consecutive generations in natural evolution [168]. GA starts with a population of candidate solutions (represented by chromosomes) for the problem at hand, characterising the natural ecosystem and makes it evolve by iteratively applying a set of stochastic operators like selection, crossover and mutation.

The chromosomes contain genes which encode a particular trait of the individual which may take different settings called alleles. An individual is also referred to as genome and the set of genes in a genome are referred to as genotype, which is the base for an organisms phenotype or the traits and characteristics it exhibits. They use metaphors which follow the theory of
natural evolution such as selection, recombination and mutation to evolve the solution to a problem.

The population consists of a number of co-existing organisms that compete for the same resources. The distinguishing traits of organisms that are most fit and capable of gathering resources will be carried over to the next generation. The entire population of the ecosystem is said to evolve over time, to accommodate organisms that are fitter than the previous generation. Thus, the characteristics that promote survival are preserved across generations.

At the beginning of the run of a genetic algorithm a population of chromosomes is created randomly, representing different solutions to the problem at hand. A fitness measure (objective function) is defined for evaluating the chromosomes and assigns a fitness score to each chromosome to assess its ability to solve the problem at hand. The algorithm then selects two members from the current population as parents to create offspring in the next generation. There are different methods for the selection process, however the chances of being selected as parents for the next generation is proportional to their fitness scores. Offspring for the next generation are created from parents by exchanging the genes in a process called cross-over. The algorithm then mutates the bits in a chromosome depending on a predefined mutation rate. After cross-over and mutation, the off-springs are evaluated just like their parents to measure their fitness of survival in the population. Fig.5.3 depicts the flowchart of GA.
5.8.1 GA operators

5.8.1.1 Selection

Selection is the process of choosing successful solutions from the current population as parents, which mate and recombine to create off-springs for the next generation. Selection is very crucial for the convergence rate of the GA, as good parents generate better and fitter solutions. Different techniques for selection in GA include,

- Tournament selection: Several tournaments are played among individuals chosen at random from the population. The winner of the tournament is selected for next generation to become a parent. With
Chapter 5 Classifier Optimisation

sufficiently large tournament size, weak individuals have a lower probability to be selected, and thus selection pressure can be adjusted by altering the tournament size.

- **Roulette wheel selection**: In this method of selection, segments are created for each individual according to their fitness. A random number is generated and the individual whose segment spans the random number is selected. This technique is called so as it is homologous to a roulette wheel in which all chromosomes in the population has a slice proportional to its fitness.

- **Rank selection**: Rank selection is mostly employed when the individuals in the population have very close fitness values and selection by roulette wheel will have very low selection pressure towards fitter individuals resulting in a bad selection of parents. In rank selection, the population is ranked according to their fitness and the parents are selected depending on their ranking, with more preference given to higher ranked individuals.

- **Steady-state selection**: GA in steady state means that there are no generations, and instead of replacing children of selected parents in the next generation, two best individuals out of the two parents alongwith their children are added back into the population so that the population size remains constant.

5.8.1.2 **Crossover**

Crossover is the process of combining parents to produce offspring and is analogous to reproduction and biological crossover. Crossover operation has the primary responsibility of converging the search algorithm to an optimal solution. There are different techniques of crossover, some of which are detailed as in the following paragraphs.
• Single point crossover: A random crossover point is selected and the bits next to the cross-sites (tails of the parents) are swapped to create new off-springs as depicted in Fig.5.4.

![Fig.5.4 Depiction of single point crossover](image)

• N-point crossover: Also called multi-point crossover, N crossover points are selected at random and the offsprings are created by combining the parents at the crossover point as depicted in Fig.5.5.

![Fig.5.5 Depiction of N-point crossover](image)

• Uniform crossover: In uniform crossover a crossover mask with binary values is created, and offsprings are created by copying genes from the parents according to the values in this mask. At positions where there is a ‘1’ in the mask, genes are carried from one parent, and at positions where there is a ‘0’ in the mask, genes are carried from the other parent.

5.8.1.3 Mutation

Mutation operator, analogous to the biological mutation alters the value of one or more genes, to maintain genetic diversity across generations. Mutation allows exploration of the search space and is essential to the convergence of GA. Commonly used mutation schemes are random, swap, scramble, inversion, uniform as well as gaussian mutation.
Random Mutation: Bits to be mutated are selected randomly and flipped.

Swap Mutation: Values of individual genes are swapped between two positions on the chromosome.

Scramble Mutation: Values of a selected subset of genes are randomly scrambled or shuffled.

Inversion Mutation: Values of a selected subset of genes are inverted.

Uniform Mutation: The chosen gene to be mutated is replaced by a random value between a user specified upper and lower bounds.

Gaussian Mutation: A Gaussian distributed random values is added to the chosen gene, and if it falls outside user specified bounds, the gene value is clipped.

The proposed SVM based underwater classifier was optimised with GA. The fitness measure employed is F-score. The results of the experiments for different population sizes are tabulated in Table 5-1. Each run of the algorithm consists of 500 iterations, except when the algorithm is terminated when the average relative change in fitness value is stalled over 25 generations.

<table>
<thead>
<tr>
<th>Population Size</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>F-score (class 1)</strong></td>
<td>0.60</td>
<td>0.68</td>
<td>0.71</td>
<td>0.71</td>
</tr>
<tr>
<td><strong>F-score (class 2)</strong></td>
<td>0.74</td>
<td>0.75</td>
<td>0.76</td>
<td>0.75</td>
</tr>
<tr>
<td><strong>F-score (class 3)</strong></td>
<td>0.73</td>
<td>0.74</td>
<td>0.75</td>
<td>0.74</td>
</tr>
<tr>
<td><strong>F-score (class 4)</strong></td>
<td>0.84</td>
<td>0.85</td>
<td>0.86</td>
<td>0.86</td>
</tr>
<tr>
<td>F-score (class 5)</td>
<td>0.79</td>
<td>0.79</td>
<td>0.80</td>
<td>0.79</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>F-score (class 6)</td>
<td>0.83</td>
<td>0.89</td>
<td>0.92</td>
<td>0.90</td>
</tr>
<tr>
<td>F-score (class 7)</td>
<td>0.75</td>
<td>0.75</td>
<td>0.77</td>
<td>0.75</td>
</tr>
<tr>
<td>F-score (class 8)</td>
<td>0.82</td>
<td>0.86</td>
<td>0.87</td>
<td>0.87</td>
</tr>
<tr>
<td>F-score (class 9)</td>
<td>0.79</td>
<td>0.79</td>
<td>0.80</td>
<td>0.79</td>
</tr>
<tr>
<td>F-score (class 10)</td>
<td>0.60</td>
<td>0.61</td>
<td>0.66</td>
<td>0.63</td>
</tr>
<tr>
<td>F-score (class 11)</td>
<td>0.59</td>
<td>0.65</td>
<td>0.7</td>
<td>0.69</td>
</tr>
<tr>
<td>F-score (Average)</td>
<td>0.73</td>
<td>0.76</td>
<td>0.78</td>
<td>0.77</td>
</tr>
<tr>
<td>Overall Accuracy (%)</td>
<td>69.1</td>
<td>71.3</td>
<td>75.6</td>
<td>73.8</td>
</tr>
</tbody>
</table>

5.9 **BAT Algorithm**

BAT algorithm was formulated by Xin-She Yang [169] based on the echolocation behaviour of bats. Bats use sonar to detect prey and avoid obstacles by emitting ultrasonic bursts and use the time delay information between the emission and detection of the echo, variation of loudness in the echo and the time difference between their ears to visualize their surroundings. Bats emit about 10 to 20 ultrasonic sound bursts per second typically of 5 to 20 ms duration. They tend to decrease loudness and increase the rate of emission to about 200 pulses per second when they approach a prey. They are able to detect the distance and orientation of the target, distinguish between different types of prey and can even estimate the moving speed of the prey such as small insects.

The characteristics of the echolocation pulse of bats vary with species and the following approximations are adopted to idealize the echo location characteristics of bats in order to formulate the BAT algorithm.
i. All bats use echolocation and they know the difference between food/prey and background barriers in some magical way.

ii. Bats fly randomly with velocity $v_i$ at position $x_i$ with a fixed frequency $f_{min}$ in, varying wavelength $\lambda$ and loudness $A_0$ to search for the prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r$ in the range $[0, 1]$, depending on the proximity of their target.

iii. Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) $A_0$ to a minimum constant value $A_{min}$.

With the above approximations, the algorithm is formulated as follows. The algorithm commences by initializing the position and velocity of a population of bats. During the course of iterations, the position, velocity and frequency of the bats are updated as,

$$x_i(t + 1) = x_i(t) + v_i(t + 1) \quad 5.5$$

$$v_i(t + 1) = v_i(t) + (x_i(t) - G_{best})Q_i \quad 5.6$$

where $G_{best}$ is the best solution obtained so far and $Q_i$ indicates the frequency of the $i^{th}$ bat which is updated in each course of iteration as follows,

$$Q_i = Q_{min} + (Q_{max} - Q_{min})\beta \quad 5.7$$

where, $\beta$ is a random number drawn from a uniform distribution in $[0,1]$. To introduce exploitation of search space into the algorithm, a local random walk is performed around randomly picked solutions and is described in equation 5.8.

$$x_{new} = x_{old} + \epsilon A^t \quad 5.8$$
where $\epsilon \in [-1,1]$ is a random number, and $A_t = \langle A^t \rangle$ is the average loudness of all the bats at the time step $t$. The loudness of the bats decreases and pulse emission rate increases when the bats approach the prey which is simulated in the algorithm to ensure convergence. During the course of iterations, when the solutions are improved, the loudness and emission rates are updated by the following equations to converge to the optimal solution.

$$A_{t+1}^i = \alpha A_t^i$$

$$r_{t+1}^i = r_t^i [1 - \exp(-\gamma t)]$$

where $0 < \alpha < 1, \gamma > 0$ and as simulations proceeds and the algorithm converges, $A_t^i \to 0, r_t^i = r_0^i$ as $t \to \infty$. The pseudo-code of BAT algorithm is shown in Fig.5.6.

BAT algorithm is in a way improvement over the particle swarm optimisation algorithm. The update of velocities and positions of the bats are similar to updating the pace and range of the swarming particle in PSO. However, an intense local search controlled by loudness and pulse rate makes BAT algorithm superior to PSO due to its effectiveness in balancing exploration and exploitation.

The proposed SVM based underwater classifier was optimised with BAT algorithm. The results of the experiments for different population sizes are tabulated in Table 5-2. Each run of the algorithm consists of 500 iterations, except when the algorithm is terminated when the average relative change in fitness value is stalled over 25 generations.
Chapter 5 Classifier Optimisation

Define pulse frequency
Initialize pulse rate, loudness
Initialize microbats (position, velocity and frequency)
Calculate the fitness and find the initial best solution

while (max no. of iteration not reached)
{
Generate new solutions by eqns 5.5, 5.6 & 5.7
if (rand > pulse rate)
    Select a solution among the best solutions
    and generate a local solution around the best
    solution, eqn 5.8
endif

Evaluate the new solutions by their fitness values
if (fitness has improved & solution is not too loud)
    decrease loudness and increase pulse
    emission rate, eqns 5.9 & 5.10
    accept the new solutions
endif

find the current best solution
}

Fig.5.6 BAT algorithm pseudo-code

Table 5-2 Performance results with BAT algorithm based optimisation for varying population size

<table>
<thead>
<tr>
<th>Population Size</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-score (class 1)</td>
<td>0.60</td>
<td>0.65</td>
<td>0.63</td>
<td>0.63</td>
</tr>
<tr>
<td>F-score (class 2)</td>
<td>0.73</td>
<td>0.75</td>
<td>0.74</td>
<td>0.73</td>
</tr>
<tr>
<td>F-score (class 3)</td>
<td>0.72</td>
<td>0.77</td>
<td>0.75</td>
<td>0.72</td>
</tr>
<tr>
<td>F-score (class 4)</td>
<td>0.86</td>
<td>0.87</td>
<td>0.86</td>
<td>0.86</td>
</tr>
<tr>
<td>F-score (class 5)</td>
<td>0.79</td>
<td>0.85</td>
<td>0.81</td>
<td>0.81</td>
</tr>
<tr>
<td>F-score (class 6)</td>
<td>0.85</td>
<td>0.90</td>
<td>0.88</td>
<td>0.87</td>
</tr>
<tr>
<td>F-score (class 7)</td>
<td>0.75</td>
<td>0.78</td>
<td>0.77</td>
<td>0.75</td>
</tr>
</tbody>
</table>
5.10 Whale Optimisation Algorithm

Whale Optimisation Algorithm proposed by S. Mirjalili and A. Lewis, is a nature inspired meta-heuristic algorithm, formulated by the abstraction of hunting behavior of humpback whales [167].

![Bubble-net feeding behaviour of humpback whales](image)

Fig.5.7 Bubble-net feeding behaviour of humpback whales

Humpback whales have a unique co-operative foraging behaviour which is popularly known as bubble-net feeding method shown in Fig.5.7, in which they use co-operative strategies to disorient and corral the fish into a bubble net that they create. They prefer to hunt school of small fish such as salmon, krill or herring. The whales gather together beneath the surface and
exhales out of their blowholes, to blow bubbles along a circle or 9 shaped path while continuing to encircle their prey and corralling the fish into the bubble net. The whales then simultaneously swim upwards with mouths wide open to feed on the trapped fish [170].

The WOA algorithm is modeled in three phases by simulating i.e. encircling the prey, the bubble or net attacking (exploitation phase), and the search for prey (exploration phase), behaviour of the humpback whales [167].

5.10.1 Encircling prey

Humpback whale recognizes the location of their prey and encircles them. The WOA starts with an initial population of solutions. The current best candidate solution is assume as the target prey. After the best search agent is defined, the other search agents will hence try to update their positions towards the best search agent which is mathematically formulated in equations 5.11 and 5.12.

\[
\vec{D} = |\vec{C}X^* - \vec{X}(t)| \\
\vec{X}(t + 1) = \vec{X}^*(t) - \vec{A}\vec{D}
\]

where \(t\) indicates the current iteration, and \(\vec{A}\) and \(\vec{C}\) coefficient vectors, \(\vec{X}\) is the position vector and \(X^*\) is the position vector of the best solution obtained so far. The vectors \(\vec{A}\) and \(\vec{C}\) are given by

\[
\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \\
\vec{C} = 2 \cdot \vec{r}
\]

where \(\vec{a}\) is linearly decreased from 2 to 0 over the course of iterations (in both exploration and exploitation phases) and \(\vec{r}\) is a random vector in [0,1].
5.10.2 Bubble-net attacking method

Humpback whales simultaneously swim around their prey within a shrinking circle and also along a spiral-shaped path, which is modelled with a 50% probability for updating the positions of the whales, either between the shrinking encircling mechanism or the spiral model as in equation 5.15.

\[
\ddot{X}(t + 1) = \begin{cases} 
  -\ddot{A} \cdot \ddot{D} & \text{if } p < 0.5 \\
  \ddot{D}' \cdot e^{b1 \cdot \cos(2\pi l)} + \ddot{X}^i(t) & \text{if } p \geq 0.5 
\end{cases} \quad 5.15
\]

To model the shrinking encircling mechanism the values of \( \ddot{A} \) is chosen to be a random value in the interval \([-a, a]\), where \( a \) is decreased from 2 to 0 over the course of iterations. For modeling the spiral position update, a spiral equation is created between the position of whale and prey to mimic the helix shaped movement. \( \ddot{D}' = |\ddot{X}^i(t) - \ddot{X}(t)| \) indicates the distance of the \( i^{th} \) whale to the best solution obtained so far, which indicates the position of the prey, \( b \) is a constant for defining the shape of the spiral and \( l \) is a random number in \([-1,1]\).

5.10.3 Search for prey

Humpback whales search randomly for prey. Hence, in order to model this random behaviour and allow exploration of the search space, the position of a randomly chosen search agent is updated instead of the best agent found so far. The mathematical model is as follows

\[
\ddot{D} = |\ddot{C} \cdot \ddot{X}_{rand} - \ddot{X}(t)| \quad 5.16
\]
\[
\ddot{X}(t + 1) = \ddot{X}_{rand} - \ddot{A} \ddot{D} \quad 5.17
\]

To ensure exploitation, \( \ddot{A} \) has random values either greater than 1 or less than -1.
Chapter 5 Classifier Optimisation

The pseudo-code of WOA is shown in Figure 5.8. The WOA algorithm starts with a population of random solutions. At each iteration, the search agents update their position with respect to the position of the randomly chosen search agent in the searching for prey phase, and with respect to the position of the best agent in the encircling phase.

```
Initialize the whale population, X*, i = 1,2, ...., n
Select the best search agent X*
while (t<maximum no. of iterations)
{
    update a, A, C, l and p for each search agent
    if (p<0.5)
        if (|A|<1)
            Update the position of the current search agent
            as in encircling phase
        else if(|A|≥1)
            Select a random search agent (X_rand)
            Update the position of the current search agent
            as in exploration phase
    end if
    else if(p≥0.5)
        Update the position of the current search agent as
        in exploitation (bubble net) phase
    endif
    Calculate fitness of each search agent
    Update X* if there is a better solution
    t=t+1;
}
```

Fig.5.8 WOA pseudo-code

The algorithm also chooses between shrinking encircling mechanism and the spiral model according to a randomly drawn probability measure. The algorithm effectively employs both exploration and exploitation. Adaptive variation of the search vector $\hat{A}$ allows the WOA algorithm to smoothly
transit between exploration and exploitation: by decreasing $A$, some iterations are devoted to exploration ($|A| \geq 1$) and the rest is dedicated to exploitation ($|A| \leq 1$). The algorithm is terminated when the termination criterion is reached.

The proposed SVM based underwater classifier was optimised with WOA. The results of the experiments for different population sizes are tabulated in Table 5-3. Each run of the algorithm consists of 500 iterations, except when the algorithm is terminated when the average relative change in fitness value is stalled over 25 generations.

Table 5-3 Performance results with WOA based optimisation for varying population size

<table>
<thead>
<tr>
<th>Population Size</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-score (class 1)</td>
<td>0.62</td>
<td>0.63</td>
<td>0.70</td>
<td>0.67</td>
</tr>
<tr>
<td>F-score (class 2)</td>
<td>0.73</td>
<td>0.73</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>F-score (class 3)</td>
<td>0.72</td>
<td>0.72</td>
<td>0.73</td>
<td>0.73</td>
</tr>
<tr>
<td>F-score (class 4)</td>
<td>0.80</td>
<td>0.81</td>
<td>0.82</td>
<td>0.82</td>
</tr>
<tr>
<td>F-score (class 5)</td>
<td>0.83</td>
<td>0.85</td>
<td>0.86</td>
<td>0.86</td>
</tr>
<tr>
<td>F-score (class 6)</td>
<td>0.81</td>
<td>0.81</td>
<td>0.82</td>
<td>0.82</td>
</tr>
<tr>
<td>F-score (class 7)</td>
<td>0.73</td>
<td>0.74</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>F-score (class 8)</td>
<td>0.79</td>
<td>0.83</td>
<td>0.83</td>
<td>0.83</td>
</tr>
<tr>
<td>F-score (class 9)</td>
<td>0.76</td>
<td>0.80</td>
<td>0.81</td>
<td>0.81</td>
</tr>
<tr>
<td>F-score (class 10)</td>
<td>0.63</td>
<td>0.63</td>
<td>0.65</td>
<td>0.65</td>
</tr>
<tr>
<td>F-score (class 11)</td>
<td>0.61</td>
<td>0.62</td>
<td>0.67</td>
<td>0.66</td>
</tr>
<tr>
<td>F-score (Average)</td>
<td>0.73</td>
<td>0.74</td>
<td>0.76</td>
<td>0.76</td>
</tr>
<tr>
<td>Overall Accuracy (%)</td>
<td>68.7</td>
<td>70.2</td>
<td>73.1</td>
<td>72.0</td>
</tr>
</tbody>
</table>
5.11 Stochastic Fractal Search (SFS)

Stochastic Fractal Search (SFS) algorithm is a meta-heuristic algorithm, formulated by Salimi [171] based on the random fractals observed in nature. Fractal refers to objects or quantities that display self similarity, in somewhat technical sense, in all dimensions [172]. Fractal shapes can be generated by common techniques such as iterated function systems, strange attractors, L-systems, escape time fractals, finite subdivision rules and random fractals. Random fractals can be created by physically motivated models such as diffusion limited aggregation (DLA) model. SFS algorithm is inspired from random fractals grown by DLA model, which can model clusters describing a bacterial colony. In DLA model, virtual particles moving through space following a random walk diffuse and stick together around a seed particle. A cluster is built up over time as more and more particles collide and clump together. While forming the cluster, the probability of the particle which sticks to the farthest end of the cluster is high in comparison to the one that penetrates the interior.

The problem solutions to an optimization problem are considered as individual particles in a population. SFS relies on three simple principles to find a solution

i. Each particle has an electrical potential energy which is updated based on its fitness value

ii. Each particle diffuses, and cause some other random particles to be created, and the energy of the seed particle is divided among generated particles

iii. The best generated particles from the diffusion process are considered, and the rest of the particles are eliminated
Two main processes to perform the SFS, are the diffusion process and the update process. In the diffusion process, the particles diffuse around its current position to form a cluster and in the updating process, the velocity and position of the particles is updated based on its position and the position of other particles.

The algorithm begins by randomly placing the particles at different locations in the search space. Each particle is initialized with equal energy obtained from equation 5.18.

\[ E_i = \frac{E}{p} \]  
5.18

where \( E \) is the maximum electrical potential energy considered to solve the problem.

After initialization, the fitness function of all particles is calculated and the best point \( BP \) is tracked. Each particle is then diffused in each generation, which creates new particles. The diffusion process is modelled through Gaussian walks as in equations 5.19 and 5.20.

\[ GW_1 = \text{Gaussian} (\mu_{BP}, \sigma) = (\varepsilon \cdot BP - \varepsilon' \cdot P_i) \]  
5.19

\[ GW_2 = \text{Gaussian} (\mu_p, \sigma) \]  
5.20

where \( \varepsilon \) and \( \varepsilon' \) are uniformly distributed random numbers in the range \([0,1]\). \( P_i \) is the \( i^{th} \) point, and \( BP \) is the best point in the cluster. \( \mu_{BP} \), \( \mu_p \) and \( \sigma \) are Gaussian parameters where \( \mu_{BP} \) is exactly equal to \( BP \) and \( \mu_p \) is equal to \( P \). For a particular generation \( g \), the standard deviation is computed as in equation 5.21.

\[ \sigma = \frac{\log(g)}{g} \cdot (P_i - BP) \]  
5.21
The term $\frac{\log(g)}{g}$ causes the size of Gaussian jumps to decrease as generations proceed, thereby forcing the algorithm to move closer to the solution. The diffusion process introduces the exploitation property of the algorithm and increases the chances of the algorithm in finding the optimal solutions.

After the diffusion process, all points (the total number being $N$) are ranked based on their individual fitness values, and each of them are assigned the probability of entering the next generation, the value of which follows a uniform distribution as in equation 5.22.

$$Pa_i = \frac{\text{rank}(P_i)}{N} \quad 5.22$$

As per the above equation 5.22, the individual with a higher rank will have a higher probability to be selected into the next generation.

For each point $P_i$ in the cluster, with $Pa_i < \varepsilon$, where $\varepsilon$ is a random value in the range [0,1], the $j^{th}$ component of $P_i$ is updated using the equation 5.23,

$$P'_i(j) = P_r(j) - \varepsilon \cdot (P_t(j) - P_i(j)) \quad 5.23$$

where $P'_i$ is the updated position of $P_i$, and $P_r$ and $P_t$ are randomly selected points in the group. All the points are again sorted based on their ranks calculated by equation 5.22. Again the position of points with $Pa_i<\varepsilon$, where $\varepsilon$ is a random value in the range [0,1], is updated according to equations 5.24 and 5.25.

$$P''_i = P'_i - \hat{\varepsilon} \cdot (P'_i - BP) \quad \text{for } \varepsilon' \leq 0.5 \quad 5.24$$

$$P''_i = P'_i + \hat{\varepsilon} \cdot (P'_i - BP) \quad \text{for } \varepsilon' > 0.5 \quad 5.25$$
where, $P_r$ and $P_i$ are randomly selected points and $\hat{a}$ are random numbers generated by the Guassian normal distribution. The new point $P_i''$ is replaced by $P_i'$ if its fitness function value is better than $P_i'$. The energy is distributed among the new particles according to their fitness value. The distribution energy equation for the selected particles is given by

$$E_i^j = \left[\left(\frac{f_j}{f_i + \sum_{k=1}^{n} f_k}\right)\right] \times E_i$$  \tag{5.26}

Where $f_j$ is the energy of the diffused particle and $f_i$ is its fitness value before diffusion and $E_i$ is calculated as per equation 5.26. The updating process contributes to the exploration property of the SFS algorithm.

The proposed SVM based underwater classifier was optimised with SOS algorithm. The results of the experiments for different population sizes are tabulated in Table 5-4. Each run of the algorithm consists of 500 iterations, except when the algorithm is terminated when the average relative change in fitness value is stalled over 25 generations.

Table 5-4 Performance results with SFS algorithm based optimisation for varying population size

<table>
<thead>
<tr>
<th>Population Size</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-score (class 1)</td>
<td>0.64</td>
<td>0.67</td>
<td>0.69</td>
<td>0.68</td>
</tr>
<tr>
<td>F-score (class 2)</td>
<td>0.67</td>
<td>0.69</td>
<td>0.70</td>
<td>0.69</td>
</tr>
<tr>
<td>F-score (class 3)</td>
<td>0.76</td>
<td>0.76</td>
<td>0.76</td>
<td>0.76</td>
</tr>
<tr>
<td>F-score (class 4)</td>
<td>0.90</td>
<td>0.91</td>
<td>0.93</td>
<td>0.91</td>
</tr>
<tr>
<td>F-score (class 5)</td>
<td>0.80</td>
<td>0.81</td>
<td>0.81</td>
<td>0.80</td>
</tr>
<tr>
<td>F-score (class 6)</td>
<td>0.81</td>
<td>0.82</td>
<td>0.82</td>
<td>0.82</td>
</tr>
<tr>
<td>F-score (class 7)</td>
<td>0.75</td>
<td>0.77</td>
<td>0.77</td>
<td>0.76</td>
</tr>
<tr>
<td>F-score (class 8)</td>
<td>0.79</td>
<td>0.80</td>
<td>0.80</td>
<td>0.79</td>
</tr>
</tbody>
</table>
F-score (class 9) | 0.77 | 0.77 | 0.77 | 0.77
--- | --- | --- | --- | ---
F-score (class 10) | 0.68 | 0.70 | 0.70 | 0.70
F-score (class 11) | 0.63 | 0.67 | 0.68 | 0.66
F-score (Average) | 0.75 | 0.76 | 0.77 | 0.76
Overall Accuracy (%) | 70.5 | 72.4 | 74.5 | 73.1

### 5.12 Symbiotic Organisms Search

Symbiosis is derived from two Greek words ‘sym’, which means ‘together’ and ‘bios’ which means ‘life’ and can be stated as close and often long term interaction and reliance between two variant biological species. Symbiotic relationships exist in organisms as a strategy for them to adapt with changes in environment, which may help them increase their fitness and survival advantage in the ecosystem over a long term. Symbiotic relationships can be obligate or facultative. Obligate symbiosis is when two organisms are in a symbiotic relationship because they can’t survive without each other. Facultative symbiosis is when the species live together by choice. Pollination symbiosis is an obligate symbiosis whereas cleaning symbiosis is a facultative symbiosis. There are three main types of symbiotic relationships: mutualism, commensalism and parasitism depending on the nature and character of interaction between the associated organisms. Cheng and Prayogo formulated Symbiotic organisms Search (SOS) algorithm based on symbiotic interaction strategies observed in nature [173]. Three phases are introduced in the SOS algorithm to resemble the biological interaction occurring in the three types of symbiosis.

SOS algorithm commences with a randomly generated initial population representing the initial ecosystem, in which each organism
corresponds to a candidate solution with an associated fitness value that reflects its survival cost in the ecosystem. In the initial ecosystem, a group of organisms are randomly generated within the search space. Each organism interacts with the other organisms randomly through all phases. This process is repeated for all the organisms in the ecosystems, until termination criteria are met. The pseudo-code of SOS algorithm is shown in Figure 5.9.

![Fig.5.9 SOS pseudo-code](image)

### 5.12.1 Mutualism Phase

Mutualism is a symbiotic relationship in which each individual benefit from the activity of the other, like the relationship between bees and flowers
in which the bee benefits from the nectar it gets from the flower and the flower benefits from the pollination carried out by the bees.

Mutualistic relation can be mathematically modelled as follows. An organism $X_i$ engages in a mutualistic relationship with another organism $X_j$ with the goal of increasing their mutual survival advantage in the ecosystem. Based on the mutualistic symbiosis between organisms $X_i$ and $X_j$, the new candidate solutions are calculated as follows

$$X_{i_{new}} = X_i + \text{rand}(0,1)(X_{best} - \text{Mutual Vector} \times BF_1) \quad 5.27$$

$$X_{j_{new}} = X_j + \text{rand}(0,1)(X_{best} - \text{Mutual Vector} \times BF_2) \quad 5.28$$

$$\text{Mutual Vector} = (X_i + X_j)/2 \quad 5.29$$

$BF_1$ and $BF_2$ are factors which determine the degree of benefit to each organism.

This phase aids in exploration of new regions, as organisms located far away in the search space are brought to interact by $\text{Mutual Vector}$. Further, the two interacting individuals are updated concurrently rather than singly.

**5.12.2 Commensalism Phase**

Commensalism is a symbiotic relationship in which one organism (the commensal) benefits, and the other is apparently unaffected (or receives minimal benefit) like that of algae and barnacles growing on turtles and whales. The new candidate solutions in this phase are found by exploiting promising regions around the best solution. This phase controls the convergence of the algorithm and is mathematically modelled as follows

$$X_{i_{new}} = X_i + \text{rand}(-1,1)(X_{best} - X_j) \quad 5.30$$
5.12.3 Parasitism Phase

Parasitism is a symbiotic relationship in which one organism (the parasite) benefits, at the expense of other such that the other organism is adversely affected, like fleas harming the hosts on which they live. To model this phase a Parasite Vector is created in the search space by duplicating organism $X_i$, and modifying it along a randomly selected dimension. Both the organisms are then evaluated, and the fitter organism is allowed to resume its position in the ecosystem. Parasitism phase allows exploration of the search space and may arrive at unique solutions that may be located in completely different regions of the search space.

The proposed SVM based underwater classifier was optimised with SOS algorithm. The results of the experiments for different population sizes are tabulated in Table 5-5. Each run of the algorithm consists of 500 iterations, except when the algorithm is terminated when the average relative change in fitness value is stalled over 25 generations.

Table 5-5 Performance results with SOS algorithm based optimisation for varying population size

<table>
<thead>
<tr>
<th>Population Size</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-score (class 1)</td>
<td>0.68</td>
<td>0.75</td>
<td>0.71</td>
<td>0.69</td>
</tr>
<tr>
<td>F-score (class 2)</td>
<td>0.76</td>
<td>0.78</td>
<td>0.78</td>
<td>0.76</td>
</tr>
<tr>
<td>F-score (class 3)</td>
<td>0.74</td>
<td>0.76</td>
<td>0.75</td>
<td>0.74</td>
</tr>
<tr>
<td>F-score (class 4)</td>
<td>0.88</td>
<td>0.89</td>
<td>0.89</td>
<td>0.88</td>
</tr>
<tr>
<td>F-score (class 5)</td>
<td>0.90</td>
<td>0.92</td>
<td>0.91</td>
<td>0.90</td>
</tr>
<tr>
<td>F-score (class 6)</td>
<td>0.78</td>
<td>0.79</td>
<td>0.78</td>
<td>0.78</td>
</tr>
<tr>
<td>F-score (class 7)</td>
<td>0.88</td>
<td>0.89</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td>F-score (class 8)</td>
<td>0.85</td>
<td>0.86</td>
<td>0.85</td>
<td>0.85</td>
</tr>
</tbody>
</table>
### Chapter 5 Classifier Optimisation

<table>
<thead>
<tr>
<th>F-score (class 9)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.85</td>
<td>0.87</td>
<td>0.87</td>
</tr>
<tr>
<td>F-score (class 10)</td>
<td>0.71</td>
<td>0.73</td>
<td>0.73</td>
</tr>
<tr>
<td>F-score (class 11)</td>
<td>0.69</td>
<td>0.73</td>
<td>0.70</td>
</tr>
<tr>
<td>F-score (Average)</td>
<td>0.79</td>
<td>0.82</td>
<td>0.80</td>
</tr>
<tr>
<td>Overall Accuracy (%)</td>
<td>77.5</td>
<td>80.4</td>
<td>78.9</td>
</tr>
</tbody>
</table>

#### 5.13 Improving the Parameter Optimisation by Modified - Symbiotic Organisms Search

The right balance between a meta-heuristic algorithm’s ability to explore and exploit determines its efficiency to arrive at the optimal solution. Exploitation ability is introduced in meta–heuristic algorithms usually by directing the search towards the best solution. In the proposed modified symbiotic organisms search (m-SOS) algorithm, a balanced exploitation is introduced through a cognitive component as well as a social component. The cognitive component is dependent on the best value of a particular organism ($X_{best}$) and the social component is dependent on the best value among all the organisms ($G_{best}$). A better coverage of the search space is achieved through the social component and the cognitive component directs the search towards the possibly best solution in the neighbourhood. The social component reduces the algorithms chance to get trapped in a local minima. In the proposed $m$–SOS algorithm another modification proposed is to introduce a weighted mutual vector. By introducing so, the factor Mutual Vector × BF in the SOS algorithm controls the degree by which the candidate solution approaches the best solution. The weights are determined by the fitness values obtained in the previous iterations. Hence, it may be considered as the momentum component which determines the impetus with which the candidate solution moves towards the best solution, based on the
previous experience in terms of the fitness value. The right balance between exploration and exploitation determines the efficacy of the algorithm. In m–SOS algorithm exploration is also introduced through the GA operators crossover and mutation. Randomly generated indices of the solutions $X_{inew}$ and $X'_{fnew}$ are subjected to crossover operators to generate new solutions $X'_{inew}$ and $X'_{fnew}$. The resulting solutions $X'_{inew}$ and $X'_{fnew}$ if found to have better fitness values are accepted, else $X_{inew}$ and $X_{fnew}$ are retained. In the parasitism phase, the organism is randomly mutated to generate the parasite vector. The algorithm can be described in detail as follows

### 5.13.1 Mutualism Phase

The new candidate solutions for $X_i$ and $X_j$ are calculated based on the mutualistic symbiosis between organism $X_i$ and $X_j$.

$$X_{inew} = X_i + \text{rand}(0,1)(X_{best} - \text{Mutual Vector} \times BF_1(G_{best} - X_i))$$

$$X_{fnew} = X_j + \text{rand}(0,1)(X_{best} - \text{Mutual Vector} \times BF_2(G_{best} - X_j))$$

$X_{best}$ is the best value of that particular organism, and $G_{best}$ is the best value among all organisms.

$BF_1$ and $BF_2$ are factors which determine the degree of benefit to each organism.

$$\text{Mutual Vector} = (w_1X_i + w_2X_j)/2$$

where $w_1 = \eta_{i-1}/100$ and $w_2 = \eta_{j-1}/100$

$\eta_{i-1}$ is the percentage of correctly classified samples in the previous iteration.
Chapter 5 Classifier Optimisation

\[ Mutual\ Vector = \left( w_1X_i + w_2X_j \right)/2 \]  
\[ \text{where } w_1 = \eta_{i-1}/100 \text{ and } w_2 = \eta_{j-1}/100 \]

\[ idx = [i_1, i_2, \ldots, i_M], M < N \]

\[ i_1, i_2, \ldots, i_M \] are random numbers between 1&N

\[ X_{CO} \in X_i \quad \forall x \in X_{CO} \to x \in X_i \]

\[ X_{CO}(i) = X_i(idx(i)), i = 1, \ldots, M \]

\[ Y_{CO} \in X_j \quad \forall y \in Y_{CO} \to y \in Y_i \]

\[ Y_{CO}(i) = X_j(idx(i)), i = 1, \ldots, M \]

\[ X' = X_i(idx(i)) = Y_{CO}(i) \]

\[ Y' = X_j(idx(i)) = X_{CO}(i) \]

The new solutions can be calculated as

\[ X'_{inew} = X' + \text{rand}(0,1)(X_{best} - X_i) \]

\[ - \quad \text{Mutual Vector} \times BF_1(G_{best} - X_i) \]

\[ X'_{jnew} = Y' + \text{rand}(0,1)(X_{best} - X_j) \]

\[ - \quad \text{Mutual Vector} \times BF_2(G_{best} - X_j) \]

accept the best solutions

\[ \text{if } (\text{fitness}(X'_{inew}) > \text{fitness}(X_{inew})) \]

\[ \text{accept } X'_{inew} \text{ else retain } X_{inew} \]

\[ \text{if } (\text{fitness}(X'_{jnew}) > \text{fitness}(X_{jnew})) \]

\[ \text{accept } X'_{jnew} \text{ else retain } X_{jnew} \]
5.13.2 Commensalism Phase

The new candidate solutions in this phase are calculated as

\[ X_{\text{inew}} = X_i + r\text{and}(-1,1)(X_{\text{best}} - X_j)(G_{\text{best}} - X_j) \]

The proposed SVM based underwater classifier was optimised with \( m\)-SOS algorithm. The results of the experiments for different population sizes are tabulated in Table 5.6. Each run of the algorithm consists of 500 iterations, except when the algorithm is terminated when the average relative change in fitness value is stalled over 25 generations.

Table 5.6 Performance results with the proposed \( m\)-SOS algorithm based optimisation for varying population size

<table>
<thead>
<tr>
<th>Population Size</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-score (class 1)</td>
<td>0.77</td>
<td>0.81</td>
<td>0.80</td>
<td>0.78</td>
</tr>
<tr>
<td>F-score (class 2)</td>
<td>0.81</td>
<td>0.84</td>
<td>0.84</td>
<td>0.82</td>
</tr>
<tr>
<td>F-score (class 3)</td>
<td>0.74</td>
<td>0.79</td>
<td>0.79</td>
<td>0.76</td>
</tr>
<tr>
<td>F-score (class 4)</td>
<td>0.90</td>
<td>0.91</td>
<td>0.91</td>
<td>0.90</td>
</tr>
<tr>
<td>F-score (class 5)</td>
<td>0.92</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
</tr>
<tr>
<td>F-score (class 6)</td>
<td>0.71</td>
<td>0.76</td>
<td>0.76</td>
<td>0.75</td>
</tr>
<tr>
<td>F-score (class 7)</td>
<td>0.93</td>
<td>0.95</td>
<td>0.95</td>
<td>0.94</td>
</tr>
<tr>
<td>F-score (class 8)</td>
<td>0.85</td>
<td>0.87</td>
<td>0.86</td>
<td>0.86</td>
</tr>
<tr>
<td>F-score (class 9)</td>
<td>0.84</td>
<td>0.88</td>
<td>0.88</td>
<td>0.85</td>
</tr>
<tr>
<td>F-score (class 10)</td>
<td>0.78</td>
<td>0.82</td>
<td>0.82</td>
<td>0.80</td>
</tr>
<tr>
<td>F-score (class 11)</td>
<td>0.79</td>
<td>0.84</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>F-score (Average)</td>
<td>0.82</td>
<td>0.85</td>
<td>0.85</td>
<td>0.84</td>
</tr>
<tr>
<td>Overall Accuracy (%)</td>
<td>82.5</td>
<td>85.1</td>
<td>88.4</td>
<td>83.3</td>
</tr>
</tbody>
</table>
5.14 Summary

This chapter throws light into optimisation of a classifier and the various concepts associated with it. Different optimisation strategies that can be adopted are briefed. Meta-heuristic optimisation algorithms strike a balance between exploration and exploitation of the search space. Meta-heuristic optimisation is adopted in this work for parameter optimisation. The various meta-heuristic optimisation algorithms adopted are described.

The results of adoption of optimisation algorithms to the proposed classifier, has improved its performance. Thus, it can be concluded that, tuning the algorithmic parameters to the optimum is essential for improving the classifier performance.
CONCLUSIONS

This thesis addresses one of the emerging topics in Sonar Signal Processing, viz. improving the performance of the target classifier for noise sources in the ocean. The underlying classifier implemented is a multi-class SVM based classifier. The main challenges faced by underwater target classification systems are due to diverse noise sources that vary with time and location. The performance of underwater classifiers can be improved by selecting the most representative feature vector that characterizes the signals and also by setting the optimal parameters of the underlying classifier. Different procedures for feature extraction has been studied and implemented for generating the feature vector. Procedures for dynamic feature selection according to changing underwater environment, has also been implemented resulting in an improvement in the classifier performance. The optimal choice of the classifier parameters, kernel function and kernel function parameters has been found by meta-heuristic algorithms. Different meta-heuristic algorithms have been implemented for parameter tuning of the underlying classifier, which has shown to improve the classifier performance. A modified-SOS (m-SOS) algorithm is also proposed which has shown to give higher performance compared to other algorithms implemented. This chapter brings out the salient highlights of the work along with enlisting the scope and direction for future research in this area.
6.1 **Highlights of the Thesis**

Underwater target recognition has gained considerable significance due to its strategic as well as commercial importance. The composite and dynamic nature of the propagation medium imposed by the ocean makes underwater target recognition a very challenging task. The ocean, as a propagation medium consists of dynamically varying composite noise sources comprising of man-made noises such as noise due to shipping, natural noises due to environment such as wind, waves, currents and rains, and biological noises emanated by underwater living organisms, that establish a perpetual noise backdrop. Underwater target activity reflected by the acoustic activity of the targets of interest, are captured by hydrophones. However, the hydrophones receive an acoustic mixture of requisite signals embedded with the ocean noise. Individual targets of interest are identified from hydrophone captured acoustic mixture, through their characteristic signatures that are patterned by feature recognition algorithms, which are then provided to the classifier for classification into different classes. In this work, a support vector machine (SVM) based target classifier is used to distinguish between targets of 11 classes. The work reported in the thesis entitled *Underwater Target Classifier with Improved Success Rate using Meta-Optimal Support Vector Machines* addresses one of the emerging topics in Sonar Signal Processing, viz. improving the performance of an underwater target classifier for noise sources in the ocean which is achieved through carefully selected feature vector and also through optimising the classifier parameters. The following are the salient highlights of this thesis.

6.1.1 **Need and Requirement of optimising the target classifier**

The introductory chapter of the thesis throws light on the various noise sources in the ocean as well as the need and requirement of optimising
the target classifier. The chapter also highlights the applications of underwater target classification. The underlying principle of operation of the proposed classifier is also briefly introduced in this chapter.

### 6.1.2 Preparation of a State-of-the-art Literature

The development of a classifier involves extraction of target specific acoustic signatures using suitable feature extraction and feature selection algorithms and adoption of a suitable classifier for classification. The performance of the classifier is improved through optimisation of the classifier parameters. As prelude to the development of a classifier, a state-of-the-art literature survey has been prepared on various aspects such as the underwater acoustic environment, acoustic feature extraction techniques, and various classifiers such as the statistical classifiers, lazy learning algorithms, decision tree classifiers, neural network classifiers and support vector machines. Literature review has also been prepared on different optimization techniques that can be used for improving the classifier performance.

### 6.1.3 Feature Vector Based Classifier

The methodology suggested to be adopted for realizing the proposed classifier involves the formulation of the acoustic signature of the targets of interest using suitable feature extraction techniques. Various acoustic feature extraction schemes have been highlighted in the thesis. Cepstral based techniques have been found to give better classification performance. To reduce the dimensionality of the feature vector by removing redundant and irrelevant features, feature selection algorithms are used. A reduced but highly representative feature vector will lower the complexity of the classifier. Various feature selection algorithms have been attempted on the proposed classifier and their performance is analysed. The feature vectors of known targets are labelled according to their classes to create a knowledge
base for the classifier. The classifier uses the labelled signals in the knowledge base to compare against the feature vector of an unknown signal, based on which the system performs the decision-making process.

### 6.1.4 SVM Based Multi-class Classifier

Once the feature vector has been extracted and the knowledge base is created, a suitable classifier needs to be identified to do the final classification task. In this work an SVM based classifier, which can learn non-linear decision surfaces efficiently, is adopted. SVMs are relatively easy to implement and very robust due to its sound theoretical background. They also have the advantage of creating a model with minimized Vapnik–Chervonenkis (VC) dimension, resulting in a low expected probability of error and thus good generalization performance. SVM’s were originally proposed as binary classifiers, but were later extended to solve multi-class problems by decomposing the multi-class problem into simple binary classification problems. Two popular approaches of solving multi-class problems are, one-against-one approach and one-against-all approach. In this work a multi-class SVM based target classifier for classifying 11 classes of acoustic targets is developed using one-against-all approach.

### 6.1.5 Parameter Optimisation of the classifier

The algorithmic parameters of the classifier impact its performance. Particularly, for an SVM based classifier, which is acclaimed for its high generalization capabilities, setting the right kernel parameters is a very determining factor in the classifier performance. Hence, attempting to set the right classifier parameters results in its performance improvement. Since, the underwater environment is highly dynamic in nature with changing channel properties, dynamic selection of algorithmic parameters, kernel and kernel parameters are required. Optimization algorithms are resorted to, for
scanning the parameter space to determine the most suitable set of parameters. In this work, parameter optimization is attempted using five meta-heuristic algorithms, namely GA, BAT, WOA, SFS and SOS. Though all the optimization algorithms have shown to improve the performance of a multi-class SVM based classifier, SOS algorithm exhibited superior performance improvement over others. A modification to SOS algorithm is also proposed which is called $m$-SOS (modified-SOS) algorithm. The $m$-SOS algorithm has shown to better improve the performance of the classifier in selecting the optimal parameter setting for the underlying classification task.

6.2 Future Scope for Research

The work presented in this thesis has a significant role to play in view of its practical applications. This work also has substantial scope for further research for improving the overall system performance. Some of the possible proposals for future work in this area are enlisted below.

6.2.1 Expansion of Knowledge Base

The proposed prototype system for identifying the noise sources in the ocean works on a simulated environment with a limited data set. By expanding the knowledge base, more training data can be obtained. A well trained classifier will certainly yield better performance in actual environment. Attempts were made to obtain more data from the Indian seas, but could not succeed to the expectations, and so could be taken up as a separate major project in collaboration with appropriate funding agencies.

6.2.2 Hardware Implementation

The proposed SVM based classifier works on a simulated environment and the modules have been developed in Matlab. The hardware
version of the system can be developed using high-end Digital Signal Processors, and FPGA systems.

6.2.3 Augmentation of Feature Vector

The performance of the classifier can be improved by augmenting the feature vector used, with higher order features such as bispectrum and trispectrum which are based on third and fourth order statistics respectively. However, as the order increases, the computational complexity and storage requirement also increases, which necessitates the requirement of efficient hardware systems for their implementation.

Most feature extraction techniques based on cepstral analysis are based on auditory models of human ear. Developing a feature extraction technique based on the auditory model of marine mammals may result in better signature features for underwater targets. Incorporation of features motivated by auditory models of marine mammals may also be worked up on.

6.2.4 Incorporating Meta-meta optimal SVMs

Meta-optimization refers to employing an optimization algorithm to optimize the parameters of another algorithm. In this work, we are employing different meta-heuristic algorithms to optimize the parameters of SVM. Meta-meta-optimization can be resorted to, for optimizing the parameters of the meta-heuristic algorithm which is optimizing the parameters of the SVM classifier. Parameter free algorithms such as Teaching Learning Based Optimisation (TLBO) can be attempted as the top level algorithm for meta-meta-optimization, for optimising the parameters of the meta-heuristic algorithm which is optimising the parameters of underlying SVM based classifier.
6.3 Summary

An attempt has been made in this chapter to bring out the salient highlights of the work carried out for the implementation of an underwater target classifier with improved success rate using meta-optimal SVMs. A discussion on the scope and directions for future research works in this area has also been presented.


Bibliography


Underwater Target Classifier with Improved Success Rate using Meta-Optimal Support Vector Machine

Improve Classification Accuracy,” 2015.


214


List of Publications


The Supervised Learning Model

The supervised learning model can be described using three components:

i. a generator of random vectors $x$, drawn independently from a fixed but unknown distribution $P(x)$;

ii. a supervisor that returns an output $y$, the class label for each $x$ in the test dataset, according to a conditional distribution function $P(y|x)$, also fixed but unknown;

iii. a learning machine capable of implementing a set of functions $f(x, \alpha)$, $\alpha \in \Lambda$.

The problem of learning is that of choosing from the given set of functions $f(x, \alpha)$, $\alpha \in \Lambda$, the one which predicts the class label of the input in the best possible way. The selection is based on a training set of $l$ random independent identically distributed (i.i.d.) observations $(x_1, y_1), \ldots, (x_l, y_l)$, drawn according to $P(x, y) = P(x) \ P(y|x)$. The function $f(x, \alpha)$, is called a predictor, a hypothesis or a classifier. The response of the algorithm, measured by the error of the classifier is the probability that it does not predict the correct label $y$ on a random input $x$, which cannot be calculated since the underlying distribution is unknown. The best available approximation to the response of the learning algorithm is a loss functional which measures the loss or disagreement $L(y, f(x, \alpha))$ between the response of the learning machine to a given input $x$, which is a randomly drawn new sample. The expected value of loss is given by the risk functional also called the expected risk, and is defined by equation A.1.
Appendix

\[ R(\alpha) = \int L(y, f(x, \alpha)) dP(x, y) \]  

A.1

The goal is to find the function \( f(x, \alpha) \) which minimises the risk functional \( R(\alpha) \) from the class of functions \( f(x, \alpha), \alpha \in \Lambda \), in the situation where the joint probability distribution is unknown and the only available information is contained in the training set.

**Statistical Learning Theory**

According to Vapnik, a pioneer in statistical learning theory and the developer of SVM, there are three main problems in machine learning: pattern recognition, regression estimation and density estimation. In all cases, the goal is to choose a model from the hypothesis space, which is closest (with respect to some error measure) to the underlying function in the target space.

Considering the problem of binary pattern recognition, the principle behind classification. The classifier’s output \( y \) can take only two values \( y = \{0, 1\} \), and let \( f(x, \alpha), \alpha \in \Lambda \), be the set of models or indicator functions. The classification problem can be expressed as

\[ L(y, f(x, \alpha)) = \begin{cases} 
0, & \text{if } y = f(x, \alpha) \\
1, & \text{if } y \neq f(x, \alpha)
\end{cases} \]  

A.2

The general setting of the learning problem can be described as follows. Let the underlying probability distribution of data, \( P(z) \) be defined on the space \( Z \). Consider the set of functions \( Q(z, \alpha), \alpha \in \Lambda \). The goal is to minimise the risk functional

\[ R(\alpha) = \int Q(z, \alpha) dP(z), \quad \alpha \in \Lambda \]  

A.3
where the probability measure $P(z)$ is unknown. However the i.i.d. sample $z_1, \ldots, z_n$, where $z$ describes a pair $(x,y)$ is available.

The learning problem considered minimises the risk functional given by equation A.3 where $Q(z,\alpha)$ is the specific loss function, on the basis of empirical data.

For the above classification problem, the risk functional given by equation A.3 requires the probability of classification error. Since the underlying probability distribution is unknown, the problem therefore is to find the model which minimises the classification error when only the data is known, and the underlying probability distribution is unknown.

**Empirical Risk Minimization Induction Principle**

As mentioned earlier, a typical machine learning task requires the data to be divided for training, validation and testing phases. During the training phase, the learning algorithm receives as input, a labeled training set sampled from an unknown distribution. The goal of the learning algorithm is to find the approximation (hypothesis) that minimises the error with respect to an unknown probability distribution and unknown dependency between the input and the output. The error of the classifier is the probability that it does not predict the correct label on a random data point generated by the underlying distribution. Since we do not know the underlying probability distribution and the dependency between the input and output, the expected value of error given by equation A. cannot be calculated. However, we can measure the empirical error or the training error, which is the average error incurred by the learning model due to wrong classification of samples drawn from already trained data. Hence, in order to minimise the risk functional, for an unknown probability measure, the empirical risk minimization (ERM)
Appendix

induction principle is usually used. The expected risk functional given by equation A. is replaced by the empirical risk functional

\[ R_{\text{emp}}(a) = \frac{1}{t} \sum_{i=1}^{t} Q(z, \alpha_i) \]  

constructed on the basis of i.i.d. training set.

The principle is to approximate the function \( Q(z, \alpha_0) \) which minimises risk specified by equation A. by the function \( Q(z, \alpha_1) \) which minimises empirical risk dictated by equation A.. This principle is called the Empirical Risk Minimization (ERM) induction principle. The difference between the expected and empirical risk is called generalization error and denotes the difference between error on the training set and error on the underlying joint probability distribution. It is a measure of how accurately an algorithm is able to predict outcome values for previously unseen data.

**Consistency of Empirical Risk Minimization Principle**

The ERM principle is consistent for a set of functions \( Q(z, \alpha), \alpha \in \Lambda \) and for the probability distribution function \( F(z) \) if the following two sequences converge in probability to the same limit.

\[ R(\alpha_i) \xrightarrow{l \to \infty} \inf_{\alpha \in \Lambda} R(\alpha) \]  

\[ R_{\text{emp}}(\alpha_i) \xrightarrow{l \to \infty} \inf_{\alpha \in \Lambda} R(\alpha) \]

In other words, the ERM method is consistent if it provides a sequence of functions \( Q(z, \alpha_i), l = 1, 2, \ldots \) for which both expected risk and empirical risk converge to the minimal possible value of risk \( \inf_{\alpha \in \Lambda} R(\alpha) \) as depicted in Figure A.1. Equation A. asserts that the values of achieved risks converge to the best possible and equation A. asserts that we can estimate the minimal possible value of risk on the basis of the values of empirical risk.
The above definition does not exclude trivial cases of consistency, that depend on whether the given set of functions contains a minorizing function. Therefore, in order to create a theory of consistency of the ERM method that would not depend on the properties of the elements of the set of functions, but would depend only on the general properties of this set of functions, the definition of consistency is to be modified to exclude the trivial consistency cases.

The modified definition for consistency states that the ERM method is nontrivially consistent for the set of functions $Q(z, \alpha), \alpha \in \Lambda$ and the probability function $P(z)$, if for any nonempty subset $\Lambda(c), c \in (-\infty, \infty)$ of this set of functions defined as

$$\Lambda(c) = \{\alpha: \int Q(z, \alpha) dF(z) > c, \alpha \in \Lambda\} \quad A.7$$

the convergence

$$\inf_{\alpha \in \Lambda(c)} R_{\text{emp}}(\alpha) \xrightarrow{\ell \to \infty} \inf_{\alpha \in \Lambda(c)} R(\alpha) \quad A.8$$
Appendix

is valid. In other words, the ERM is nontrivially consistent if it provides convergence for the subset of functions that remain after the functions with the smallest values of risks are excluded from this set.

The conditions for non trivial consistency are described by the theory of consistency of learning process which describes the necessary and sufficient conditions for convergence of the ERM inductive principle.

**The Theory of Consistency of Learning Process**

The theory of consistency is an asymptotic theory. It describes the necessary and sufficient conditions for the convergence of solutions obtained using ERM method to the best possible as the number of observations is increased.

**The Key Theorem of the Learning Theory**

The Key theorem of the learning theory was proposed by Vapnik and Chervonenkis in 1989. It advocates that the conditions required for the consistency of ERM principle are equivalent to the conditions for existence of uniform one sided convergence.

The Key theorem is stated as follows. Let $Q(z,\alpha), \alpha \in \Lambda$ be a set of functions that as a bounded loss for probability measure $P(z)$.

\[
A \leq \int Q(z,\alpha) dP(z) \leq B \quad \forall \alpha \in \Lambda \tag{A.9}
\]

Then for the ERM principle to be consistent, it is necessary and sufficient that the empirical risk $R_{\text{emp}}(\alpha)$ converge uniformly to the actual risk $R(\alpha)$ over the set $Q(z,\alpha), \alpha \in \Lambda$ as follows

\[
\lim_{l \to \infty} \text{prob} \{ \sup_{\alpha \in \Lambda} (R(\alpha) - R_{\text{emp}}(\alpha)) > \varepsilon \} = 0 \quad \forall \varepsilon \tag{A.2}
\]
This type of convergence is called uniform one sided convergence. The theorem is called the key theorem because it asserts that the conditions for consistency of ERM principle are necessarily (and sufficiently) determined by the worst function over the set of functions given by equation A.3, converges in probability to zero.

\[
\Delta(\alpha_{\text{worse}}) = \sup_{\alpha \in \Lambda} (R(\alpha) - R_{\text{emp}}(\alpha))
\]

Therefore with the above condition, from this theorem it follows that the analysis of the ERM principle requires an analysis on the properties of uniform convergence of the expectations to their probabilities over a given set of functions.

**Conditions for Uniform Convergence**

For uniform two sided convergence of the frequencies to their probabilities

\[
\lim_{l \to \infty} \text{Prob}\left\{ \sup_{\alpha \in \Lambda} |R(\alpha) - R_{\text{emp}}(\alpha)| > \epsilon \right\} = 0
\]

where \( R_{\text{emp}}(\alpha) \) defines frequency and \( R(\alpha) \) defines probability, it is necessary and sufficient that the equality defined in equation A.5 be valid.

\[
\lim_{l \to \infty} \frac{H^\Lambda(l)}{l} = 0, \quad \forall \epsilon > 0
\]

where \( H^\Lambda(l) \) is the expectation of random entropy over the joint distribution function \( P(z_1, \ldots, z_l) \)

\[
H^\Lambda(l) = E \ln N^\Lambda(z_1, \ldots, z_l)
\]

where \( N^\Lambda(z_1, \ldots, z_l) \) is the number of different vertices of the \( l \)-dimensional cube that is obtained on the basis of sample \( z_1, \ldots, z_l \) and the set of functions \( Q(z, \alpha), \alpha \in \Lambda \). In other words, \( N^\Lambda(z_1, \ldots, z_l) \) represents the number of different separations of the sample that can be obtained using functions from the given set of indicator functions.
Equation A.5 describes the necessary and sufficient condition for consistency of the ERM principle, which should be satisfied by any learning machine minimizing the empirical risk.

Conditions for Fast Convergence of the ERM Principle

The annealed VC entropy $H^A_{\text{ann}}(l)$ for sets of indicator functions is defined as

$$H^A_{\text{ann}}(l) = E \ln N^A(z_1, \ldots, z_l)$$

and the growth function $G^A(l)$ is defined as

$$G^A(l) = \ln \sup_{z_1, \ldots, z_l} N^A(z_1, \ldots, z_l)$$

These functions are determined in such a way that for any $l$ inequalities, $H^A(l) \leq H^A_{\text{ann}}(l) \leq G^A(l)$ is valid. The asymptotic rate of convergence is fast if for any $l > l_0$ dictated by equation A.9 holds true.

$$P\{R(\alpha_l) - R(\alpha_o) > \varepsilon\} < e^{-c\varepsilon^2l}$$

where $c$ is a constant greater than zero.

The sufficient condition for fast convergence is given by equation A.

$$\lim_{l \to \infty} \frac{H^A_{\text{ann}}(l)}{l} = 0$$

The necessary and sufficient condition for fast convergence is given by equation A..

$$\lim_{l \to \infty} \frac{G^A(l)}{l} = 0$$

The above condition specified in equation A. is independent of probability distribution and also describes the necessary and sufficient condition for consistency of ERM for any probability measure.
Equation A. describing the sufficient condition for fast convergence and equation A.5 describing the necessary and sufficient condition for consistency are valid only for a given probability measure $P(z)$. However, equation A describes the necessary and sufficient conditions for consistency of learning machine implementing ERM as well as the sufficient condition for fast convergence.

**VC Dimension**

The growth function $G^A(l)$ has a remarkable property that it either satisfies the equality specified by equation A. or is bounded by the inequality specified by equation A.

\[
G^A(l) = l \ln 2 \quad \text{A.20}
\]

\[
G^A(l) < h \left( \ln \frac{l}{h} + 1 \right) \quad \text{A.21}
\]

where $h$ is an integer for which

\[
G^A(h) = h \ln 2 \quad \text{A.102}
\]

\[
G^A(h + 1) \neq (h + 1) \ln 2 \quad \text{A.11}
\]

In other words, the growth function will be either a linear function or will be bounded by a logarithmic function. This is depicted in Fig. A..

The VC dimension of the set of indicator functions $Q(z, \alpha), \alpha \in \Lambda$ is infinite if the Growth function for this set of functions is linear. An alternate way of defining VC dimension is as follows. The VC dimension of a set of indicator functions $Q(z, \alpha), \alpha \in \Lambda$ is the maximum number $h$ of vectors $z_1, \ldots, z_h$ which can be separated in all $2h$ possible ways. If for any $n$ there exists a set of $n$ vectors which can be shattered by the set, then the VC dimension is equal to infinity.
Fig. A.2 The growth function is either linear or bounded by a logarithmic function. It cannot, for example, behave like as in dashed line.

**Distribution Independent Bounds for the Rate of Convergence of Learning Process**

Consider a set of totally bounded loss functions $Q(z, \alpha), \alpha \in \Lambda$, which possess a finite VC dimension $h$ described by equation A.24.

$$0 \leq Q(z, \alpha) \leq B, \quad \alpha \in \Lambda$$  \hspace{1cm} A.12

By the theory of bounds for sets of totally bounded functions the inequality in equation A.13 holds true with a probability of at least $1 - \eta$ for all functions described by equation A.12.

$$R(\alpha) \leq R_{emp}(\alpha) + \frac{B\varepsilon}{2} \left( 1 + \sqrt{1 + \frac{4R_{emp}(\alpha)}{B\varepsilon}} \right)$$  \hspace{1cm} A.13
Underwater Target Classifier with Improved Success Rate using Meta-Optimal Support Vector Machine

where \( \varepsilon = 4 \frac{h(\ln \frac{2l}{\eta} + 1) - \ln \eta}{l} \)

The theory of bounds provides bounds for risks of all indicator functions for \( Q(z, \alpha), \alpha \in \Lambda \), described by equation A.146.

\[
I(z, \alpha, \beta) = \theta\{Q(z, \alpha) - \beta\}, \quad \alpha \in \Lambda \tag{A.14}
\]

where \( \alpha < \beta < A \) is some constant and \( \theta(u) \) is a step function.

The bounds follow from the bound on uniform convergence for sets of totally bounded functions that have finite VC dimension.

**Structural Risk Minimization Induction Principle**

The ERM principle is intended for dealing with a large sample size. The sample size \( l \) is considered to be large if the ratio, \( ll/h \) (ratio of the number of training patterns to the VC dimension of functions of a learning machine) is large. The ERM principle can be justified by considering the inequalities in equation A.13. When \( ll/h \) is large, the second summand on the right side of inequality in equation A.13 becomes small. The actual risk is then close to the value of empirical risk. A small value of empirical risk provides a small value of expected risk.

However, if \( ll/h \) is small, a small \( R_{emp}(\alpha_l) \) does not guarantee a small value of the actual risk. In this case, to minimise the actual risk \( R(\alpha) \), one has to minimise the right hand side of inequality in equation A.13 simultaneously over both terms, one which depends on the value of the empirical risk while the second depends on the VC dimension of the set of functions. However, the ERM principle does not consider the capacity of the learning machine and hence tends to overfit the data. Predictions done by ERM are often unable to generalise well and prone to overfitting. This is due to the fact that
there can be infinitely many functions that have minimal risk, amongst which a single unique function will have the highest generalization ability. Hence, in order to develop a model that has good generalization capabilities and works well on previously unseen data, a principle which takes into account, the capacity of the learning machine should be developed. Thus the minimization of $R(\alpha)$ requires a new principle, based on the simultaneous minimization of two terms in equation A.13. To minimise risk, it is necessary to find a method which, along with minimizing the value of empirical risk, controls the VC dimension of the learning machine. Structural Risk Minimization (SRM) principle which is described in the following, is intended to minimise the risk functional with respect to both empirical risk and VC dimension of the set of functions.

Fig. A.3 A structure on the set of functions is determined by the nested subsets of functions

Let the set $S$ of functions $Q(z, \alpha), \alpha \in \Lambda$, be provided with a structure consisting of nested subsets of functions $S_k = \{ Q(z, \alpha), \alpha \in \Lambda_k \}$ as depicted in Fig. A., such that $S_1 \subset S_2 \subset \ldots \ldots \subset S_n$ and $S^* = \bigcup_k S_k$.

An admissible structure is one satisfying the following three properties

i. The set $S^*$ is everywhere dense in $S$

ii. The VC dimension $h_k$ of each set $S_k$ of functions is finite
iii. Any element $S_k$ of the structure contains totally bounded functions,
$$0 \leq Q(z, \alpha) \leq B_k, \alpha \in \Lambda_k$$

For a given set of observations $z_1, \ldots, z_l$ the SRM principle chooses the function $Q(z, \alpha^k)$ minimizing the empirical risk in the subset $S_k$ for which the guaranteed risk is minimal.

![Diagram](image)

Fig. A.4 Bound on the risk: The bound on the risk is the sum of empirical risk and the confidence interval. The empirical risk decreases with the index of the element of the structure, while the confidence interval increases.

The SRM principle defines a trade off between the quality of the approximation of the given data and the complexity of the approximating function. As the subset index $n$ increases indicating an increase in the complexity of the learning machine, the minima of the empirical risks decrease. However, the term responsible for the confidence interval (the second summand in inequality of equation A.13) increases. This is depicted in Fig. A.. The SRM principle takes both factors into account by choosing
Appendix

the subset $S_n$ for which minimizing the empirical risk yields the best bound on the actual risk. For any distribution function the SRM method provides convergence to the best possible solution with probability one. A learning machine implementing SRM will yield a classifier with minimal model complexity and risk. In other words SRM method is universally strongly consistent.
ADDENDUM

The addendum to the thesis entitled ‘Underwater Target Classifier with Improved Success Rate using Meta-Optimal Support Vector Machines’ contains relevant information about the thesis, which was inadvertently omitted during the preparation of the thesis.

Sources of Data

The database used in the classifier consists of noises of 11 classes of acoustic targets. Data of different targets have been recorded during scheduled cruises conducted off Cochin and Mangalore. For incorporating the channel effects and the environment, ambient noise has also been recorded. Other recordings have been collected from the open source databases available in the internet.

Man Made Noises

(a) Ships:

For the proposed study, three sound signatures of ships have been collected and labelled as Ship1, Ship2 and Ship3. The noise Ship1 is a recording of a commercial ship cruising at approximately 20 knots and about 3.2 km away from the hydrophone and Ship2 is a recording of a Merchant vessel in the Cochin Shipyards which was recorded as the vessel was approaching from 1.7 km away. The noise Ship3 was collected from an open source database in the internet.

(b) Boats:

Small boats such as Zodiac, have outboard motors whose propellers creates sound and these boats are popular in coastal waters. The propeller
produces a cavitation noise which is at higher frequencies than larger vessels, as they have high rotation rates. The database in the thesis included noises from 4 boats, labelled as Boat1, Boat2, Boat3 and Boat4. The noise Boat1 and Boat2 are recordings of a Zodiac with a 35 HP engine and 50HP engine respectively. The noises Boat3 and Boat4 were collected from open source databases in the internet.

(c) **Humpback Whale:**

Humpbacks are best known for their vocalizations and is one of rorquals which have two characteristics in common, viz. dorsal fins on their back, and ventral pleats running from the tip of the lower jaw back to the belly area. The noise of humpback whale has been collected from an open source database in the internet.

(d) **Sealion:**

Sea lions are sea mammals of the family Otariidae. Sea lions haul out in large colonies on rocks and sandy shores on the Islands. The noise of sealions has been collected from an open source database in the internet.

(d) **Snapping Shrimps:**

Snapping shrimp produce sound by snapping of their claws. The sound produced by a shrimp colony can be so loud that the sonars may miss other nearby targets. The noise produced by a shrimp colony has been collected from an open source database in the internet as well as recorded using hydrophones in the coastal shores of Vypeen, Kochi.

**Classifier training and testing for varying SNR**

The classifier has been trained with single signal at a time. For the testing phase, the classifier is fed with a simulated signal which is actually
an additive combination of the target signal and the ambient noise recorded from the sea while the team had gone for sea trials. The test bench created had different inputs like simulated signals, the target signal corrupted by ambient noise collected from Arabian sea as well as pure signals.

For incorporating the channel effects and the environment, this ambient noise has been additively added to the target signal noise and all the algorithms has been tested for the simulated signal which is a combination of original signal and ambient noise. Since it was the real data collected during the sea trial that was added, performance of the classifier under other noise conditions with varying SNR has not been studied.

**Performance in Active Scenario**

The prototype classifier has been developed for a passive sonar scenario. In active sonar scenario, factors like Reflection, Reverberation, Multipath, Scattering, Doppler effects etc. will be influencing the target signal strength. The underwater channel effects will have to be considered and signals will have to be appropriately pre-processed before passing on to the classifier. The basic methodology of the proposed classifier is expected to yield good results in the active scenario also. However, this scenario has not been considered in the thesis.

**Population Size**

The optimal population size of a meta-heuristic algorithm is dependent on a number of factors including the number of generations and the problem to which the algorithm is applied. Commonly for low dimensional optimization tasks ($d < 100$), a population size $n > d$ is adopted as mentioned in O. Roeva et al. in the paper *Influence of the Population Size on the Genetic Algorithm Performance in Case of*
Appendix

*Cultivation Process Modelling*. S. Chen *et al.* in the paper *Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution*, had studied the effect of the influence of population size on the GA performance in which he had observed that increasing the optimal population size for a fixed number of iterations does not improve the quality of solutions. The same has been observed in our work. The stopping criteria of all the optimization algorithms viz. GA, BAT, WOA, SFS, SOS and $m$-SOS employed in this thesis, was either 500 iterations or when the average relative change in the fitness value is stalled over 25 iterations. While the prototype was implemented, the optimal population size of GA, WOA, SFS and $m$-SOS was found to be 20 and that for BAT and SOS was found to be 15. The optimal population size can vary for a different stopping criteria and a different task.

*Computational Complexity*

The simulation has been carried out on a system with Intel Core i3 CPU (M350@2.27GHz) with 4 GB RAM installed with 64-bit Windows 7 Professional Operating System. As we have followed a supervised classification scheme, the complete procedure involves training and testing phase. The training phase involves a considerable computational cost which included the time for selection of features. However, during the testing phase, the system can achieve near real time performance as the SVM classifier is already trained.
The author would also like to give courtesy to the following figures which have been reproduced from textbooks and articles:

Fig. 1.2 A. D. Waite/ Sonar for Practising Engineers, John Wiley & Sons

Fig. 3.6 C. Neumuller et al./ Proceedings of the 13th International Conference on Computer Aided Systems Theory, 1(2011), pp.367-374.

Fig. 4.32 Juan Tapia Farias/ Article entitled ‘Feature Selection Methods’ available in the internet

Fig. 4.33 R. Kohavi, G. H. John/ Artificial Intelligence, 97(1997), pp.273-324.

Fig. 4.38 T. Hastie et al./ The Elements of Statistical Learning, Springer Series in Statistics

Fig. 4.43 Ben Aisen/ Article entitled ‘A Comparison of Multiclass SVM Methods’, (2016) available in the internet

Fig 4.44 Ben Aisen/ Article entitled ‘A Comparison of Multiclass SVM Methods’, (2016) available in the internet

Fig. 5.1 Leon fedden/ Article entitled ‘The No Free Lunch Theorem’ available in the internet

Fig. 5.2 S. Mirjalili, A. Lewis/ Advances in Engineering Software, 95(2016), pp.51-67.

Fig. 5.7 S. Mirjalili, A. Lewis/ Advances in Engineering Software, 95(2016), pp.51-67.

Fig. A.1 V. N. Vapnik/ The Nature of Statistical Learning Theory, Springer
Appendix

Fig. A.2  V. N. Vapnik/ The Nature of Statistical Learning Theory, Springer

Fig. A.3  V. N. Vapnik/ The Nature of Statistical Learning Theory, Springer

Fig. A.4  V. N. Vapnik/ The Nature of Statistical Learning Theory, Springer