DEVELOPMENT OF AN AUTOMATED IMAGE ANALYSIS SYSTEM TO DETECT BELUGA WHALES IN AERIAL PHOTOGRAPHS

JASON MILLS

Development of an Automated Image Analysis System to Detect Beluga Whales in Aerial Photographs

A thesis submitted to the
School of Graduate Studies in partial
fulfilment of the requirements for the degree of
Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

October 18, 2006

St. John's Newfoundland

Your file Votre référence
ISBN: 978-0-494-30489-1
Our file Notre référence
ISBN: 978-0-494-30489-1

Abstract

One aspect of monitoring the population of beluga whales, and other marine mammal species, is counting a sample of the population from aerial photographs (or negatives). Using image processing and pattern recognition techniques, a software system for detecting and classifying beluga whales in digitized aerial photographs and negatives is developed. The image processing component includes algorithms to create a mask to cover "unreadable" areas (e.g. land and sun glare), segment whales, and generate feature data for segmented objects. The segmented objects are classified and presented to the user in an interactive GUI (graphical user interface) for final conformation and quality control.

A fundamental step in developing a good pattern recognition system is to choose and optimize a classifier. To this end, the support vector machine (SVM) classifier is compared against a traditional quadratic discriminate classifier. To optimize the classifiers, a genetic algorithm (GA) for feature selection and classifier parameter calibration is used. An obstacle in applying GAs to any problem is selecting values for the fundamental GA control parameters. This is addressed using design of experiments (DOE) to systematically analyze the GA and derive a statistical model from which the parameters can be calculated. It is demonstrated that GAs are a good method to optimize SVMs via feature subset selection and SVM parameter calibration.

Acknowlegements

I owe a debt of gratitude to many people who have helped to make this work a success. First, I like to thank Dr. Raymond Gosine and Dr. George Mann for supervising me in this work. I very much appreciate the time, knowledge, and wisdom they so willing shared with me. Thanks for your patience and the freedom alloted to me to explore my own avenue of research no matter how crazy.

Special thanks to Dr. Garry Stenson at the Department of Fisheries and Oceans (DFO, Newfoundland and Labrador region) for initiating this project and providing the principal source of funding. I am also thankful for his help whenever asked and his infinite patience and kindness while conducted my (part-time) research. This work would not have been completed without his support. I also would like to thank Dr. Mike Hammill of DFO (Quebec region) for providing the initial funding and for providing the aerial photograph data set.

I am extremely grateful to C-CORE for providing the facilities, work environment, and time necessary to complete my work.

I must acknowledge my colleagues Stefan Tarrant (thanks for sharing you knowledge and implementation of GAs), Carl Howell (I can't count the number of discussions we had, all of which have been enlightening and entertaining), and Bax Smith (who introduced me to the world of SVMs) for help, advice, and friendship.

Last, but certainly not least, to my wife Deanne: I am forever grateful for you patience, support, and encouragement.

Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations and Symbols

2FI	2 factor interaction
2QD	2-class quadratic discriminate classifier type
2SVM	2-class support vector machine classifier type
3QD	3-class quadratic discriminate classifier type
3SVM	3-class support vector machine classifier type
API	Application programming interface
BBD	Box-Behnken design
BMP	Bitmap image file format
CCD	Central composite design
CD-ROM	Compact disc - read only memory
CV	Coefficient of variation in a regression model
DB	Data base
DOE	Design of experiments
ER	Error rate
FSS	Feature subset selection
GA	Genetic algorithm
GUI	Graphical user interface
JPEG	Joint photographic experts group image file format
LOO	Leave-one-out classifier evaluation method
MB	Megabyte, approximately 1,000,000 bytes
MMD	Marine mammal detector; formal name of the software developed
MSE	Mean squared error
MS	Mean squared
NN	Neural networks
OFAT	One factor at a time experiment

OS	Operating system
PRESS	Predication error sum of squares
QD	Quadratic discriminate
QP	Quadratic programming
RBF	Radial basis function
RGB	Red, green, and blue bands in a colored image
ROI	Region of interest
RSM	Response surface methods/methodology
SS	Sum of squares
SRS	Software requirements specification
SVM	Support vector machine
TIFF	Tagged image file format
UML	Unified modeling language
dpi	Dots per inch
x	Scalar value
$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
$\mathbf{x}^T$	Transpose
$\mathbf{X}^{-1}$	Matrix inverse
$\mathbf{\Sigma}$	Covariance matrix
σ^2	Variance
s^2	Sample variance
s	Standard deviation
μ	Mean
$\bar{y}$	Mean response, or sample mean
y_i	Measured response i
$\hat{y}_1$	Predicated response i
F_0	F-value test statistic for null hypothesis
$P(x)$	Probability mass function
$p(x)$	Probability density function
$P(x\|y)$	Conditional probability of x given y
R^2	R-squared, or goodness-of-fit for regression the model

R^2_{Adj}	R^2 adjusted to reflect the number of factors in the model
R^2_{Pred}	R^2 that measures the predictive capability of the model
α	Significant level used in hypothesis testing
$N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
C	SVM regularization parameter
γ	RBF kernel parameter
b	Constant offset or threshold in a SVM
α_i	Lagrange multiplier
p_m	GA probability of mutation; mutation rate
p_c	GA probability of crossover; crossover rate
n_p	GA population size
n_g	Number of GA generations
l_c	GA chromosome length
c	Chromosome string
$f(c)$	GA fitness function for c
f	Chromosome fitness
d	Class dimensionality, or number of features
m	Number of features selected in FSS from the original set of d features
$\mathscr{C}_i$	Class i
$\mathcal{F}$	Feature space
h_i	Classification (hit) rate for $\mathscr{C}_i$
$\forall$	For-all elements in a set
$\mathbb{A}$	A set
$\mathbb{R}$	The set of real numbers
$\mathbb{Z}$	The set of integer numbers
$\triangleq$	Is defined as
$\approx$	Approximately equal
$\|\cdot\|$	2-norm (Euclidean distance), $\|\mathbf{x}\| \triangleq \sqrt{\mathbf{x} \cdot \mathbf{x}}$
v_{min}	Watershed minimum variation
$O(n)$	Big-oh complexity
ϕ	Moment invariant

Chapter 1

Introduction

1.1 Problem Statement

An important aspect of monitoring the population of many marine mammal species (i.e. doing population assessments) is the ability to count a sample of the population using aerial photographs. Presently, counting is done by manually interpreting aerial photographs or negatives gathered from aerial surveys, recording the interpretations, and transferring data from paper documents to computer systems. This is a very time consuming, monotonous, error-prone, and expensive process.

For example, the predominant organization in Canada that conduct population assessments is the Department of Fisheries and Oceans (DFO)[1]. According to DFO [1], manually counting animals as part of the assessment is one of the most complex and time consuming tasks undertaken by the department. Firstly, conducting the aerial survey to acquire the photos is expensive. Secondly, after the survey is flown the eight to ten thousand photos are printed (unless analysis is done directly from the negatives). Developing nine inch film negatives is very expensive. Thirdly, the photos must be manually analyzed by highly trained

[1] Interest in conducting private surveys has risen as companies become environmentally conscious and/or wish to legally protect themselves from alleged negative environmental impacts of their operations.

readers;[2] a process that is extremely difficult and time consuming. Complete analysis of thousands of photos can take up to one year for several scientific staff who are exclusively dedicated to the task. Fourthly, current methods require readers to manually mark each animal identified on some sort of recording paper and compare the results with other readers for quality control. Data collected is then manually entered into a computer system for further analysis.

The complexity and excessive time requirements of counting have a number of consequences: peer review and release of the assessments often do not occur until the following year; errors can be made at any step in this process; and personal health problems, such as back injuries and eye strain, have been reported. Automatic or semi-automatic analysis of aerial photographs will significantly reduce the time necessary for analyzing surveys, improve the efficiency of readers, and potentially increase the accuracy of the estimates.

In eastern and northern Canada, one species that is monitored on a regular basis is the beluga whale. Belugas are a good study species for automatic counting because they are relatively large, white targets in high contrast with their background. Once a system is developed to detect and classify belugas, it can be modified to do the same for other marine mammals.

1.2 Objective

The object of this work is to develop a software system that will assist scientific staff in the analysis of aerial photographic images to detect marine mammals. To achieve this, investigation and development of image segmentation and pattern classification algorithms for the detection and classification of belugas will be completed. This will be integrated into a software application with a GUI (henceforth just called the GUI) for ease of use.

[2] A reader is an experienced scientific person who manually examines photographs or negatives for the purpose of counting the target species.

Additionally, the GUI will have features to aid in the data collection, storage, and analysis process. This software system will automate much of the manual practices currently used, thus allowing DFO scientific staff to assess marine mammal populations in a more timely and cost effective manner than current techniques. Finally, the software will be developed with the intention of adding modules for detecting other marine mammals—such as seals, narwhals, sea otters, and penguins—without changing the core application.

1.3 Document Organization

This document is organized as follows. Chapter 2 describes the relevant background material. Included is a description of the problem, motivations for a solution, and a summary of previous works. Chapter 3 describes the approach used in this work to the solve problem, justification for the techniques chosen, along with an overview of theory when necessary. Chapters 4 to 7 describe the solution to the problem, including methodology and results. Chapter 4 describes the image segmentation techniques. Chapter 5 describes the application of DOE to calibrate the GAs used in Chapter 5 for classifier optimization. Chapter 6 describes the development of beluga whale classification techniques. Several algorithms are presented and results of each compared. The GAs calibrated in Chapter 5 are presented as a solution to classifier optimization and the results are discussed. Chapter 6 describes the developed software application that is crucial to a complete population assessment solution. In the final chapter, results are summarized, conclusions are drawn, and recommendations for future work are presented.

Chapter 2

Background

This chapter presents the relevant background to and motivations for this current work. First the background of the problem is described. This is followed by a summary of previous works to solve similar problems.

2.1 Beluga Whales

The scientific classification of beluga is as follows: order *Cetacea* (whales), suborder *Odontoceti* (toothed whales), family *Monodontidae* (the only other family member is the narwhal), genus *Delphinapterus* ("dolphin without a fin"), and species *leucas*. The beluga is commonly named the "white whale" [2, 3] because adults are varying shades of white and young are typically a light brownish gray [3, 4]. Adult belugas range in length from 3m to 5m, and weight up to 1500Kg.

Belugas inhabit cold arctic and subarctic oceans, including Baffin Bay and Hudson Bay, but occur as far south as the Gulf of St. Lawrence. They are also found in large rivers and estuaries, such as the St. Lawrence river [3]. Belugas can be found in deep waters and shallow coastal waters, depending on season, prey distribution, and ice flows. Some populations migrate to deeper, cooler waters in winter and shallow warmer waters in summer,

especially while breeding [2, 3, 4]. Belugas are social animals, and as a result, they often occur in groups of 2 to 25 individuals, called pods [5].

The estimated world wide population of belugas is reported to be from 88,000 [6] to 200,000 [3]. The population is effected by native subsistence hunting, pollution, ship traffic, climate change, and other natural causes such as food supply and predation [7, 8]. Commercial hunting in the 19th and early 20th century have had dramatic impacts on todays population. Although not considered an endangered species world wide, several individual stocks are of concern, such as the belugas of Alaska's Cook Inlet, St. Lawrence estuary, and Ungava Bay [3, 4, 7]. Indeed, some worry the beluga could disappear in said areas if current harvesting levels continue. Such concerns have lead to a call for regular population assessment programs [2, 8]. A key component of such programs are aerial transect surveys and photograph analysis of the type that will be automated in this work.

2.2 Marine Mammal Population Assessments

Monitoring marine mammal populations is a necessity. In addition to dangerously low beluga whale populations in certain areas (Section 2.1), other marine mammals in Canadian waters are at risk, such as the Eastern Bowhead and North Atlantic Blue and Right whales. Many marine mammals are hunted on a commercial and subsistence basis, including several north Atlantic seal herds. To maintain healthy population levels, quotas must be issued based on accurate population estimates. To address this issue, calls have been made to perform regular surveys to supplement and reduce the need to rely on less accurate techniques [8].

There are several methodologies for gathering data for population assessments, each with advantages and disadvantages. The four most common methods [9] are land sightings, boat sightings, acoustic surveys, and aerial surveys. Monitoring from land sites involves

using binoculars to observe mammals from several designated land positions over a pre-defined time period. This method has many limitations, including low coverage due to a small viewing range. Thus, this method is restricted to observing near shore mammals. Additionally, staff must be trained to identify mammals by shape of blow, head, or back.

Monitoring from boats involves observing mammals from survey vessels. Generally, a line transect technique (LTT) and a strip census methodology (SCM) is applied. Both methods have limitations and problems, including observer fatigue, differential attraction or avoidance of mammals to the vessel, difficulty in counting large groups of animals, and observers missing animals in shadows or in depressions behind outcrops.

The acoustic survey is a relatively new method and much work remains to be done to assess it utility. Put simply, this method involves identifying mammals by their unique underwater sounds.

The final method is the aerial survey. This method involves the use of fixed-wing aircrafts or helicopters. Two variations exist, although both are often combined to obtain the benefits of each. The first, called the visual survey, involves two or more observers in an aircraft counting mammals as they are seen. The second, called the photograph survey, uses one or more cameras attached to the bottom of an aircraft to take photographs at a controlled rate.

There are many advantages in using aerial surveys: it can be very efficient for mammals along coastlines, allows access to remote areas not easily accessible by other means, such as large ice flows, and allows more coverage in less time. The photographic approach has the additional advantage of allowing personnel to scan for animals in the photographs in a laboratory setting after the survey is complete. It has been shown that this method correlates with seaborne surveys [10].

Given the benefits of the latter approach it has been accepted as the primary surveying method in recent years, and one can only conclude that it will be used more frequently in

the future. Systematic strip transect aerial photograph surveys of the St. Lawrence beluga has been the standard approach for estimating populations since 1988 [11]. It is also being used routinely for conducting estimates of seal herds in Atlantic Canada (e.g. [12, 13]). Indeed, photograph and visual surveys are considered the most appropriate methods of estimating seal pup production [14]. It has also been stated that systematic and regular use will alleviate past problems of incomparable and inconsistent results of applying different, non-standard techniques [11, 12, 13, 14]. However, conducting aerial surveys is very time consuming and expensive. Any means of automation will go a long way in reducing cost and time, and will ultimately lead to more accurate and consistent estimates.

2.3 Aerial Surveys and Manual Photograph Analysis

This section describes aerial photograph surveys in general, and beluga whale photograph analysis in particular. The discussion is drawn from seal and beluga surveys documented in [11, 12, 13, 14, 15], correspondence with practitioners, and personal experience with photograph analysis.

Aerial photographs are obtained by flying a survey over an area of interest at key times of the year and taking high-altitude, high-resolution color (black and white for some species, such as seals) photographs of the area. A fixed-wing aircraft equipped with a large format (e.g. 9"×9") mapping film camera is used to take photographs at regular intervals. Aircraft speed and altitude is typically constant. The altitude is chosen to maximize the area covered by the photograph while maintaining the resolution necessary to manually locate the species in the photograph with the aid of a hand held magnifying glass. The flight pattern is defined by systematic strip transect lines. These are a set of parallel, (typically) equally spaced lines of varying lengths. The design of the strip transect takes into consideration survey area, altitude, camera field of view, required coverage area, and whether or not photograph

overlap is desired.

A survey is designed such that the maximum number of animals are observable while minimizing sources of error. Parameters almost always include the area of interest, time of year, time of day, weather conditions, and time required to complete the survey. However, the details are based largely on the target species habitat and behavior. For belugas it is important to minimize sun glare and wave crests. Hence, the sun angle and wind speed are important. Reconnaissance surveys are conducted to identify the best area to formally survey and provide complementary visual counts. This is particularly important for certain species, such as seals that travel with ice flows. To minimize double counting caused by movement of animals while conducting the survey, as much of the survey area as possible is covered in the shortest time possible. A survey is often flown over the same area as a previous survey so comparisons can be made between data sets while eliminating the need to identify a new survey area and define new transect lines.

The analysis of photographs for the purpose of counting marine mammals is called reading the photograph. It is performed by readers; these are trained scientific personnel with extensive knowledge of the target species and prior experience reading said species. The reader is tasked with viewing each photograph and recording the number and types of animals found (adult and young for beluga surveys) and the percent of the image that is unreadable. Unreadable parts of an image consists of areas the target animal cannot occur in or cannot be identified in because of the masking effect of noise. Examples include land, sun glare, and areas of extensive waves crests.

Each photograph is analyzed in a systematic manner to ensure a complete reading is performed. A transparent acetate with a $n \times n$ (typically $n = 10$) grid is placed over the photograph to guide the reader's search and act as a reference system when describing animal locations within the photograph. Annotations are made on the grid (or a sheet of paper with an identical grid) to mark the location of animals, other significant objects

(e.g. boats, land), and to insert comments.

For beluga whale surveys, the 9"×9" negatives are analyzed directly rather than analyzing the developed photographs. Developing photographs from negatives is costly and does not improve the inspection process nor the analysis results. The negatives are analyzed with the aid of a specially designed light table and a low-magnification (e.g. 10×) magnifying instrument, such as a dissecting microscope or hand held magnifying glass. A roll of negatives is attached to one end of table and each negative is scrolled across the table, which is illuminated from beneath, and analyzed in sequence.

Manual identification of belugas is not always conclusive, even for the most experienced reader. Problems include partial occlusion of one whale by another or by a wave crest, very deep whales which have a low contrast with the background, whales resembling wave crests, masking of whales by water disturbances created by the whales themselves or by natural phenomena, and whales located in areas of intense noise caused by wave crests, sun glare, sun speckles, or wave fronts. As a result, readings consist of labeling whales as certain (high confidence the target is a beluga) or uncertain (low confidence the target is a beluga).

Once all the negatives (or photos) have been analyzed, measures are taken to ensure the most accurate final reading. Such measures usually involves a reader re-reading a subset of images they have already read, reading a sample of the images read by another reader, and comparing sample readings to readings performed by more experienced readers. From these results correction factors are calculated for each reader and applied to all readings for that reader. Additionally, all photos with uncertain observations are often checked by more experienced readers who may edit the initial analysis by confirming or rejecting detections as necessary and adjusting the counts accordingly. More likely, all readers discuss the questionable targets until agreed upon classifications are made.

The entire reading procedure, including quality control, can take many months for sev-

eral people. It is the intention of this work to automate or provide software support for the above described activities, thus reducing time and sources of error.

2.4 Automated Mammal Detection: Previous Efforts

As far as can be determined, no software system has been developed to encapsulate the process and activities described in Section 2.3. Additionally, no research has been conducted on detecting and classifying said species from digital images. In this way, this work is completely original.

Commercial and public domain image analysis software are available (e.g. *MicroGOP Software* [16], *ImageTool* [17], *Gimp* [18], and *Adobe Photoshop* [19]). However, these are specific to an unrelated problem, or are designed for general image manipulation and basic image processing. The former does not provide the features necessary to build the required system; the latter is too generic and requires a great deal of image analysis knowledge to be used effectively to solve complex problems [1]. Additionally, such tools do not provide the sophisticated features necessary to automate the process required to detect, classify, and count whales. Some software, such as image processing programming libraries (e.g. *Matrox Imaging Library* [20]), provide basic building blocks for software development and exploration of possible solutions to image processing problems. They are not designed to be used out-of-the-box to solve specific, application domain problems. Additionally, none of these tools provide solutions to the problem of automating the data collection, data presentation, and manual processes associated with photograph analysis as detailed in Section 2.3.

Limited published work has been completed in automating the counting of mammals using aerial photographs, mainly because automated counting and image processing techniques are not widely used to aid population assessments [21]. Gilmer *et al.* [22] developed a method to aid in counting geese from aerial photographs. Randomly selected areas from

each of the digitized images were chosen to extract statistics to manually determine a single gray-level, image specific threshold and pixels-per-goose relationships. When the threshold was applied, each image was segmented into two classes of pixels, goose and non-goose. The number of goose pixels were counted, and the goose-per-pixel relationship was used to determined the total number of geese. The results showed accurate counts and a significant improvement in time to analyze each image compared to manual methods, even when considering the time required to derive the image-specific thresholds. However, the method is relatively manual and solved the simplest case only: white geese on a dark background.

Building on Gilmer's work and the authors own previous work [23], Bajzak [24] developed a more refined computer aided technique for counting snow geese in aerial photographs. His technique used two computer programs. The first prints the density values from a selected subimage. These are used to manually determine tonal range of snow geese and the minimum and maximum number of pixels that represent birds. After conducting experiments to fine tune these parameters, the second program uses the parameters as criteria to identify individual birds. His investigation found that identifying individual birds as clusters of pixels provided more unbiased and more accurate counts than Gilmer's techniques. He also found using image tone alone is best suited for uniformly colored species. Compared to traditional manual techniques, the time required to analyze an image was reduced even though a lot of time is necessary to manually determine image specific parameters required for segmentation and classification.

Using a more interactive approach, Cunningham *et al.* [25] developed software to aid in counting geese. The software supports features to access and enhance images, mark animals that should be counted, and output collected data formatted in tables in text files. The basic approach is to manually or interactively mark the objects to be counted. In the manual method a paint brush tool is used to mark the objects. In the interactive method the user interactively selects thresholds to segment the image. The user then marks a sam-

ple of the segmented objects to define selection parameters, such as size, and a counting routine counts objects with similar features. The technique showed good results, but is semi-automated.

Laliberte and Ripple [21] investigated methods to automate counting of wildlife from aerial photographs and made comparisons with Gilmer and Cunningham's work. Laliberte and Ripple's goals were to developed general image processing techniques simple enough to require only basic image processing knowledge and use of public-domain software, and to test the feasibility of using IKONOS high-resolution satellite imagery. Using *ERDAS IMAGINE* [26] and *ImageTool* [17] a sequence of steps was developed to allow a user to effectively apply the features of these tools to segment and count snow geese, Canada geese, and caribou in scanned aerial photos. The details differ, but in general a smoothing filter is applied, followed by a visual inspection of subimage histograms to pick a threshold to segment objects, whose areas are used to calculate the total number of animals. This interactive methods are less labor-intensive than manual techniques and showed good results that were highly correlated with manual counts. The experiments with 1m resolution IKONOS satellite imagery resulted in very small targets, leading to difficulties in segmentation and discrimination.

In a completely different avenue of research, Trathan [27] developed techniques to count macaroni penguins from digitally scanned color aerial photographs. The results were compared with manual photograph counts and was found to be highly correlated. Although his methods lead to good results, it is not a completely automated solution. Rather, image processing consists of a set of manual steps aided by MATLAB's [28] image processing routines. In essence, the steps are: (1) define a ROI around penguin colonies, (2) from a representative sample, calculate the foreground (penguins) and background descriptive statistics, (3) threshold the image at T calculated from the statistics, (4) apply a blob filter, and (5) count the segmented blobs. The main disadvantage of this approach is that

a lot of time is spent performing manual analysis, including defining ROIs, inspection of histograms, determining a threshold, and saving results of intermediate processing steps.

Gosine *et al.* [29] conducted a feasibility study in automatic counting of sea lions from aerial images, video, or still pictures. The developed method involved thresholding an enhanced edge image to produce a binary image of sea lions and not-sea lions. The user then manually discriminates between sea lion objects and not-sea lion objects for the first image frame in the video. Basic object features (e.g. size, shape, mean, standard deviation) and a specialized intensity gradient across the object are calculated and recorded. A database of such information is used to build a nearest-neighbour classifier [30] that is applied to subsequent images in the video scene to discriminate sea lion objects from not-sea lions. The results indicate good agreement between manual counts and automated counts. Unlike previous works summarized above, the counting algorithm was almost completely automated. However, a software system was not developed to encapsulate the algorithms for ease of use, to aid the user in the required manual steps, and to present a report of the results.

It is uncertain why more techniques and software tools have not been developed for automatic counting of mammals, in particular marine mammals. It could be argued that historically computer processing capabilities, image processing, and pattern recognition techniques were inadequate to deal with such complex problems; this is certainly no longer the case. Computer processing power has advanced dramatically over the last 20 years allowing high resolution images to be processed in a fraction of the time it once did. Additionally, image processing and pattern recognition techniques are being applied to many similar challenging problems. In forestry, techniques have been developed to segment and classify tree crowns [31, 32, 33]. In the medical field, there are numerous examples, including counting cells, classifying muscles fibers [34], and analysis of pores in soil aggregates [35].

Chapter 3

General Approach and Justification

This chapter describes the proposed approach. It gives an overview of the techniques applied in the solution and justification for using these techniques based on a literature review. Theory is kept to a minimum and presented only when necessary.

3.1 The Approach

The basic approach can be summarized as a classical pattern recognition system consisting of five steps: (1) sensing, (2) segmentation, (3) feature extraction, (4) classification, and (5) post-processing [30]. The first step provides input for the system, such as the digitized images from a camera. The segmentation step is concerned with extracting objects (targets) to be classified from the signals. The next step is feature extraction, where properties (the features) describing the segmented objects are calculated. For each object, a feature vector $\mathbf{x}$ of length d, where d is the number of features, is created. Next follows the classification step, where $\mathbf{x}$ is assigned to one of the *a priori* defined classes (groups) of objects. Finally, in the post-processing step the results of classification are analyzed and some action performed. In this work, all steps must be addressed. However, the bulk of the work is contained in segmentation (Chapter 4) and classification (Chapter 5 and Chapter 6). Hence,

the remainder of this chapter will focus on these steps.

In this work pattern recognition is classification: assign x to one class, $\mathscr{C}_i$, from the set of predefined classes $\{\mathscr{C}_1, \mathscr{C}_2 \cdots \mathscr{C}_n\}$, where n is the number of classes. There are several approaches to pattern recognition [36], including syntactical or structured [37], knowledge-based [38] and statistical [30, 39, 40]. However, since one of the most effective methods to build a classifier is to learn patterns from examples [30] (known as training the classifier), this work takes the last approach. Because we know *a priori* the distinct classes, a supervised learning approach is taken.

There are many statistical, supervised, classifiers available. Because there is no universal best classifier for all problems [30], selection of one over the other and configuration of these for a given problem is not trivial. To address this issue, two classifier types are chosen and different configurations of these are compared. The first is the classical, often used, Bayesian approach, known as the minimum error-rate quadratic discriminate (QD). The second is a relatively new classifier known as the support vector machine (SVM).

A classifier normally requires optimization in order to achieve the best performance possible. One form of optimization consists of selecting the best classifier parameter values. Another more commonly used form, known as feature subset selection (FSS), consists of choosing a subset of the initial set of features used to describe the objects. In this work, a GA is used to perform both forms simultaneously for SVMs and FSS for QDs. The results of this optimization process is a set of values upon which the classifiers can be compared; for example, accuracy, false positive rate, and number of features. A caveat in using GAs is that values for the fundamental GA control parameters must be chosen, but there is no systematic way to select these values. In this work this problem is solved using design of experiments (DOE) to develop a model that is used to calculate the optimal values (within the limits of the design space) . This process is known as calibrating the GA.

Finally, the developed image processing and pattern recognition algorithms are inte-

grated into a software system that scientific personnel can use to aid in population assessments. The software GUI presents an intuitive interface to the algorithms, allows the user to load and manipulate images, allows initiation of automated detection and classification, and allows interaction with the results.

3.2 Bayesian Classification

The foundation of statistical pattern recognition is Bayesian decision theory. It was first proposed by Thomas Bayes [41] over a century ago, then generalized by Laplace [42], and first used for classification by Chow [43]. What follows is an overview on how it is used for classification in this work. For details see Section E.1 and [30, 39, 40, 44].

Bayesian classification is based on Bayes formula

$$P(\mathscr{C}_i|\mathbf{x}) = \frac{p(\mathbf{x}|\mathscr{C}_i)P(\mathscr{C}_i)}{p(\mathbf{x})} \tag{3.1}$$

where $P(\mathscr{C}_i)$ is the prior (*a priori*) probability, $P(\mathscr{C}_i|\mathbf{x})$ is the posterior (*a posteriori*) probability, $p(\mathbf{x}|\mathscr{C}_i)$[1] is the likelihood of $\mathscr{C}_i$ with respect to $\mathbf{x}$, and $p(\mathbf{x})$ is the evidence. $p(\mathbf{x})$ acts as a scale factor that ensures the posterior probabilities sum to one. For classification, this term is usually dropped. The basic approach is to minimize the probability of error by choosing the class that maximizes the posterior probability $P(\mathscr{C}_i|\mathbf{x})$. This leads to Bayes decision rule

$$\text{if } P(\mathscr{C}_i|\mathbf{x}) > P(\mathscr{C}_j|\mathbf{x}) \; \forall (j \neq i) \text{ classify as } \mathscr{C}_i \tag{3.2}$$

It can be shown that Bayes decision rule provides the optimal classifier performance (minimum error rate) [30]. A Bayesian minimum error rate classifier can be formulated using

[1]Following the convention of [30], an uppercase $P(\cdot)$ is used to denote a probability mass function and a lowercase $p(\cdot)$ is used to denote a probability density function.

the discriminate function

$$g_i(\mathbf{x}) = P(\mathscr{C}_i|\mathbf{x}) = p(\mathbf{x}|\mathscr{C}_i)P(\mathscr{C}_i) \tag{3.3}$$

If f is a monotonically increasing function, $g_i(\mathbf{x})$ can be replaced with $f(g_i(\mathbf{x}))$ without changing the classification results. Hence, the discriminate can be written as

$$g_i(\mathbf{x}) = \ln p(\mathbf{x}|\mathscr{C}_i) + \ln P(\mathscr{C}_i) \tag{3.4}$$

To define the structure of the classifier for a given problem $p(\mathbf{x}|\mathscr{C}_i)$ and $P(\mathscr{C}_i)$ must be determined. $P(\mathscr{C}_i)$ is usually chosen manually based on prior knowledge of the problem. Determining the conditional density $p(\mathbf{x}|\mathscr{C}_i)$ is much more difficult, indeed it is the biggest drawback of the Bayesian method. There are two common methods: maximum-likelihood and Bayesian estimation. [30] makes several statements that favour the former method, even though the latter may be favored theoretically. First, the results of maximum-likelihood are frequently nearly identical to Bayesian methods. Second, it has good convergence properties as the number of samples increases. Third, the technique is easier to understand, implement, and is computationally less complex. Finally, the results are easier to interpret and understand. The maximum-likelihood method assumes the parameters are fixed and the best estimate of them is one that maximizes the probability of obtaining the training samples [30].

Following the maximum-likelihood approach, $p(\mathbf{x}|\mathscr{C}_i)$ is modeled as a Gaussian distribution, and for good reason. First, as [30] points out, the Central Limit Theorem states that the aggregate effect of the sum of a large number of small independent random disturbances will lead to a Gaussian distribution. This describes most patterns: a class is the prototype pattern, and objects classified into that class are randomly corrupted versions of the prototype. A second, and perhaps a more pragmatic reason for choosing the Gaussian, is that it

is analytically tractable [30]. Using the Gaussian, for $\mathscr{C}_i$,

$$p(\mathbf{x}|\mathscr{C}_i) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right] \tag{3.5}$$

where $\mathbf{x}$ is the d dimensional feature vector of an object, $\boldsymbol{\mu}_i$ is the d dimensional mean vector of $\mathscr{C}_i$, and $\boldsymbol{\Sigma}_i$ is the $d \times d$ covariance matrix of $\mathscr{C}_i$. Because the normal distribution of $\mathscr{C}_i$ is completely specified by $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$, we say $p(\mathbf{x}) \approx N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. The final Bayesian minimum error rate quadratic discriminate function for $\mathscr{C}_i$ can be derived by substituting Equation 3.5 into Equation 3.4, resulting in the following

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(\mathscr{C}_i) \tag{3.6}$$

In the above equation, $(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) = r^2$ is known as the squared Mahalanobis distance from $\mathbf{x}$ to $\boldsymbol{\mu}$. This equation describes the distribution of samples about the mean $\boldsymbol{\mu}$. Positions of constant distance from $\boldsymbol{\mu}$ define hyperellipsoids of constant density and the volume of these hyperellipsoids measures the sample scatter [30]. The use of $\boldsymbol{\Sigma}$ makes the Mahalanobis distance metric invariant to scaling between features and corrects for correlation between different features. These important properties makes this metric superior to other distance measures such as the Euclidean and Manhattan distance. It also provides hyperquadratic (curves of various shapes and linear) decision boundaries.

3.3 Support Vector Machines

Support vector machines (SVM) are a new, elegant, and powerful class of supervised learning algorithms particularly well suited for classification, regression, and novelty detection. SVMs are based on statistical learning theory and some simple ideas about what learning from examples is fundamentally about [45]. From a theoretical point of view, they are

simple enough to be analyzed mathematically using computational learning theory for both simplistic theoretical problems and complex real world problems [45]. This stands in contrast to some other methods, such as neural networks, that when used on complex problems are very difficult to analyze. SVMs can achieve this because one can reason about them as a simple linear classifier in high dimensional feature space, even though nonlinear problems are being solved. The key in simplifying the SVM is that the complex model computations are not done in this high dimensional space, but rather in the input space through the use of kernels [45]. In this regard, SVMs are a linear machine that can function on non-linear data. Given a data set consisting of two classes, the SVM constructs a hyperplane decision surface between the two classes such that the margins between the plane and the two classes are maximized.

SVMs were first introduced as a binary, nonlinear classification technique in 1992 by Boser, Guyon and Vapnik [46]. This was based on earlier work by Vapnik and his colleagues where they developed theories to explain learning from a statistical point of view (e.g. [47, 48]). However, the driving force for the development of SVMs was the problem of finding the appropriate balance between accuracy attained on a training set and the capacity of the machine in order to achieve the best generalization performance [49]. This is often discussed in the context of over fitting, capacity control, and bias variance trade off [49]. In 1995 Cortes and Vapnik [50, 51] extended SVMs for non-separable classification problems, at which point SVMs started to gain momentum. Because of its beginnings with Vapnik's works and his many publications on the subject, Vapnik may be considered the father of SVMs. Since then a plethora of research has been conducted and published. Some significant developments include extensions for novelty detection and multi-class classification. Work continues to address various SVMs weaknesses, some of which are mentioned later in this section.

Over the last 10 years SVMs have been successfully applied to varying problems, pre-

dominantly in pattern recognition. The literature invariable reports SVM classification results that are equal to or better than the results of other classifiers. Describing or even enumerating the voluminous work done to date is beyond the scope of this thesis. For a complete survey of applications of SVMs to pattern recognition see [52].

3.3.1 The General SVM

Let $\mathbb{T} = \{\mathbf{x}_i, y_i\}$, $i = 1 \ldots n$, be a set of training data, where $\mathbf{x}_i \in \mathbb{R}^d$ is an input sample vector belonging to a class $y_i \in \{+1, -1\}$, d is the dimensionality of the sample space (the length of the vectors), and n is the number of samples in the data set. The discriminate function for linear, separable patterns is

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} y_i \alpha_i \mathbf{x} \cdot \mathbf{x}_i + b\right) \tag{3.7}$$

The Lagrange multipliers α_i are obtained by solving the following dual quadratic programming (QP) optimization problem

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \; i = 1 \ldots n \\ & C \geq 0 \end{aligned} \tag{3.8}$$

The coefficients α_i define a maximum separating hyperplane in a high-dimensional feature space [53]. These are constrained by the upper bound C, called the regularization parameter, which governs the number of errors the classifier will tolerate during training—the higher the value of C, the higher the cost associated with training error and the more

Table 3.1: Common SVM kernels. In all cases, the kernel specific variables (γ, d, b, c) are chosen a priori by the user. For the neural network kernel, only certain values of b and c satisfy Mercer's theorem. Note for the RBF $\gamma = 1/2\sigma^2$.

Name	Kernel $K(\mathbf{x}, \mathbf{x}_i)$
Linear	$\mathbf{x} \cdot \mathbf{x}_i$
Polynomial	$(\gamma \mathbf{x} \cdot \mathbf{x}_i + 1)^d$
Radial basis function (RBF)	$e^{-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2}$
Neural network (sigmoidal)	$\tanh(b(\mathbf{x} \cdot \mathbf{x}_i) - c)$

complex the classifier. This parameter must be chosen *a priori* by the user.

Non-linear, non-separable patterns are made linearly separable in high-dimensional feature space $\mathcal{F}$ using a non-linear mapping function $\Phi : \mathbb{R}^d \rightarrow \mathcal{F}$. In $\mathcal{F}$, the training algorithm only depends on the data through the dot product $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Mercer's theorem (1908) states that for certain mappings Φ and any two points $\mathbf{x}_i$ and $\mathbf{x}_j$ the inner product of the mapped points can be evaluated using a kernel function K without explicitly knowing the mapping [54]. Let $K(\mathbf{x}_i, \mathbf{x}_j) \triangleq \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, then in Equations 3.7 and 3.8 $\mathbf{x} \cdot \mathbf{x}_i$ and $\mathbf{x}_i \cdot \mathbf{x}_j$ are replaced with K giving the following discriminate function for non-linear, non-separable patterns:

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b\right) \tag{3.9}$$

Using Mercer's theorem it can be shown that a given K is suitable for a SVM, but K and a suitable mapping Φ cannot be derived. Hence it falls to the SVM developer to derive an experimental K and use Mercer's theorem to evaluate whether or not it can be used in a SVM. This is not an easy task as revealed by the relatively small number of kernels developed. The most common kernels cited in SVM literature are shown in Table 3.1.

3.3.2 Multi-class SVMs

Theoretically, SVMs are two class classification machines. For many pattern recognition problems this is not sufficient. As a result, methods have been developed to extend two

class SVMs to handle multi-class pattern recognition problems.

All accepted methods for extending binary SVMs fall into one of two groups. The first, referred to as binary-based methods, perform k-class classification by combining the results of two or more binary classifiers. Here the problem is broken into several binary subproblems and outputs are combined to make a multi-class decision. The most common methods in this group are one-against-all [55]), one-against-one [56, 57] and directed acyclic graph (DAG) SVM [58] The second group, referred to as the all-at-once approach, do k-class classification all at once. In this approach the SVM optimization problem is reformulated to account for all classes. Examples in this group include method originally proposed by Vapnik and later investigated by Weston and Watkins [59, 60, 61], and the Crammer and Singer (CS) method [62].

The binary-based methods are the most commonly used in the literature surveyed, in particular one-against-one and one-against-all. This is not surprising given that these are easy to understand and implement. In terms of implementation, they have the advantage of reusing the existing binary SVM implementations to the maximum extent, where as all-at-once methods require a reformulation of the SVM. For problems with large data sets, this reformulation generally leads to an explosion in the number of variables in the quadratic programming problem, making them very complex and computationally expensive to solve. There are some decomposition techniques that can be used to minimize this problem, and some have proposed SVMs that take advantage of this (e.g. [63, 64]), but such SVMs are not widely used.

Two independent studies have suggested that the binary based methods, in particular the one-against-one method, is the best overall, especially when accuracies are important. The first was conducted by [60, 61], who compared one-against-one, one-against-all, and two k-class all-at-once methods on six benchmarks and concluded that in terms of accuracies there is no significant difference between any of them. On the other hand, the all-at-once

methods used the fewest support vectors, leading to a slightly faster classification time.

Arguably a more thorough comparative study was conducted by [63], who compared five methods on ten benchmarks. Compared to the aforementioned studies, larger data sets were used and a decomposition method was established to reduce the number of quadratic programming variables in the all-at-once methods. In terms of accuracy, they are all statistically very similar, but one-against-one slightly outperformed the others. In terms of training time, the one-against-one approach outperforms the other methods in nearly all tests; only the DAG method had similar results. The above results were obtained using the RBF kernel. When using the linear kernel, the one-against-one method equaled or outperformed all other methods in all but one out of six experiments.

3.3.3 Configuring SVMs

A big question in developing SVMs is how to chose the kernel, a value for the regularization parameter C, and values for the kernel parameters. Choosing a kernel is a manual choice made by the developer. Typically, a kernel that best models the problem is chosen, but this can be difficult if the form of the problem is unknown. The RBF kernel maybe the best choice for a number of reasons [65]. First, it is a non-linear mapping, with the added benefit that the linear kernel is a special case of the RBF kernel [66]. Second, the sigmoid kernel may lead to a kernel matrix that is not positive definite, and theoretical and experimental studies have shown that the sigmoid rarely outperforms the RBF [67]. Finally, the polynomial kernel has more hyperparameters to choose and has more numerical difficulties (if a high degree is used) than the RBF [68].

When using the RBF kernel, the kernel parameter γ (kernel width) must be chosen by the user. γ determines the area of influence a support vector at the center of the RBF has over the data space—as the value of γ increases, the area of influence weakens. Hence, choosing a value too low results in underfitting, and choosing a value to high results in

overfitting. Like C, γ can dramatically effect the complexity of the machine and its classification performance.

The choice of values for C and the kernel parameters (collectively known as the SVM parameters) are critical to SVM performance, but is often naively neglected [69] and set to default values such as $C = 1$ and $\gamma = 1/d$, where d is the number of features. In the absence of *a priori* knowledge of the form of the problem, an often cited approach for SVM parameter (C and the kernel parameters) selection is to conduct experiments: train, test, and compare SVM with different parameter values. One heuristic search technique used when the SVM is configured with the RBF kernel is the SVM parameter grid search [65, 67]. Recently, studies have suggested GAs for SVM parameter selection, but limited work has actually been conducted. A summary of previous efforts using this approach is given in Section 3.7.

3.4 Classifier Optimization

In the context of this work, classifier optimization refers to maximizing the accuracy of a given classifier. This can be accomplished in several ways, including choosing the best classifier parameters, training on a good sample set, selecting and extracting features that describe each class as best as possible, and selecting the best combination of those features. There are constraints placed on the developer as too which of these optimization possibilities can be used. For example, it is often difficult or impossible to improve the sample set due to the difficulty or cost associated with gathering and analyzing data. As a result, optimization usually means feature subset selection (FSS) and, for some classifiers, parameter selection (Section 3.7).

FSS refers to the finding a subset of m features from an original set of d features ($m \leq d$) that gives the smallest classification error (or best accuracy). This is a difficult problem be-

cause the developer does not know how combinations of features interact to create decision boundaries, and if the dimensionality (d) is large, trying every possible combination is difficult or impossible. Nevertheless, FSS is important for a number of reasons:

- Feature vector size and the presence of irrelevant features can increase computation time and memory requirements.

- Typically there exists some subset of m features that increases the generalized performance of the classifier over the initial set of d features. This is often the case when there are irrelevant features, redundant (highly correlated), or biased features in the set.

- It reduces the curse of dimensionality [70, 71, 72]. For classes that are trained on unbalanced data, FSS is even more important because the negative effects of this curse are amplified for unbalanced classes [73].

FSS algorithms typically have three components [74]: (1) a search algorithm that searches the feature subset space of size 2^d, (2) an evaluation function that takes a feature subset as input and outputs a value indicating the "goodness" of the feature subset, and (3) a performance function that takes the best feature subset and uses it in a classifier to classify instances of the dataset.

There are two paradigms to solve this problem: the filter method and the wrapper method [75]. The filter method analyzes the feature vectors independently of the classification algorithm and removes features with undesired properties, such as redundancy and irrelevance. The classifier accuracy is not considered in this method; rather, the features are selected based on some heuristic goodness. In the wrapper method, the classification algorithm that will be used in the final system is or is a part of the evaluation function. Here, the estimated accuracy of the classifier when trained on a feature subset is used to evaluate the suitability of the subset.

It has been shown [75, 74] that the wrapper method can provide better classifier accuracy than the filter method. This is most likely a result of the wrapper taking into account the interaction between the classifier and the feature vectors it is trained on [76]. On the other hand, because at least one classifier—evaluation methods such as cross validation and hold out require many classifiers—is trained and tested for each feature subset they are generally computationally more expensive and thus slower to execute [75, 77, 74, 76]. They can also suffer from over fitting if a small training set is used [75].

There are several search strategies documented in the literature: exponential, randomized, and sequential [78, 74]. Exponential algorithms essentially search through the entire feature set space. These have the potential to achieve the best accuracies; however, they have a complexity of $O(2^d)$, so can only be used when the dimensionality is small. Examples include branch-and-bound and exhaustive algorithms. Randomized algorithms, whose complexity depends on the randomization algorithm and stopping criteria, randomly searches through the feature space. These have been reported to achieve very high accuracies [78, 79, 80, 81], but may not select the fewest number of features [74]. Examples include genetic and simulated annealing algorithms. Sequential algorithms sequentially add or remove features depending on the results of the evaluation function. These have a complexity of $O(d^2)$. Although they can compete with randomized methods, on some large features sets with complex interactions among features, their performance is considered brittle: sometimes good sometimes very poor [81]. Examples include sequential floating forward (SFFS) and backward sequential selection (BSS).

Research on feature selection has been ongoing for many years resulting in a multitude of literature on the subject. For tutorials see [82, 83]. For a taxonomy of feature selection techniques, see [84, 85]. For comparative studies see [86, 84, 82, 87]. In recent years the GA has been studied as a randomized feature space search algorithm, most generally using the wrapper method. Several works have shown that GAs are comparable, perhaps

even superior, to the best sequential methods. [88] is one of the earlier works showing the superiority of GAs compared to traditional methods. [89, 90, 91, 92] followed with further evidence. However, [86, 84, 93] all argue that GAs are comparable to sequential methods, but contradict each other on which is superior and under what conditions (such as size of d). More recently [87] conducted a thorough comparative study into the subject. Experiments were conducted using three sequential methods, a simple GA, and four variations of a hybrid-GA. The conclusion is that SFFS is the best of the sequential algorithms (collaborating with previous work cited above), the simple GA is always better than the SFFS algorithm regardless of problem size, and the hybrid-GA may perform better than the simple GA for large problems.

3.5 Classifier Evaluation

One of the most common methods of evaluating a classifier is estimating its error rate based on resampling. There are several common methods, including resubstitution, hold-out, k-fold cross validation, leave-one-out (a special case of k-fold cross validation), and bootstrapping. These have been studied for decades in the literature with no consensus as to which is superior in estimating the true error rate of a classifier [94]. Nevertheless, a few general remarks are merited. The resubstitution and hold-out methods can only be used with very large sample sets [94]. For smaller data sets, leave-one-out is arguably superior in most cases, but has a very high computational complexity that often precludes its use. k-fold cross validation and bootstrapping, which have much smaller complexities, have been documented as sensible alternatives [30, 40, 94], although one is not recommend above the other.

The literature on SVM evaluation seem to favor k-fold cross validation or hold-out. This is probably because SVMs having slow training times, especially for large C and large data

set sizes, hence only fast evaluation methods are suitable. Using k-fold cross validation with k chosen appropriately (most of the literature set $k = 10$), a good estimate of accuracy can be made in a reasonable amount of time. The estimated accuracy is good partly because all samples participate at least once in training and testing. As a result, training and testing covers the complete data set, but testing always occurs on unseen data, thus avoiding bias toward training samples that occur when testing on the same training data.

3.6 Genetic Algorithms

Genetic algorithms (GAs) are a class of search and optimization algorithms based on the principles of natural adaptive systems, most importantly the theory of natural selection and evolution. The GA attempts to achieve the robustness, adaption, and efficiency properties of an evolving biological species. The basic principles include a randomization of an initial population of individuals (encoded solutions, also called chromosomes), evolving new generations of individuals from older generations, survival of the fittest in each generation, creation of new individuals from selection of bits and pieces of the previous generations most fit individuals via crossover of chromosome pairs, and random mutation. GAs have been proven theoretically and empirically to provide robust search in complex spaces [95] and tend to converge to near globally optimal solutions [96].

Genetic algorithms arose from work in cellular automata by John Holland in 1975 [97]. Holland and his colleagues had two objectives [95]: (1) to abstract and explain evolutionary processes in natural system, and (2) to design artificial systems that mimic the natural system. Following this, most of the work up to the early 1980s focused on the theoretical aspects of GAs with few real world application [98]. During this time GA research carried out by De Jong [99] and Hollstein [100] stand out. Hollstein focused on the effect of different mating and selection strategies [98], and De Jong studied the properties and con-

figurations that constitute a robust search procedure [98]. Built on his own work and that conducted by these pioneers, David E. Golberg published a seminal book in 1989 entitled *Genetic Algorithms in Search, Optimization and Machine Learning* [95]. This is perhaps the most often cited book on GAs.

Since that time much research has been conducted on GAs, with many real world applications as a result. GAs have been used in such diverse fields as engineering, medicine, political science, social sciences, physical sciences, biological sciences, and business. GAs have been developed to primarily address optimization problems, such as FSS, but have been applied in scheduling, trend spotting, data fitting and clustering, and path finding [98]. What follows is a high level overview of the GA being applied in this work with justification for design choices. For more details see [95, 98, 101, 102, 103, 104, 96, 105].

The five basic components of a GA are [96]: (1) a chromosomal representation of the possible solutions to the problem, (2) a method to generate the initial generation (population) of n_p chromosomes, where n_p is the population size, (3) an evaluation function to calculate the fitness of individuals, (4) genetic operators that create children from parents for the next generation of n_p chromosomes, and (5) parameter values to configure the GA.

A chromosome uses a finite alphabet $\mathbb{A}$ to encode a possible solution to the problem in a string c such that $c \in \mathbb{A}^n$. Most commonly, $\mathbb{A} = \{0, 1\}$ and the chromosome is called a bit-string. There are advantages to using this alphabet [96]. First, bit-strings can encode a wide variety of information and have been effectively used in many different problem domains. Secondly, such chromosomes have been studied throughly, resulting in a good understanding of how operators behave on said chromosomes and the required GA parameter values.

The most common initialization method is uniform random generation of chromosomes. For general search, research purposes, or when there is a limited understanding of the desired solution, this is the best approach. The main reason is features of the final solution

will have been produced by the GA algorithm (strict methods of evolution) and not by a biased initialization procedure [96].

The evaluation (fitness) function takes a chromosome c as input and produces a fitness value f as output. The evaluation function is critical to the success of the GA. According to [96], $f(c)$ should not stress improvements too much or alternative genetic information in the population will not be considered, resulting in rapid dominance of a single strain. This results in the GA only searching the solution space around the previously most fit individuals and thus converging too quickly on a local suboptimal solution. On the other hand, if $f(c)$ does not stress good improvements enough, good chromosomes might be permanently lost after a few generations resulting in slow convergence on a poor solution. $f(c)$ can be defined to place constraints on possible solutions, for example, by placing penalties on individuals that violate the constraints. However, [96] warns that care must be taken when issuing penalties since time can be wasted evaluating illegal individuals, and the GA may converge prematurely on individuals that have no penalty attached, and genetic information stored in other chromosomes may not be considered because the path to those chromosomes is blocked by the required production of intermediate, highly penalized, chromosomes.

The bulk of computation in a GA resides in the fitness function. When a GA is used for classifier optimization, the fitness function is normally the classifier itself (at least in the wrapper approach). In this case the classifier may have to be trained and tested many times to produce a single fitness value, such as when k-fold cross validation is used to evaluate the classifier. Hence, the complexity of $f(c)$ must be thoroughly considered when designing the GA. Operators such as selection, crossover, and mutation generally have linear complexity and are insignificant compared to $f(c)$ [106].

Genetic operators manipulate chromosomes to produce new individuals. There are three main GA operations in every GA: selection, crossover, and mutation, usually applied in that

order for every generation. For bit-string chromosomes, these have been thoroughly studied and are well understood [96]. Many variations of these have been studied over the last 30 years, but the basic ones explained next are still commonly used in general purpose GAs.

The purpose of selection is to choose which individual's genes will make it to the next generation. Following the genetic principle of survival of the fittest, the selection operator must be implemented such that individuals with higher fitness are more likely to participate in mating, thus passing their genes on to the next generation. One method of achieving genetic like selection is to implement the selection operator using the biased roulette wheel method [95] (Section F.2). This selects individuals based on probabilities proportional to normalized fitness values. The result is that the best chromosomes get more copies in the mating pool, average individuals remain about the same, and the worst fit individuals die off, as required by principle of survival of the fittest.

Once a mating pool is created, the crossover operator is applied. Here, pairs of individuals are chosen randomly and uniformly to mate producing two new individuals. This is repeated until all individuals have a chance to mate. The new offspring is created by crossing the genes of the parents, hence swapping genetic material of both. A common method described by [95] and others is one-point crossover (Section F.3).

Whether crossover of individuals c_i and c_j occurs or not is governed by the crossover probability p_c. If crossover does not occur, (c_i, c_j) move to the next generation unchanged (unless they are modified by mutation; this is described next). Otherwise, the offspring (c_i', c_j') move to the next generation. Crossover is the primary method of diversification and exploration in a GA. It is through this rearrangement of genetic material that the best adapted traits of both parents are spread throughout the population and passed on to the next generation.

The mutation operator (Section F.4) attempts to mimic gene mutation in biological species. In a GA, each chromosome in the current generation is exposed to the muta-

tion operator where each single bit gene has the probability p_m of being mutated. Mutation has the effect of maintaining diversity of genetic information (possible solutions). In some cases this can lead to the development of new desirable traits that gives a chromosome an advantage over other individuals in the current population and may help the GA get past a local optimum [98]. However, using mutation as the single means of diversity results in a random walk through solution space [95]. Therefore, mutation is the secondary, or background, operator to the primary means of diversity: crossover. As a result, p_m is usually very low compared to p_c

A GA will execute until a stopping criteria is reached. Often, the criteria is an *a priori* fixed number of generations, n_g, to execute the GA. Choosing n_g in this manner is difficult. If it is to small, the GA will not explore the problem space and thus will not converge on a solution. If it is to large, once the GA reaches the optimum, the remaining processing time is wasted with no further significant improvements. To overcome this problem, n_g is sometimes selected at runtime when the GA has determined sufficient convergence has occurred.

Choosing values for the fundamental GA parameters (n_g, n_p, p_m, and p_c described above) is a difficult problem. It has been studied throughly for bit-string chromosomes, including the original work by De Jong [99]. The "standard parameter values" derived by De Jong are: $p_m = 0.001$, $p_c = 0.6$, and $n_p = 50$; De Jong did not derive a standard value for n_g. De Jong's standard parameter values are most often used in the literature. Some recommend using the standard parameter values as a starting point and via experimentation tune these parameters appropriately [98]. Interestingly, some have implemented meta-GAs to optimize the GA parameters of the GA for the problem on hand [96, 105]. On the other hand some—such as [103] who attempted to find the ideal values for most situations—conclude that the GA is robust enough to perform well within a fairly wide range of parameter values [101].

3.7 SVM Optimization using Genetic Algorithms

Support vector machine optimization consists of two parts: choosing the optimal SVM parameters and FSS. SVM optimization of this nature is a new subject with most significant work being published after 2000, probably because it was originally thought (e.g. [59]) that FSS is not needed for SVMs; this has since been shown by several researchers not to be the case (e.g. [64]). Most previous works focus on the FSS and the problem of optimizing SVM parameters is not addressed. When parameters are chosen, some undocumented heuristic approach is used. Worse, "default" parameters are chosen without exploration of alternatives. In this work GAs are used to select both the feature subset and SVM parameters. From this point forward, when a GA is used to optimize a SVM, the collective unit is called a GA-SVM. What follows is a summary of the readily available works on GA-SVMs. A common conclusion is that SVMs optimized using FSS with a GA enhances performance.

One of the earliest and most thorough works was done by Fröhlick [107, 108]. His work focused on using a GA for FSS, although the SVM regularization parameter was also selected by the GA. Several experiments were conducted with the aim of comparing his method with the more traditional filter and wrapper approaches. Classifier fitness was set proportional to accuracy and each classifier was evaluated using cross validation and leave-one-out. He concluded that optimizing SVM parameters (as indicated by his use of optimizing C only) is a useful means to improve performance. Also, if their is no constraint placed on the number of features to select (other than $m \leq d$), GAs with cross validation tend to select the fewest number of features. Further, depending on the data set, this GA optimized SVM can show comparable or better classification results compared to those where no FSS has occurred. Even though he only conducted experiments on optimizing C, he suggested that the GA offers the opportunity to optimize all SVM parameters in parallel to feature selection. The author notes that such optimization may be important because the

choice of kernel parameters and C is influenced by the feature subset, and vice versa.

As far as can be determined, this is the only work that investigates optimizing SVMs using a GA for both FSS and SVM parameter selection. As such, it most closely resembles the work conducted in this thesis. However, unlike Fröhlick's work, in this thesis, GAs are used to optimization the feature subset and all SVM kernel parameters, not just C. In addition, the results of optimization are compared against the traditional approach of using a grid search and cross validation to select SVM parameter values. Of all the similar readily available works, all focus on GA feature selection as the sole means of optimization. Unless otherwise stated, in the following cited works the SVM parameter value selection process is not specified and the GA is the basic GA described in Section 3.6.

In [109], a SVM based system was developed to predict mortality in patients with unstable angina. Three SVMs, one for each RBF, polynomial, and linear kernel, were constructed and compared. The hold-out method was used for evaluation. The results showed the RBF-SVM performed better on all performance measures. The GA-SVM selected far fewer features than the other techniques investigated, such as principle component analysis; and when the RBF kernel was used, the GA-SVM also had the best performance on all measures.

[110] used a GA-SVM for classification in a face verification system. Because the number of features was large (up to 4096 features), FSS reduced the memory requirements for the system. The results of using a GA-SVM is compared to using a SVM alone, with the former showing superior performance (reduced dimensionality, increased accuracy) in all experiments. The SVM was constructed using a polynomial kernel of fourth order. The fitness value of the chromosome was considered proportional to the classification accuracy of the SVM.

Yu and Cho [111] published a well documented study of GA-SVMs used to discriminate between an owner entering a password on the computer and an imposture. The SVM

was configured with a Gaussian kernel with SVM parameters chosen using an undescribed heuristic search. The basic GA was used, but with a fitness function that takes into consideration accuracy, dimensionality, and learning time. Evaluation was performed using k-fold cross validation (k unspecified). The results showed a great reduction in dimensionality with a significant increase in accuracy over unoptimized SVMs.

Sun *et al.* [79] argued that FSS is one the most important techniques to enhance detection of objects in images and shows that GAs are a simple and effective method to do FSS. The problems of detecting faces and vehicles are used to test the developed GA-SVM framework. The SVM was configured with a RBF kernel because it was shown to have superior performance on various experiments when compared to other kernels. Three-fold cross validation was used for classifier evaluation. The results are compared to manually choosing what was considered the four best feature subsets of varying sizes and a sequential method. The GA-SVM selected the fewest features and had the highest accuracy.

Schröder *et al.* [112] also used a GA-SVM for FSS. In this research the goal was to select suitable features (channels) from electroencephalography signals to enhance brain-computer interface communication and serve as a basis to discriminate brain states in human subjects. The SVM configuration is not specified. In all cases the fitness function was accuracy, and ten-fold cross validation was used for evaluation. The results were compared to (1) SVMs with all features selected, (2) SVMs with physiologically motivated selected features, and (3) SVMs with exhaustively selected features. The GA-SVM is statistically comparable to (3) in number of features selected and accuracy attained.

3.8 Design of Experiments for GA Calibration

A significant problem in applying GAs to any problem is selecting the "control" parameters, not limited to but including, mutation rate, crossover rate, and population size. The

significance of choosing these parameters is well documented (e.g. [99, 103, 113, 95, 105]). However, there still lacks a systematic, reliable way to select these values,—that is, calibrate the GA [114]. In the literature most researchers either blindly use De Jong's standard parameters, use the standard parameters as a starting point and tweak them, or "derive" the "optimal" values via intuition, prior knowledge, and random experimentation. Some (e.g. [98]) have suggested using hand optimization: start with De Jong parameters and change each parameter one at a time. This is the well known, but frowned upon, OFAT (one factor at time) experiment methodology. The major problem with the this approach is that it does not consider factor (parameter) interactions. In this work it is proposed to use DOE to construct a model for calibration; call this process DOE-GA.

DOE is a theoretically sound, systematic statistical approach to efficiently and effectively investigate the effects of multiple factors (parameters) on a phenomenon. The technique is a combination of experiment design, analysis of variance (ANOVA), and regression analysis. DOE theory and practical usage is described thoroughly in [115, 116, 117, 118].

DOE allows a mathematical response surface model to be constructed. The model can be used to analyze a response, make predications, and optimize a response by selecting the best factor levels to achieve some response goal. It is important to stress that unlike other methods (e.g. OFAT), DOE accounts for multi-factor interaction. This occurs when the effect of one factor is influenced by the level of another.

DOE is used extensively in chemical, manufacturing, and to a lesser extent, civil engineering [119]. However, it is rarely used in the computing and intelligent systems fields. As pointed by [120], this is a missed opportunity. As such, using DOE to calibrate a GA is a relatively unexplored research area. The papers published are all recent and exclusively focus on DOE for calibrating GAs for scheduling problems. As far as can be determined similar techniques have never been investigated where GAs perform classifier optimization.

One of the earliest and most thorough works—and not specific to scheduling—was con-

ducted by [114]. The purpose of this work was to derive a general parameterization procedure based on DOE. The multi-factored constrained optimization problem with a known solution is used to investigate this approach. The factors studied are mutation rate, crossover rate, and population size. A 3^3 full factorial design with five replications was used. In each replication the GA started with a new seed to randomly generate the initial population (initialization). The results suggest that individual factors alone are not important, but the interaction between them is very important. This highlights the importance of considering the GA parameters together, not individually; hence the conclusion that "these parameters cannot be optimized by studying them one at a time, in an isolated manner". Although a comparative study is not presented, the authors state that tests indicate the DOE approach is as effective as other methods currently in use. Moreover, they conclude that DOE-GA should be more effective than the current set of guidelines and rules of thumb.

[121] used DOE to select the best settings for a bi-criteria genetic algorithm developed specifically to minimize scheduling costs for a complex job-shop scheduling problem. The GA factors chosen for study were the ratio of population size and generation size, crossover rate, mutation rate, crossover operator, and mutation operator. The authors chose a sequential DOE strategy to reduce experiment execution time. First, an efficient L4 fractional factorial in a eight level Latin-square with two replications, each started with a new random seed, screening design was executed. Based on these results, a 2_{IV}^{5-1} fractional factorial with two replicates was conducted to investigate significant factors from the screening experiment. The final model was built using the crossover operator, mutation operator, mutation rate and the interaction of crossover operator and population/generation ratio. The results were not conclusive. Additionally, crossover rate and mutation rate were found to be insignificant, which conflicted with other researchers work. As a result, they conclude that significant GA factors and their interaction are application specific.

[122] also performed a sequential study on job-shop scheduling, albeit with different

factors, namely crossover rate, mutation rate, population size, generation size, problem complexity, and length of the block swapped in crossover. For screening, a 2_{IV}^{6-2} design was used. The results suggested that population size is not important, which disagrees with current thought. In the subsequent 2_{IV}^{6-2} design, narrower experiment ranges for population size and generation size was used to confirm the results of the sequential experiments. The results did not agree. The authors explain this contradiction is because the surface is nonlinear over the entire design space. As a result, they conclude there is no general guideline in setting the GA parameters to achieve the best solution for all problems. Rather, setting parameters is problem specific.

A more recent paper [120] investigated the use of DOE to (1) optimize GA parameters, (2) demonstrate that this technique can be more widely used in modeling and optimization of other computer programs, and (3) show that applying DOE in a sequential strategy is the best for computer programs. As in the above works, the GA being investigated is designed to optimize a scheduling problem. Several designs where considered, all using population/generation combination, mutation rate, and crossover rate as factors. A 3^3 full factorial design with five replicates, L_{8+1} fractional factorial, a 2^3 central composite design (CCD), and a Box-Behnken design (BBD) were used. In the end, the authors conclude a sparse design (minimum runs) such as the 2^3 design will often achieve results comparable to more complex designs. If analysis of the results suggests so, further experiments can be easily added, for example, forming a CCD or BBD. Later trials are easily added without penalty because, when applying DOE to computer programs, there is no change in response over time if the same GA parameter settings and initialization are used. The sequential strategy can also lead to significant savings in time and resources if complex, computationally intensive programs, such as GAs, are being investigated.

Some other related work involving the use of GAs and DOE (but not using DOE to calibrate the GA) are summarized by [122]. These included [123, 124, 125, 126].

Chapter 4

Image Segmentation

This chapter describes the development of image processing methods used to segment objects that resemble whales from the raw images. This is step two in the general approach described in Section 3.1. The chapter starts with a description of the data set, including common features seen in most images. This is followed by sections describing the main algorithms developed, followed by a discussion of the results.

4.1 Data Set

The raw data set consists of 6 rolls of 9"×9" color aerial photograph negatives. These were taken during surveys conducted by DFO in August of 1995 and 1997 over the St. Lawrence River estuary. The survey was flown using a strip transect line technique described in Section 2.3 at an altitude of 4000 feet. The fixed winged aircraft's altitude and position was controlled by a certified pressure altimeter and a satellite-linked global positioning system (GPS). A large format (9"×9") mapping camera was attached to the aircraft and fitted with 6" lenses, 420 nm 2× filters, and a motion compensation system. The initial, ideal conditions were a sun angle $\geq$ 30°, winds $<$ 10 knots, a ceiling of 4000 feet, and no fog [127].

The film negatives were read by DFO scientist and technicians using methods described in Section 2.3. The results were recorded on paper with grid lines matching the acetate covering the negatives. Using this method, whales (adults and young), wave crests, land, sun glare, boats, and other objects were marked. These were provided with the data set and acted as the ground truth data for algorithm development and testing.

Each negative frame was examined with the goal of picking a representative set for scanning. All frames in the selected set contained whales or other features typical in aerial photographs. A total of 103 frames were selected for scanning. Aerial photographic film requires specialized scanning equipment that is capable of scanning large format negatives at high resolution. Hence, the images were scanned by a third party company that specializes in such tasks.

The higher the scan resolution the better the quality of the final image, and the more pixel data is available for analysis. However, higher scan resolutions results in larger images, thus increased data storage and computer processing power to analyze the image. Additionally, the price of scanning each image increases with scan resolution used. On the other hand, low resolutions make it difficult to sensibly segment whales and extract discriminating features. Taking into consideration the above factors, the selected frames were scanned in color (24-bit, 3-band RGB, TIFF) at a resolution of 907dpi (resulting in 8430×8429 pixel images) to produce as much detail as possible while maintain a reasonable image size ($\approx$ 213 MB). Triathlon Incorporated (Richmond, BC, Canada) scanned the images at about $47 per frame and stored the digital image library on 35 CD-ROMs.

4.2 Features in Aerial Photographs

This section describes the most common features seen in the aerial images. As expected, nearly all of these features appear in the source negatives as well. The intent is to give an

overview of the features that a reader would encounter in manual inspection of a typical image (or negative), how the reader would handle such features, and to develop a better understanding of the challenges involved in developing an automation technique. Examples of typical image features are shown in Figure 4.1 and examples of common whale forms are shown in Figure 4.2.

4.2.1 Non Whale Features

A typical image consists of dark navy blue to light blue-green water with some sun glare and an associated halo of wave crests concentrated in one part of the image. Sun glare (Figure 4.1a) occurs in almost every image. It is typically localized to one area of the image, usually the side or corner, and generally covers between 10 and 30% of the image. It is generally shades of white, and rarely shades of red. Areas of intense sun glare are considered unreadable because they completely mask the water.

Wave crests (Figures 4.1b) are typically manifested as sun reflecting of waves. These are generally seen in highest density as a halo around the most intense sun glare and diminishes in intensity outward from the sun glare center. Smaller, more oval shapes, are called sun speckles (Figure 4.1c). When a collection of high density wave crests are observed, that area is marked as unreadable because whales are completely masked. However, if the density of waves crests is sufficiently low whales can sometimes be identified; although with great difficulty because wave crests often resemble whales. In most cases, whales distinguished in such areas are labeled uncertain. When whales are identified, it is because the waves are not elongated or are of a different size than typical whales. Sometimes whales are distinguished because they are swimming in a direction approximately perpendicular to the predominant wave crests (Figure 4.2h). When wave crests are isolated, it can be very difficult to classify it as a whale or a crest.

A related feature to wave crests and sun glare are wave fronts (Figure 4.1d). These

are predominantly elongated, snake like patterns that cross the image. They vary in color from shades of white to light blue. The texture can resemble fog or clouds with a wispy appearance, or they can appear as a dense collection of very small wave crests. Parts of these fronts can appear as whales deep under the water surface. Whether they are marked as unreadable or not depends on their ability to mask clear water, which is large influenced by the pattern, color, and size of the front.

A smaller percentage of images ($\approx$ 10% of the data set) have land in them (Figures 4.1e and 4.1f). Readers identify land, estimate its percentage of total image, and mark the area as unreadable. Varying percentages of land cover can be present, from 0 to 100%.

All images have a near black to black border that covers all four sides of the image. This is an artifact of scanning negatives and photographs. About 7% of the image is lost to borders.

Other objects more rarely seen are boats, buoys, clouds, and foam layers on top of water (Figures 4.1g, 4.1h, and 4.1i).

4.2.2 Whales

A typical image has 0 to 10 whales. In digital images at the resolution used here, the maximum whale size is about 25 $\times$ 7 pixels. Because they are located in a 8430 $\times$ 8439 pixel image, they are nearly indistinguishable unless viewed under height magnification. Adult whales located at or very near the surface of the water appear as elongated white objects surrounded by darker blue-green water (Figure 4.2a). These whales are the easiest to detect and mark correctly. The main challenge is when they appear in odd shapes or are curved because they are turning, diving, or splashing (Figure 4.2e).

Whales can be located deep in the water (Figure 4.2b). These are not easily identified since their characteristic white, elongated body is masked by water. These whales often appear as oval to round light blue-green shapes distinguished from the surrounding water

mostly because of the slight increase in average intensity. Slight variations in water texture and color, caused most often by partial wave fronts, sun glare, and wave crests, can have a similar appearance. Hence readers often mark these whales as uncertain.

Young whales (Figures 4.2f and 4.2g) are especially problematic, and thus many are labeled as uncertain. These whales are often less then half the size of adults. The smallest of juveniles lose the characteristic elongated shape and appear very much like small wave crests, small wave fronts, sun speckles, or very deep adults. When young are deep, they are nearly indistinguishable from surrounding texture and color variations in the water. Juveniles are most often identified by their close proximity to adults, often being partially occluded by an adult.

Belugas are often seen as pods ranging from 2 to 12 whales (Figure 4.2c). Isolated whales in pods are clearly distinguishable. Often, however, whales are very close to, touching, or partially occluding others. This makes it very hard to distinguish one whale from the other (Figures 4.2c and 4.2i). Young whales in pods are particularly difficult to identify because they can appear as a partially occluded whale or a splash from another whale.

4.3 Challenges

Following are the main image processing challenges.

- Segmenting natural objects in natural scenes, which implies varying feature values, such as size, shape, and color.
- Identifying whales from resembling wave crests.
- Detecting deep whales.
- Separating touching and occluded whales.

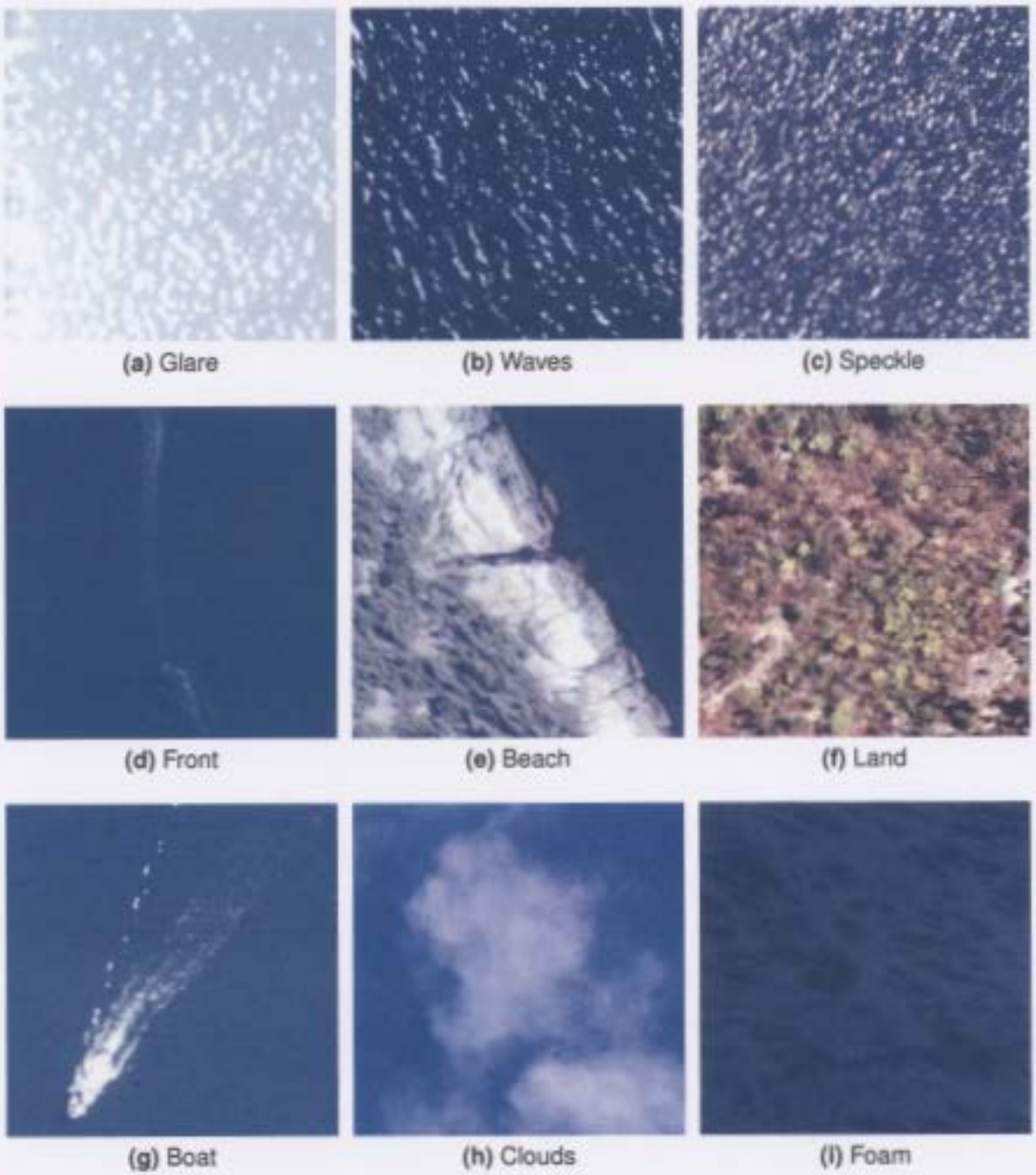


Figure 4.1: Subimages of features typically found in images.

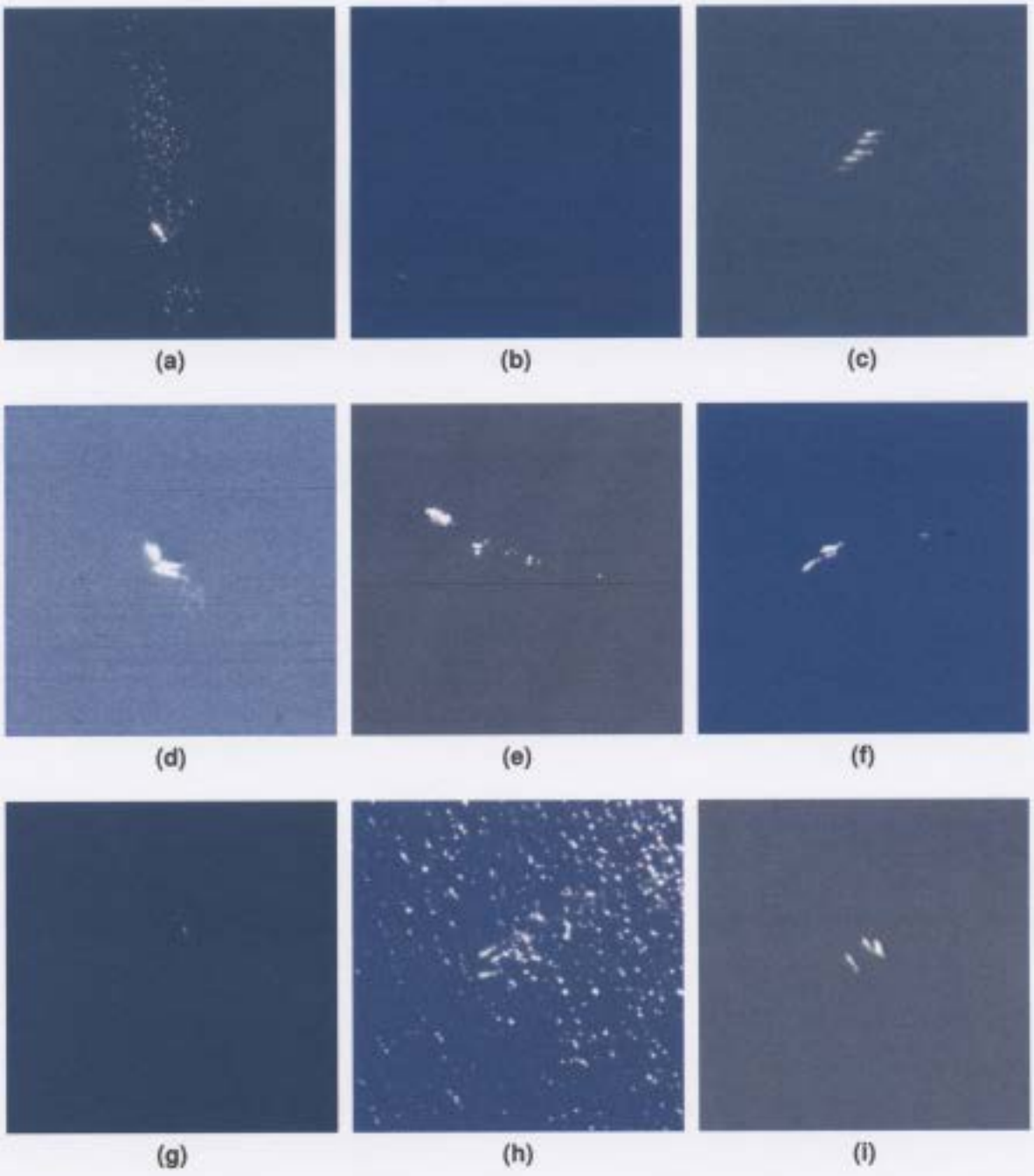


Figure 4.2: Subimages of common whale forms. (a) A single adult whale in sun speckles. (b) Two deep whales. (c) A pod of five isolated whales. (d) A pod of four touching and occluding whales. (e) A single whale splashing water. (f) Two adults and two young. (g) A single young whale. (h) Three whales in an area of wave crests. (i) A pod of four whales, with one occluding another.

- Identifying non-whale objects: wave crests, sun glare, land, boats, wave fronts, and clouds.

- Targets are small relative to the image; a whale is a maximum of 25×7 pixels in a 8430×8429 pixel (203.46 MB) image.

4.4 The High Level Segmentation Algorithm

The high level image segmentation algorithm consists of three steps: (1) create an "unreadable" mask and apply it to the source image, (2) segment the source image using an adaptive thresholding algorithm, and (3) perform secondary segmentation and blob analysis. Sections 4.5, 4.6, and 4.7 describe the development of each step in this algorithm, and the chapter ends with a discussion of the results.

4.5 The Unreadable Mask

Areas in an image that a reader (the person doing the counting) would consider "unreadable" typically consists of land, sun glare, extensive waves crests, or image borders; that is, places where it is difficult or impossible to observe whales. The unreadable mask algorithm creates a mask so subsequent image processing algorithms can ignore these areas.

The first step in the mask algorithm is to reduce the image size by 90% using a nearest neighbor interpolation resizing algorithm. This algorithm is fast but produces low quality images, which is acceptable for creating the unreadable mask since small details are not important. This resizing step is key in reducing the computation time for the mask algorithm, indeed the overall image segmentation algorithm. Even though it takes time to resize the image, this time is small compared to the time required for the subsequent filtering operations. Without this step, the time spent creating a mask would prevent practical use of

the software. Without resizing, the mask algorithm takes more than 47500s $\approx$ 13.2h to process a typical 24-bit, 8430×8429 (203.36 megabytes) image (benchmarked on a 900MHz computer with 500 megabytes of RAM). With resizing, it takes 160s.

The next step consists of a filtering (pixel classification) operation. Here, a 5×5 pixel window is centered on each pixel in the image and the origin (center) of the window is classified as readable or unreadable. The end result is a binary image. If any of the following criteria are true, the origin is labeled unreadable:

1. The origin is nearly[1] black. This typically indicates the pixel is on the image border or in a very dark shadow caused by objects of high relief. Objects of high relief are either on land (e.g. buildings) or are unreadable objects on the water, such as a boat.

2. The mean of the window is high. This occurs in areas of intense sun glare, wave crests, or beach rock.

3. The texture of the window is not smooth. Clear water has a smooth texture, non-water regions and wave crests often do not (Figure 4.3). The standard deviation of the gray values of pixels—calculated by taking the average of the red, green, and blue bands—covered by the window is used as a simple measure of texture [128].

4. Any pixel covered by the window has a red component larger than the corresponding blue component. This rule reflects the fact that only non-water pixels have red $\geq$ blue components. This is because higher light wavelengths, such as red, are absorbed more by water than the lower wavelengths, such as blue and green, so the red component is always lower than the blue and/or green. However, it is not sufficient to rely on the "fact" that water is "blue" and land is not because variations in atmospheric conditions often result in land regions having a blue component greater than red or

[1]Unless otherwise specified, values for parameters in all algorithms in this chapter are empirically derived.

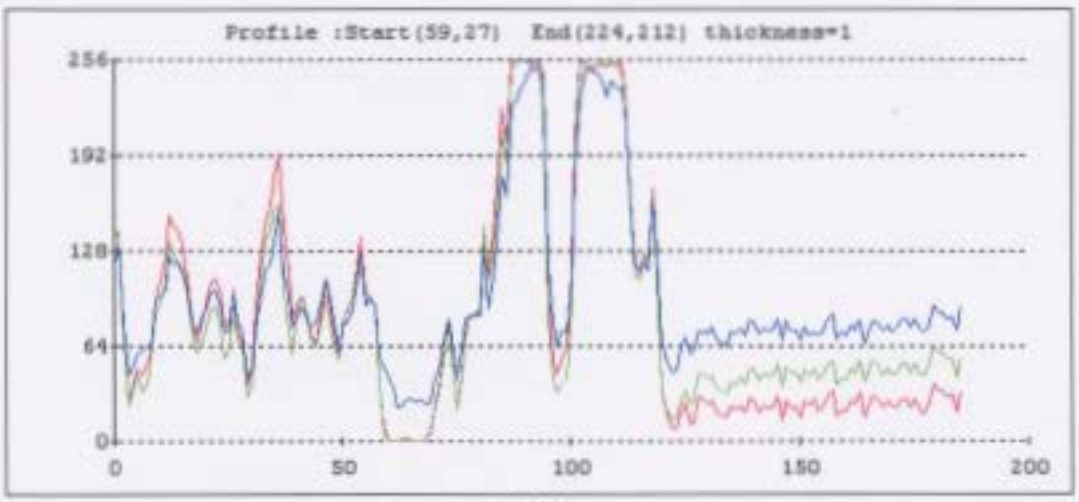
green (Figure 4.3). In addition, depending on water depth, water temperature, sun direction, and bottom color, water often has shades of green similar to land.

The above filtering operation does not label every pixel in the image correctly. In particular, it forms binary regions (blobs) of varying sizes and odd shapes with many holes. To create a uniform, smooth mask, a post processing algorithm is applied to the binary image created (Figure 4.4). First, small blobs are deleted from the mask. This ensures blobs representing pods are not included in the mask (individual whales are automatically removed because of the rigorous image size reduction). Then, small holes in the remaining blobs are filled and regions that are nearly touching are joined using an aggressive dilation operation [128] (30 iterations with a 3×3, 8-connected symmetrical structuring element). The dilation operation also increases mask area in an attempt to cover transitional areas between unreadable and water areas (e.g. wide beaches, less intense wave crest). Filling holes and dilation can be safely applied (i.e. without worrying about masking whales) since small patches of water are not generally surrounded by regions of non-water (e.g. land and wave crests), or small spaces between regions are likely unreadable regions misclassified as water. Even if such regions are truly readable water, they are generally too small to search in for whales. Finally, all remaining blobs are either closely packed pods of whales or unreadable areas. Taking into account the previous dilation and scaling factor, blobs that are small enough to represent pods or individual whales are removed. The last step in the algorithm is to rescale the mask back to the original image size.

4.6 First Phase Image Segmentation

A variant of the adaptive thresholding technique [128] is used to segment potential whales. This allows localized processing of the large (8430×8429 pixel) source image, thus allowing different thresholds to be calculated for different parts of the image. This is necessary

(a)



(b)

Figure 4.3: Land-water color band cross section. (a) A subimage of a sharp shoreline with cross section line shown. (b) RGB intensity profile at the cross section line (from left to right). The center peaks are the shoreline beach rocks. To the left is the coarse land profile and to the right is the smooth water profile.

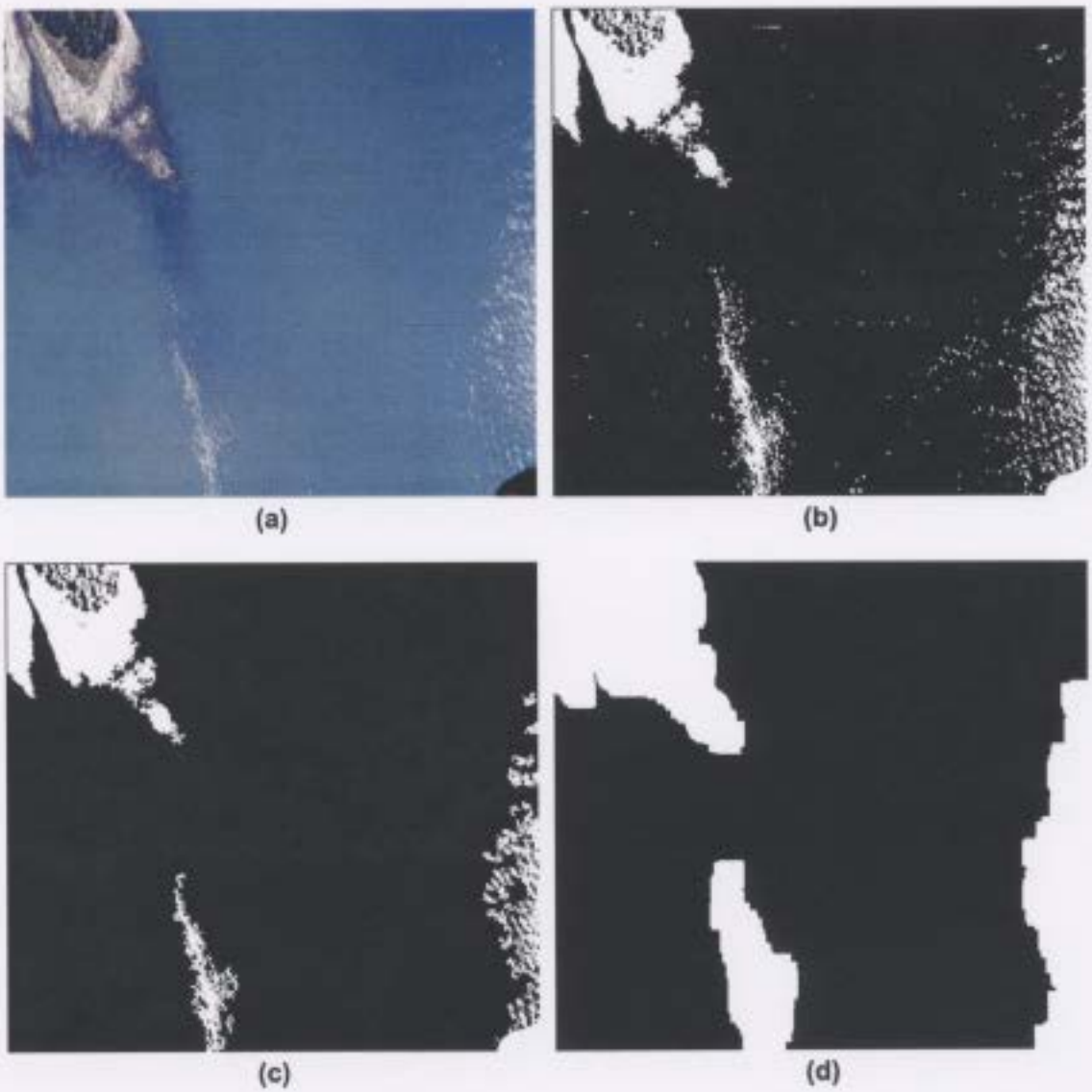


Figure 4.4: Subimages showing key steps in the unreadable mask algorithm. (a) Colored source image. (b) Binary image after segmentation. (c) Binary image after blob filter applied. (d) Binary image after holes filled and blobs dilated.

because (1) there are large intensity variations across the entire image, and (2) the total area covered by whales is statistically insignificant in relation to the size of the image.

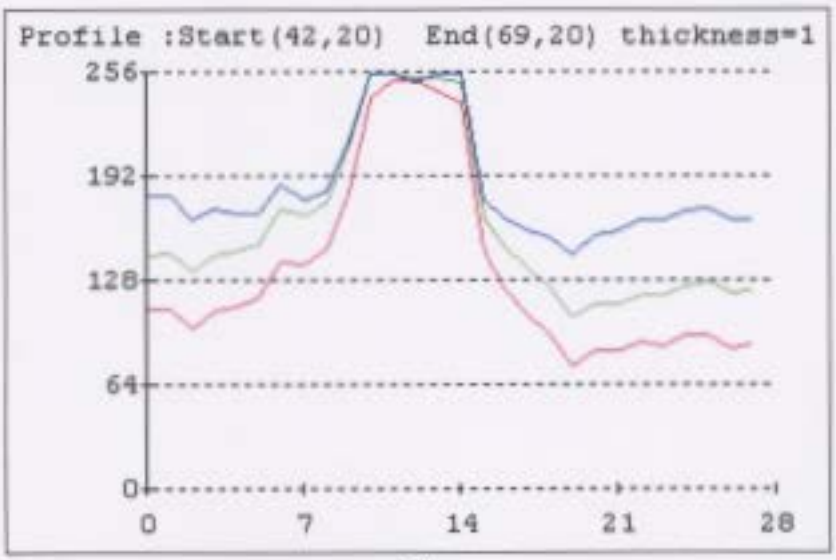
First the image is divided into a grid of 250×250 pixel, non-overlapping subimages. The subimage size is chosen such that intensity variations are minimized and there are significant statistics to obtain a well shaped normal distribution of the background intensity. The grid is created as follows. First, from top to bottom, as many 250 pixel wide rows as possible are created. If there is image area remaining at the bottom, and the area is greater than 0.75×250 pixels wide, the entire area is divided into two equal wide rows, otherwise the area is append onto the existing last row. The same technique is used to create columns from left to right.

The following algorithm steps analyze each subimage and combines the results back into a single image. It should be noted that regions covered by the mask created in Section 4.5 are not processed in any of the following steps. The first step is to create gray scale subimages for thresholding. These images are created from the red band of the source image, versus taking the average of all bands, since the biggest contrast between whales and the background is in the red band (Figure 4.5). This is to be expected given that whales reflect all light wavelengths well, as they are white in color, but the surrounding water absorbs wavelengths represented by red more than other colors.

The second step calculates the first order statistics mean μ and standard deviation s of each gray scale subimage. This is used to threshold the subimage at $T_1 = \mu + 3s$, which is about 99.85% cumulative distribution. This threshold works because the whales have a higher intensity than the background, are significantly smaller than the background (i.e. have fewer pixels), and the background intensities generally forms a normal distribution. In general, if whales exist they represent a very small proportion of the highest intensity levels. Of course, other high intensity objects are also found.

The results of thresholding are binary blob subimages representing potential whale tar-

(a) (b) (c) (d)



(e)

Figure 4.5: Example of foreground to background contrast in different color bands. (a) Colored subimage of six whales with cross section line shown. (b) Green color band. (c) Blue color band. (d) Red color band. (e) RGB intensity profile from left to right at the cross section line. The peak where all color bands are approximately equal represents the whale. To the left and right is background water.

gets. In this step, a simple filter is applied to remove objects that are obviously not whales. The filter consists of removing all blobs with an area greater than an empirically derived maximum size for a pod of whales, and all blobs not touching the subimage edge that are less than an empirically derived minimum size for a single whale. Small blobs along subimage edges are not removed because they could represent whale(s) that span adjacent subimages; these are dealt with implicitly in the secondary segmentation algorithm in the next section.

The final step of the algorithm is to recreate the full sized binary image from the binary subimages. This is done using a binary OR operation [128] between each subimage and the full sized image, which has all pixel values set to 0. The subimage grid is used to correctly position each subimage in the full sized black image.

4.7 Secondary Segmentation and Blob Analysis

The previous algorithm creates a full sized binary image, where each blob is potentially a whale or pod of whales. The following algorithm "zooms in" and analyzes each blob to improve segmentation. This includes separating close whales that are respresented by a single blob and fine tuning segmentation of both separated whales and blobs that represent single whales. The algorithm is illustrated in Figure 4.6.

The first step (Figure 4.6a and 4.6b) creates a small binary and gray scale subimage centered on the blob of interest. The subimage size is equal to the blob's bounding box inflated [2] by six pixels to ensure parts of the object that may have been missed by initial segmentation in Section 4.6 are included. The inflation value of six pixels is a conservative estimate based on experience with the segmentation algorithm.

Next, a watershed algorithm (Chapter D) is used to create a mask to separate adjacent

[2] A rectangle inflated by n pixels is defined as the original rectangle with top, bottom, left, and right boundaries extended by n pixels.

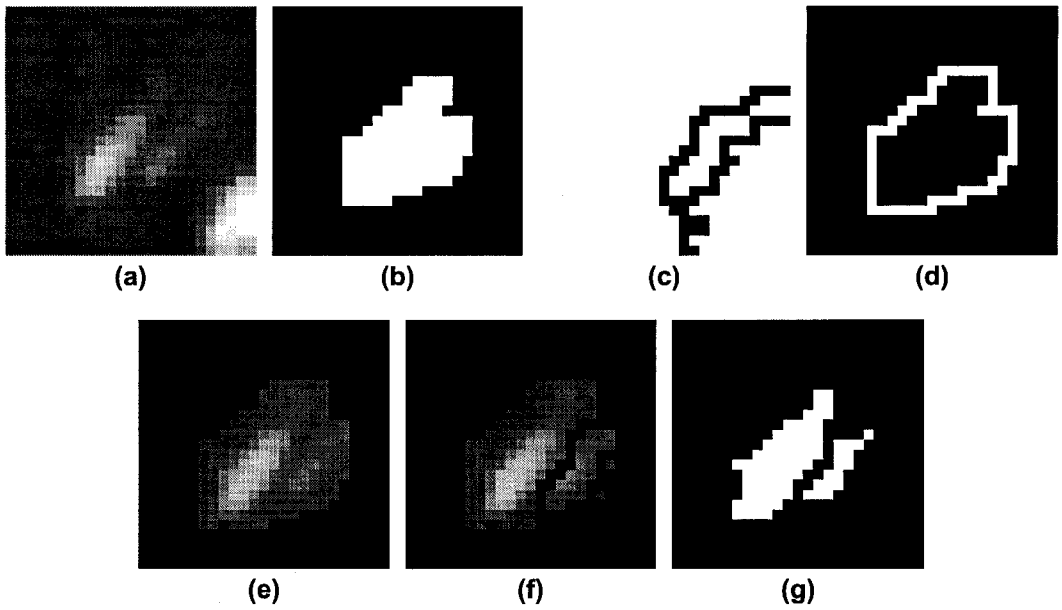


Figure 4.6: Example of the main steps in the secondary segmentation algorithm. (a) Gray scale subimage of an adult and adjacent young whale. (b) Subimage of original target blob as thresholded in Section 4.6. (c) Watershed lines derived from gray scale image in (a). (d) Object boundary. (e) Dilated gray scale blob (ROI). (f) Watershed line image in (c) applied to (e). (g) Results of thresholding (e) at T_2 and applying blob filter.

whales (Figure 4.6c). If two or more whales are adjacent to each other, but not occluding each other, a hill-valley type intensity profile exists (Figure 4.7); the whales represent hills (high intensity) and water represent valleys (low intensity). The watershed algorithm draws lines in valleys between catchment basins (lowest points in a valley), thus separating touching whales. Catchment basins are determined from pixel intensity minima; when a minima has a difference in intensity $\geq v_{min}$ (the watershed minimum variation) from the closest maxima it is considered a catchment basin. If v_{min} is too low, watershed lines are formed incorrectly, often cutting across whales (Figure 4.8). On the other hand, if it is too high, watershed lines are not formed in the valleys. Through experimentation, a suitable value for v_{min} is 20 (on a scale 0–255).

Separating adjacent whales using simple thresholding does not work. First, determining a dynamic threshold is difficult. Second, when applying the threshold, the valley between the adjacent whales is often incompletely segmented. The watershed technique above does

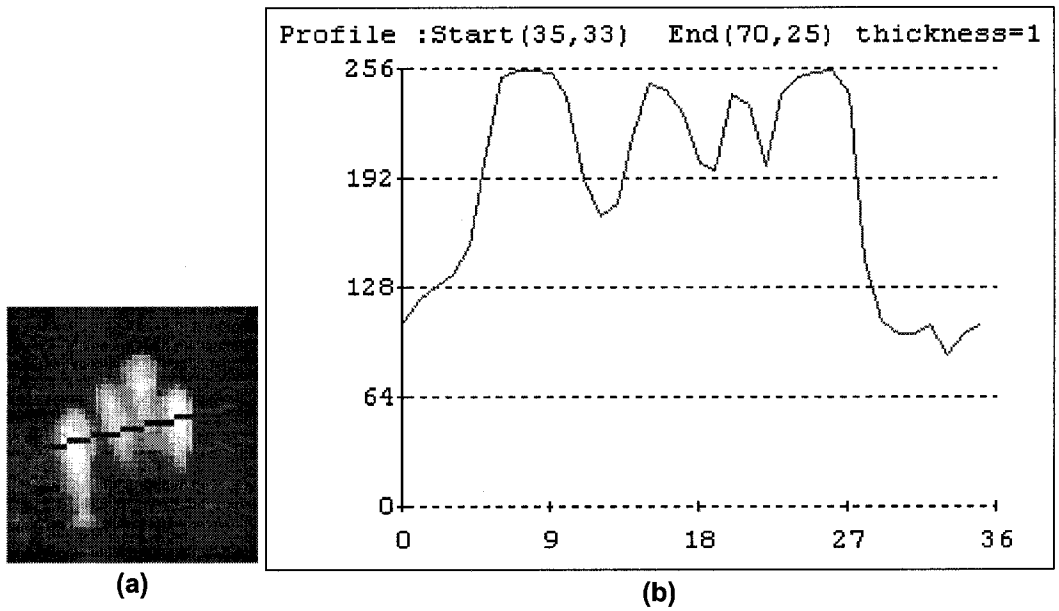


Figure 4.7: Example of the hill-valley intensity profile across adjacent whales exploited by the watershed algorithm to separate adjacent whales. (a) Cross section line drawn from left to right across four adjacent whales. (b) Intensity profile from left to right along the cross section line. The four peaks represent four whales. The valleys in between are where the watershed lines are drawn.

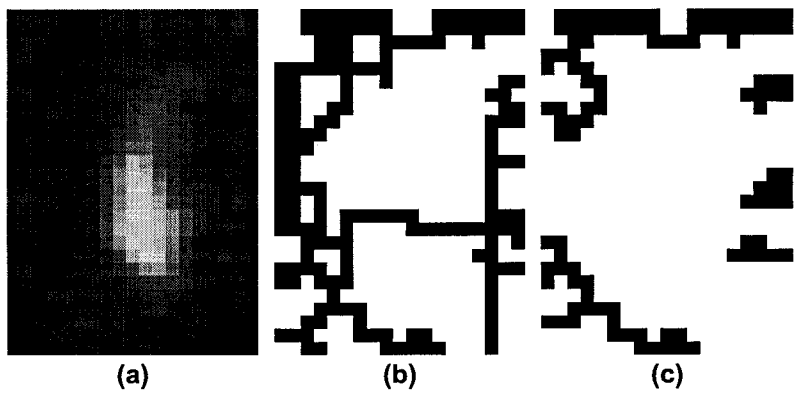


Figure 4.8: Example of whale bifurcation when using a low v_{min}. (a) Gray scale subimage of a single whale. (b) Watershed line cutting whale when a $v_{min} = 3$ is used. (c) Correctly formed watershed lines (not cutting a whale) when a $v_{min} = 6$ is used.

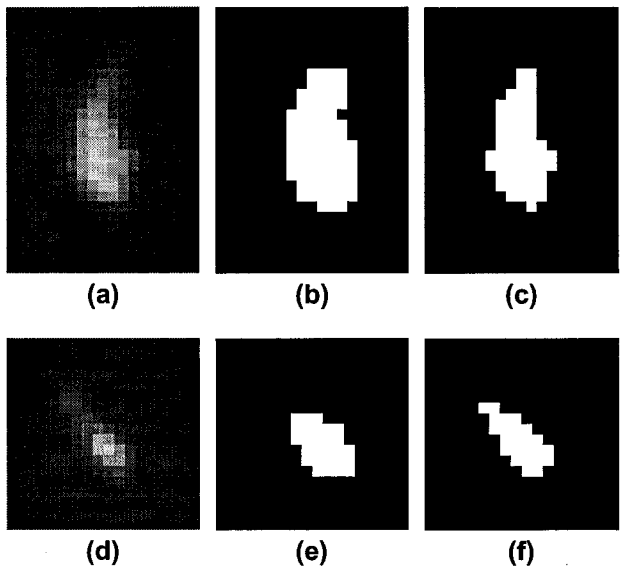


Figure 4.9: Examples of object resegmentation using foreground and background intensities. (a) and (d) are original gray scale subimages of single whales. (b) and (e) are results of initial segmentation using T_1 (Section 4.6). (c) and (f) are results of resegmentation using T_2. Note the much finer detail in the resegmented subimages.

not suffer from these problems because watershed lines are always continuous and thresholds are not used for segmentation.

Before applying the watershed mask, a resegmentation threshold T_2 is calculated. As opposed to T_1 calculated in Section 4.6, which resulted in rough segmentation, T_2 is based on the object foreground and adjacent background intensity, thus it is finely tuned to an individual region of interest (ROI) around the object. T_2 is calculated as the average of the average background (m_1) and average foreground (m_2) intensity: $T_2 = (m_1 + m_2)/2$. m_1 is calculated from the gray values of pixels along the boundary of the blob (Figure 4.6d). The boundary pixels are determined by dilating the original binary blob and subtracting the original blob from it. The gray values of pixels corresponding to the remaining binary pixels represent the background. m_2 is calculated from the gray values of the original blob pixels. Results of T_2 thresholding are demonstrated in Figure 4.9.

The final step extracts a gray scale ROI and applies the watershed mask and T_2 calculated above. The ROI is created by dilating the original binary blob and copying the

corresponding pixels in the gray scale image to a new image (Figure 4.6e). The watershed mask and the ROI image are combined using the logical AND operation [128], thus separating any adjacent whales (Figure 4.6f). The result is then threshold using T_2, forming a new binary subimage with whales accurately resegmented and adjacent whales, if present, separated. Any spurious small or very elongated blobs are removed (Figure 4.6g). The final segmented subimage is then merged back into the full sized image using the logical OR operation.

4.8 Results

The unreadable mask algorithm creates an adequate mask for all images tested; all land, sun glare, wave crests, and borders are covered. However, the dilation step results in the mask covering portions of clearly readable water if the boundary between clear water and unreadable areas is sharp, such as along image borders and sharp coastlines and wave crests. In such cases, whales close to the boundary could be covered by the mask. For borders this is a significant problem because borders are present in every image. If whales can be covered by the image border mask, the user is forced to manually examine four sides of every image. It is not as severe a problem along shorelines since whales are generally not found close to land and land is not present, or present in small amounts, in most images. In the case of wave crests, the user must manually search in such areas covered by the mask anyway since the existing algorithm can't process such areas. In addition, such areas are small and well localized in most images. Another problem area is that shallow water is not masked. As a result, variations in the sea bottom or surface-protruding objects (e.g. rocks) are sometimes detected as whales.

The adaptive thresholding algorithm (Section 4.6) works well. Only the most obscure of whales are missed, such as those with very low contrast (Figure 4.10a). Less than 1%

(a) (b)

Figure 4.10: Examples of whales that cause image processing problems. (a) Low contrast whale. (b) Four occluded and touching whales.

of whales are missed this way, and most of these are labeled uncertain by readers. In an attempt to keep this percentage as low as possible, the algorithms over segment the image so that many objects found are not-whales. Segmented not-whale objects mostly include localized changes in intensity caused by wave fronts and wave crests. However, the blob analysis algorithm in Section 4.7 and the pattern recognition algorithm described in later chapters eliminates many of these false positives.

The blob analysis algorithm (Section 4.7) improves initial segmentation and separates many adjacent whales. The biggest problem is that the conservative (high) minimum variation level for the watershed algorithm results in many adjacent whales not being separated. However, if the variation level is lowered, many individual whales are bifurcated as shown in Figure 4.8. Another problem is that occluded whales are not separated because the intensity valley between them is small or non-existent (Figure 4.10b). Whales not separated or cut by watershed lines are at risk of being filtered out in the blob analysis algorithm or misclassified as not-whales by the pattern classifier described in later chapters.

Chapter 5

Calibrating GAs Using DOE

This chapter focuses on the task of selecting parameter values for the GAs that are used for classifier optimization in the next chapter. The details of the GAs and classifiers used are described in that chapter where it is more relevant; such details are not necessary to understand the contents this chapter. For this chapter it is sufficient to have read Chapter 3, in particular the sections on classifier optimization (Section 3.4 and Section 3.7), genetic algorithms (Section 3.6), and DOE (Section 3.8).

Choosing GA parameter values is a difficult problem (Section 3.6 and Section 3.8), compounded by the fact that the values often depend on the GA application. In this work it is proposed to use DOE as an aid in selecting the parameters. Traditionally, values are determined using intuition, choosing several different combinations and running trials, using the so called "standard" settings proposed by De Jong [99], or some combination thereof. Using the systematic DOE approach, classifier specific GA parameter models are created and the optimal (within the given design space) set of parameters and values are selected from the theoretically sound, statistically derived models. Some (e.g. [114]) have described this approach as calibrating the GA. Henceforth, call GAs that have been calibrated using DOE DOE-GAs.

The general procedure for applying DOE to calibrate a GA follows.

1. Choose responses, factors, and factor levels (Section 5.1.1).
2. Choose a suitable experiment design (Section 5.1.2).
3. Execute the experiments dictated by the design (Section 5.1.3).
4. Analyze the experiment results (Section 5.2). This consists of selecting a candidate model, model reduction (i.e. selecting terms to include in the model based on significance level), hypothesis testing about the model parameters (these last two steps involves heavy use of ANOVA), and analysis of residuals for model adequacy and assumptions (Appendix C).
5. Form the model equation and plot graphs to interpret the model (Section 5.3).
6. Use the model to calibrate the GA (Section 5.4).

5.1 Experiment Design

5.1.1 Response and Factors

The chosen response (y_1) is the balanced accuracy of the classifier, which is the output of the GA's fitness function (Equation 6.1). In this chapter the response is a real number in the range 0 to 1, multiplied by 100 so it is expressed as a percent.

The factors (using standard DOE notation are represented by capital letters A, B, C[1]$\cdots$) chosen for all designs are the GA parameters that govern the fundamental behaviour of the GA. The factors are:

- A – The probability of mutation, p_m, expressed as a percentage ($p_m \times 100$).

[1]Factor C should not be confused with the regularization parameter, C, in SVMs. In this chapter, C always represents a factor unless otherwise stated.

- B – The probability of crossover, p_c, expressed as a percentage ($p_c \times 100$).
- C – The population size, n_p.
- D – Number of generations, n_g.

Others factors were considered but were rejected because they are not of interest in this work and can be easily set to constant values, as described in the experiment design setup below. An example are the SVM classifier parameters C and γ.

5.1.2 Experiment Design Choice and Setup

The chosen design for each experiment is a 2^4 full factorial, face-centered central composite design (CCD) [117] (see also Appendix B), with six center points and two replications. Because this design has more than 2 levels, it is suitable for fitting second order and lower models. Additionally, the replications at the center point and the complete experiment replication allows an estimate of pure error.

The CCD allows approximation of three regression models:

$$\text{linear} : y = \beta_0 + \sum_{i=1}^{k} \beta_i A_i + \varepsilon \tag{5.1}$$

$$\text{2FI (factor interaction)} : y = \beta_0 + \sum_{i=1}^{k} \beta_i A_i + \sum_{i=1}^{k} \sum_{j>i}^{k} \beta_{ij} A_i A_j + \varepsilon \tag{5.2}$$

$$\text{quadratic} : y = \beta_0 + \sum_{i=1}^{k} \beta_i A_i + \sum_{i=1}^{k} \sum_{j>i}^{k} \beta_{ij} A_i A_j + \sum_{i=1}^{k} \beta_{ii} A_i^2 + \varepsilon \tag{5.3}$$

where β_i is the regression coefficient, β_0 is the overall average, ε is the error estimate, A_i is factor i (also called the regressor variable), and k is the number of factors. The method of least squares is used to estimate the parameters in the approximating polynomials [117, 118, 129, 130].

Table 5.1: Experiment parameter settings.

Factor	-1	0	+1	Description
A	0.1	2	3.9	p_m as a percent
B	30	60	90	p_c as a percent
C	25	50	75	n_p
D	10	20	30	n_g

The chosen levels (low = -1, center = 0, and high = +1) are shown in Table 5.1. The values were chosen based on the authors experience using the GA for this problem with consideration for De Jong's [99] standard settings of $p_m = 0.001$, $p_c = 0.6$, and $n_p = 50$. For B and C, the design is centered on De Jong's standard parameters (where B is expressed as a percent). For A, De Jong's setting is used as the low level (expressed as a percent) and the design is centered on p_m calculated by taking into consideration the chromosome length, $l_c = 25$:

$$p_m = \frac{\left(\frac{1}{l_c}\right)}{2} \tag{5.4}$$

This equation ensures significant mutations in a generation, thus plenty of diversity to escape local minima, while at the same time ensuring that not every chromosome is mutated. Statistically, there is a 50% chance that an individual chromosome will have one bit mutated. In the authors experience, this mutation setting typically produces better results than De Jong's setting.

De Jong did not specify the number of generations (n_g) to use, and researchers use many different values. The most important point is to ensure there are enough generations for the GA to coverage. With this in mind, the levels for D were chosen such that the GA would run for a maximum of 30 generations. This maximum is based on previous experiments with the GA and given classifiers, and it ensures the GA finishes in a reasonable time.

For each replication, the GA uses a new uniform random distribution (new seed) to "facilitate determination of real predictors" [120]. Within a replication the same seed was

used for each run. This improves statistical precision by variance reduction [114, 131].

The above 2^4 CCD was executed once for the base classifier in each of the four classifier types described in Chapter 6, Section 6.5: 2-class Bayesian minimum error rate quadratic discriminate (2QD, Section 6.5.1), 3-class Bayesian minimum error rate quadratic discriminate (3QD, Section 6.5.2), 2-class SVM (2SVM, Section 6.5.3.1), and a 3-class SVM (3SVM, Section 6.5.3.1). As stated in the chapter introduction, it is not necessary to understand the details of these classifiers at this point. It is sufficient to know we have four base classifiers and configurations of these will be optimized using a GA in Chapter 6, but to do so we need to calibrate the GA that will use these classifiers. The latter is the focus of this chapter.

5.1.3 Experiment Execution

The CCD in the previous section was executed once for each of the four base classifiers. For each CCD, a GA was configured with one of the four base classifiers. This GA-classifier combination was then run twice for every point on the perimeter of the design space (+1, 0, and -1 levels) and six times at the center (all factors at level 0). A total of $m(2^k + 2k) + m_c = 2(2^4 + 2(4)) + 6 = 54$ experiments are necessary for each CCD, where m is the number of replications, m_c is the number of replications at the center point, and k is the number of factors. In accordance with the basic principles of DOE [117], the order of the 54 experiments was randomized and executed in one block. The response (y_1) was recorded for each run and analyzed using standard DOE statistical techniques. The results are detailed in the following sections.

Table 5.2: Fit summary for 2QD CCD, y_1.

Sequential Model Sum of Squares

Source	SS	DF	MS	F_0	P-value
Mean	4.164×10^5	1	4.164×10^5		
Linear	13.35	4	3.34	6.78	0.0002
2FI	6.07	6	1.01	2.41	0.0427
Quadratic	3.99	4	1.00	2.76	0.0409
Residual	9.96	31	0.32		
Total	4.164×10^5	54	7711.02		

Lack of Fit Tests

Source	SS	DF	MS	F_0	P-value
Linear	14.56	20	0.73	2.20	0.0255
2FI	8.48	14	0.61	1.84	0.0815
Quadratic	4.50	10	0.45	1.36	0.2463
Pure Error	9.57	29	0.33		

Model Summary Statistics

Source	s	R^2	R^2_{Adj}	R^2_{Pred}	PRESS
Linear	0.70	0.3561	0.3036	0.2029	29.87
2FI	0.65	0.5182	0.4061	0.2052	29.79
Quadratic	0.60	0.6246	0.4898	0.2624	27.65

5.2 Analysis of Experiment Results

5.2.1 2QD

Table 5.2 is a summary of fitness for models that can be fitted from the CCD. To choose a model to explore, we pick the highest order model that is significant (Sequential Model Sum of Squares), has an insignificant lack of fit (Lack of Fit Tests), and has the highest and most consistent R^2, R^2_{Adj}, and R^2_{Pred} summary statistics (Model Summary Statistics) [117]. The quadratic model is the best choice: it is significant, it has insignificant lack of fit ($\alpha = 0.1$), and R^2, R^2_{Adj}, and R^2_{Pred} are larger and more consistent than the other models.

Backward elimination [118] with $\alpha = 0.10$ was used to find the "best" subset of regressors to include in the final model. The reduced model statistical test for significance

Table 5.3: Reduced ANOVA and summary statistics for 2QD CCD, y_1.

ANOVA

Source	SS	DF	MS	F_0	P-value
Model	22.54	8	2.82	8.49	< 0.0001
A	0.96	1	0.96	2.88	0.0967
B	0.45	1	0.45	1.35	0.2522
C	7.36	1	7.36	22.16	< 0.0001
D	4.59	1	4.59	13.83	0.0006
D^2	3.70	1	3.70	11.14	0.0017
AB	1.50	1	1.50	4.51	0.0392
AC	3.02	1	3.02	9.08	0.0042
BC	0.98	1	0.98	2.96	0.0920
Residual	14.94	45	0.33		
Lack of Fit	5.36	16	0.34	1.02	0.4695
Pure Error	9.57	29	0.33		
Total	37.48	53			

Summary Statistics

s	0.58	R^2	0.6015
$\bar{y}$	87.81	R^2_{Adj}	0.5306
CV	0.66	R^2_{Pred}	0.4088
PRESS	22.16	Precision	12.277

(hypothesis testing) is summarized in Table 5.3. As can be seen the model is highly significant; there is only a 0.01% chance that a model F-value this large could occur due to noise. Model terms C, D, D^2, AB, and AC are highly significant (P-value $< \alpha = 0.05$) and A and BC are significant (P-value $< \alpha = 0.10$). B is not significant but must be in the model to maintain hierarchy [118]. The lack of fit is not significant relative to pure error; non-significant lack of fit is desirable. R^2, R^2_{Adj}, and R^2_{Pred} are consistent but not high, although the precision (signal to noise ratio) is good (greater than 4). This suggests we have a usable, but not an excellent, model.

As stated by [118], "it is always necessary to (1) examine the fitted model to ensure that it provides an adequate approximation of the true system, and (2) verify that none of the least squares regression (ANOVA) assumptions are violated". Appropriate analysis

Table 5.4: Fit summary for 3QD CCD, y_1.

Sequential Model Sum of Squares

Source	SS	DF	MS	F_0	P-value
Mean	3.128×10^5	1	3.128×10^5		
Linear	12.06	4	3.02	5.90	0.0006
2FI	6.86	6	1.14	2.72	0.0254
Quadratic	2.89	4	0.72	1.86	0.1381
Residual	13.60	30	0.45		
Total	3.128×10^5	53	5901.89		

Lack of Fit Tests

Source	SS	DF	MS	F_0	P-value
Linear	11.27	20	0.56	1.19	0.3299
2FI	4.41	14	0.31	0.66	0.7873
Quadratic	1.52	10	0.15	0.32	0.9686
Pure Error	13.26	28	0.47		

Model Summary Statistics

Source	s	R^2	R^2_{Adj}	R^2_{Pred}	PRESS
Linear	0.71	0.3297	0.2738	0.1597	30.74
2FI	0.65	0.5172	0.4022	0.1784	30.05
Quadratic	0.62	0.5961	0.4473	0.1643	30.57

covering both points have been completed, but to keep this chapter as brief as possible, much of the former and all of the latter has been relegated to Appendix C.

5.2.2 3QD

Table 5.4 is a summary of fitness for models that can be fitted from the CCD. Considering the sum of squares, lack of fit, and summary statistics the 2FI is arguably the best model. However, the quadratic model can be considered because it has similar summary statistics but higher lack of fit.

Starting with a quadratic model with all terms, backward elimination with $\alpha = 0.10$ resulted in the reduced model's ANOVA summarized in Table 5.5. The 2FI model was analyzed as well, but the quadratic model has slightly higher summary statistics. As can be

Table 5.5: Reduced ANOVA and summary statistics for 3QD CCD y_1.

ANOVA

Source	SS	DF	MS	F_0	P-value
Model	20.86	8	2.61	7.29	< 0.0001
A	0.093	1	0.093	0.26	0.6120
B	2.87	1	2.87	8.03	0.0069
C	0.053	1	0.053	0.15	0.7018
D	8.66	1	8.66	24.22	< 0.0001
A^2	2.65	1	2.65	7.41	0.0093
C^2	1.42	1	1.42	3.97	0.0526
AB	1.92	1	1.92	5.38	0.0251
AC	3.97	1	3.97	11.10	0.0018
Residual	15.73	44	0.36		
Lack of Fit	2.47	16	0.15	0.33	0.9890
Pure Error	13.26	28	0.47		
Total	36.58	52			

Summary Statistics

s	0.60	R^2	0.5701
$\bar{y}$	76.82	R^2_{Adj}	0.4919
CV	0.78	R^2_{Pred}	0.3455
PRESS	23.94	Precision	11.642

seen the model is highly significant; there is only a 0.01% chance that a model F-value this large could occur due to noise. Model terms B, D, A^2, AB, and AC are highly significant (P-value $< \alpha = 0.05$) and C^2 is significant (P-value $< \alpha = 0.10$). A and C are not significant but must be in the model to maintain hierarchy. The lack of fit is not significant relative to pure error; non-significant lack of fit is desirable. R^2, R^2_{Adj}, and R^2_{Pred} are consistent but not high, although the precision is good (greater than 4). This suggests we have a usable, but not an excellent, model.

5.2.3 2SVM

Table 5.6 is a summary of fitness for models that can be fitted from the CCD. The quadratic model and linear model are good choices to explore: both are significant, both have in-

Table 5.6: Fit summary for 2SVM CCD, y_1.

Sequential Model Sum of Squares

Source	SS	DF	MS	F_0	P-value
Linear	4.48	4	1.12	14.81	< 0.0001
2FI	0.52	6	0.086	1.16	0.3441
Quadratic	0.68	4	0.17	2.67	0.0470
Residual	2.10	29	0.073		
Total	3.588×10^5	52	6900.67		

Lack of Fit Tests

Source	SS	DF	MS	F_0	P-value
Linear	1.60	20	0.080	1.10	0.3997
2FI	1.08	14	0.077	1.07	0.4266
Quadratic	0.40	10	0.040	0.55	0.8371
Pure Error	1.96	27	0.072		

Model Summary Statistics

Source	s	R^2	R^2_{Adj}	R^2_{Pred}	PRESS
Linear	0.27	0.5576	0.5200	0.4572	4.36
2FI	0.27	0.6220	0.5298	0.3641	5.11
Quadratic	0.25	0.7068	0.5958	0.4187	4.67

significant lack of fit (although the quadratic has a higher P-value), and both have similar R^2_{Adj} and R^2_{Pred} (although the quadratic has a higher R^2 and R^2_{Adj}).

The linear and quadratic models were analyzed, but the quadratic had slightly higher summary statistics. Starting with a quadratic model with all terms, backward elimination with $\alpha = 0.10$ resulted in the model's ANOVA summarized in Table 5.7. The reduced model is highly significant; there is only a 0.01% chance that a model F-value this large could occur due to noise. Model terms A, C, D, B^2, and BC are highly significant (P-value $< \alpha = 0.05$). B is not significant but is added to the model to maintain hierarchy. The lack of fit is not significant relative to pure error. R^2, R^2_{Adj}, and R^2_{Pred} are consistent and relatively high and the precision is good (greater than 4). This suggests we have a good, though not excellent, model which can be used to navigate the design space.

Table 5.7: Reduced ANOVA and summary statistics for 2SVM CCD, y_1.

ANOVA

Source	SS	DF	MS	F_0	P-value
Model	5.48	6	0.91	16.06	< 0.0001
A	0.67	1	0.67	11.76	0.0013
B	1.175×10^{-3}	1	1.175×10^{-3}	0.021	0.8863
C	1.29	1	1.29	22.65	< 0.0001
D	2.55	1	2.55	44.89	< 0.0001
B^2	0.55	1	0.55	9.61	0.0033
BC	0.41	1	0.41	7.14	0.0104
Residual	2.56	45	0.057		
Lack of Fit	0.60	18	0.033	0.46	0.9540
Pure Error	1.96	27	0.072		
Total	8.03	51			

Summary Statistics

s	0.24	R^2	0.6817
$\bar{y}$	83.07	R^2_{Adj}	0.6392
CV	0.29	R^2_{Pred}	0.5762
PRESS	3.40	Precision	16.672

Table 5.8: Fit summary for 3SVM CCD, y_1.

Sequential Model Sum of Squares

Source	SS	DF	MS	F_0	P-value
Mean	1.770×10^5	1	1.770×10^5		
Linear	0.88	4	0.22	12.85	< 0.0001
2FI	0.20	6	0.033	2.23	0.0594
Quadratic	0.27	4	0.069	7.61	0.0001
Residual	0.27	29	9.199×10^{-3}		
Total	1.770×10^5	52	3403.96		

Lack of Fit Tests

Source	SS	DF	MS	F_0	P-value
Linear	0.54	20	0.027	2.77	0.0072
2FI	0.34	14	0.025	2.51	0.0195
Quadratic	0.069	10	6.944×10^{-3}	0.71	0.7081
Pure Error	0.26	27	9.789×10^{-3}		

Model Summary Statistics

Source	s	R^2	R^2_{Adj}	R^2_{Pred}	PRESS
Linear	0.13	0.5223	0.4816	0.4038	1.01
2FI	0.12	0.6398	0.5519	0.3723	1.06
Quadratic	0.095	0.8023	0.7275	0.5806	0.71

5.2.4 3SVM

Table 5.8 is a summary of fitness for models that can be fitted from the CCD. The quadratic model is the best choice for further exploration. It is highly significant, it has a very insignificant lack of fit, and it has the highest R^2, R^2_{Adj} and R^2_{Pred}, and lowest PRESS.

Starting with a quadratic model with all terms, backward elimination with $\alpha = 0.10$ resulted in the reduced model's ANOVA summarized in Table 5.9. The model is highly significant; there is only a 0.01% chance that a model F-value this large could occur due to noise. Model terms A, B, C, D, A^2, AC, and AD are highly significant (P-value $< \alpha = 0.05$) and AB is significant (P-value $< \alpha = 0.10$). The lack of fit is not significant relative to pure error. R^2, R^2_{Adj}, and R^2_{Pred} are consistent and high. The precision is good (greater than 4). Together, these results indicate we have a good model which can be used

Table 5.9: Reduced ANOVA and summary statistics for 3SVM CCD, y_1.

ANOVA

Source	SS	DF	MS	F_0	P-value
Model	1.33	8	0.17	19.72	< 0.0001
A	0.23	1	0.23	27.83	< 0.0001
B	0.11	1	0.11	12.76	0.0009
C	0.20	1	0.20	23.68	< 0.0001
D	0.32	1	0.32	37.88	< 0.0001
A^2	0.26	1	0.26	30.57	< 0.0001
AB	0.030	1	0.030	3.59	0.0648
AC	0.053	1	0.053	6.25	0.0163
AD	0.099	1	0.099	11.76	0.0013
Residual	0.36	43	8.409×10^{-3}		
Lack of Fit	0.097	16	6.081×10^{-3}	0.62	0.8392
Pure Error	0.26	27	9.789×10^{-3}		
Total	1.69	51			

Summary Statistics

s	0.092	R^2	0.7858
$\bar{y}$	58.34	R^2_{Adj}	0.7460
CV	0.16	R^2_{Pred}	0.6681
PRESS	0.56	Precision	16.530

to navigate the design space.

5.3 Models and Discussion

5.3.1 2QD

The results of the 2QD experiments for the accuracy response is a significant model. Equations 5.5 and 5.6 are the model equations in terms of coded and actual factors, respectively.

The factor effects are summarized graphically in Figures 5.1 to 5.4.

$$\hat{y}_1 = 88.18 + 0.16A + 0.11B + 0.45C + 0.36D - 0.56D^2 - 0.22AB - 0.31AC - 0.18BC \quad (5.5)$$

$$\hat{y}_1 = 82.14 + 64p_m + 2.3p_c + 0.045n_p + 0.26n_g - 0.00555n_g^2 - 37.9p_m p_c - 0.646p_m n_p - 0.0234p_c n_p \quad (5.6)$$

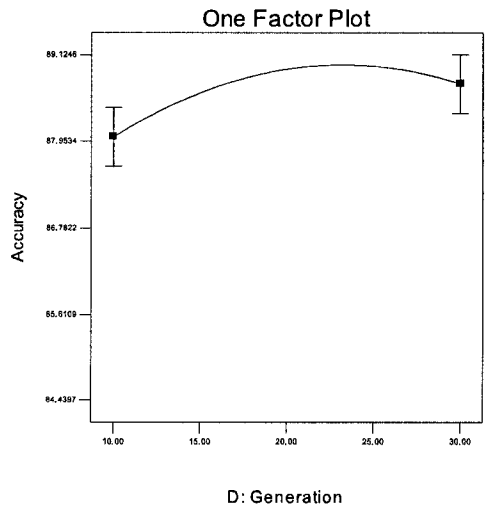
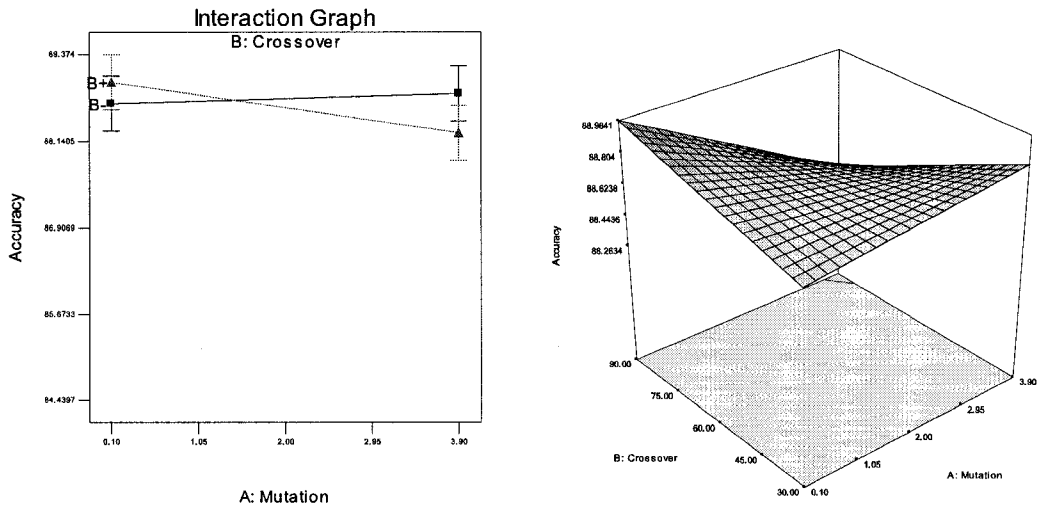


Figure 5.1: One factor plot for D for 2QD y_1 (A = 0.1, B = 90, C = 75).

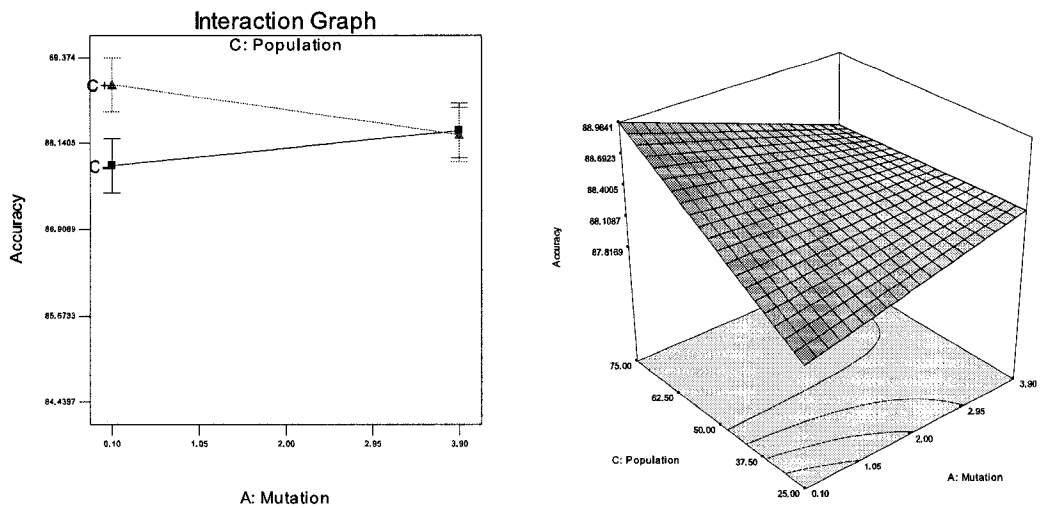
5.3.2 3QD

The results of the 3QD experiments for accuracy is a significant model. Equations 5.7 and 5.8 are the model equations in terms of coded and actual factors, respectively. The factor



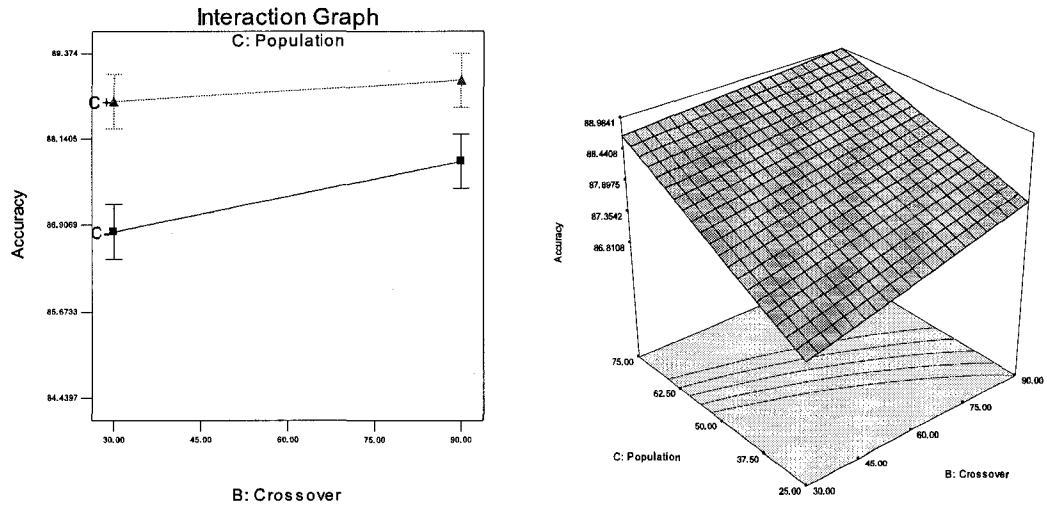
(a) (b)

Figure 5.2: (a) 2QD y_1 interaction graph for AB (C = 75, D = 23). (b) The corresponding 3D surface graph.



(a) (b)

Figure 5.3: (a) Interaction graph for AC for 2QD y_1 (B = 90, D = 23). (b) The corresponding 3D surface graph.



(a) (b)

Figure 5.4: (a) Interaction graph for BC for 2QD y_1 (A = 0.1, D = 23). (b) The corresponding 3D surface graph.

effects are summarized graphically in Figures 5.5 to 5.7.

$$\hat{y}_1 = 76.93 + 0.052A + 0.29B + 0.039C + 0.50D - 0.63A^2 + 0.46C^2 - 0.25AB - 0.36AC \quad (5.7)$$

$$\hat{y}_1 = 75.09 + 137p_m + 1.8p_c - 0.057n_p + 0.050n_g - 1700p_m^2 + 7.39 \times 10^{-4}n_p^2 - 43.8p_m p_c - 0.756p_m n_p \quad (5.8)$$

5.3.3 2SVM

The results of the 2SVM experiments for accuracy is a significant model. Equations 5.9 and 5.10 are the model equations in terms of coded and actual factors, respectively. The

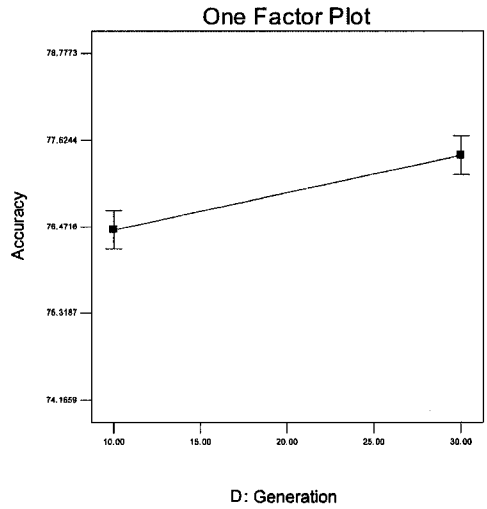
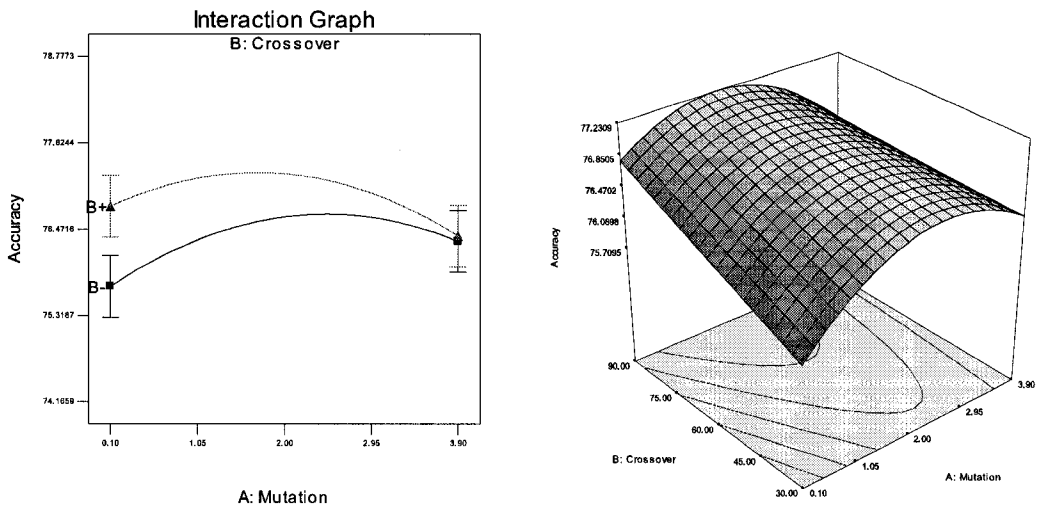
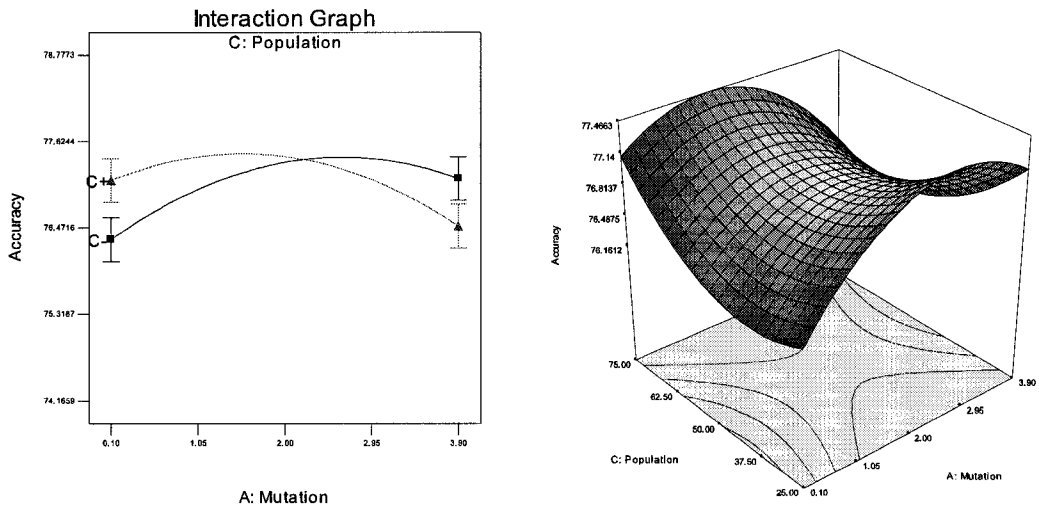


Figure 5.5: One factor plot for D for 3QD y_1 (A = 2.0, B = 60, C = 50).



(a) **(b)**

Figure 5.6: (a) 3QD y_1 interaction graph for AB (C = 50, D = 20). (b) The corresponding 3D surface graph.



(a) (b)

Figure 5.7: (a) 3QD y_1 interaction graph for AC (B = 60, D = 20). (b) The corresponding 3D surface graph.

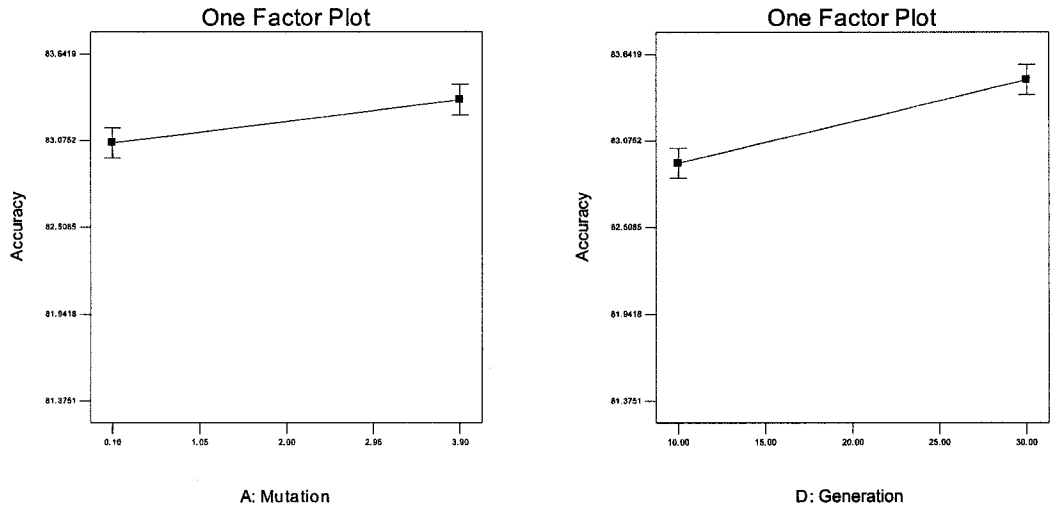
factor effects are summarized graphically in Figures 5.8 to 5.9.

$$\hat{y}_1 = 83.2 + 0.14A + 5.932 \times 10^{-3}B + 0.2C + 0.27D - 0.22B^2 + 0.12BC \quad (5.9)$$

$$\hat{y}_1 = 81.71 + 7.4p_m + 2.1p_c - 1.54 \times 10^{-3}n_p + 0.027n_g - 2.4p_c^2 + 1.57p_c n_p \quad (5.10)$$

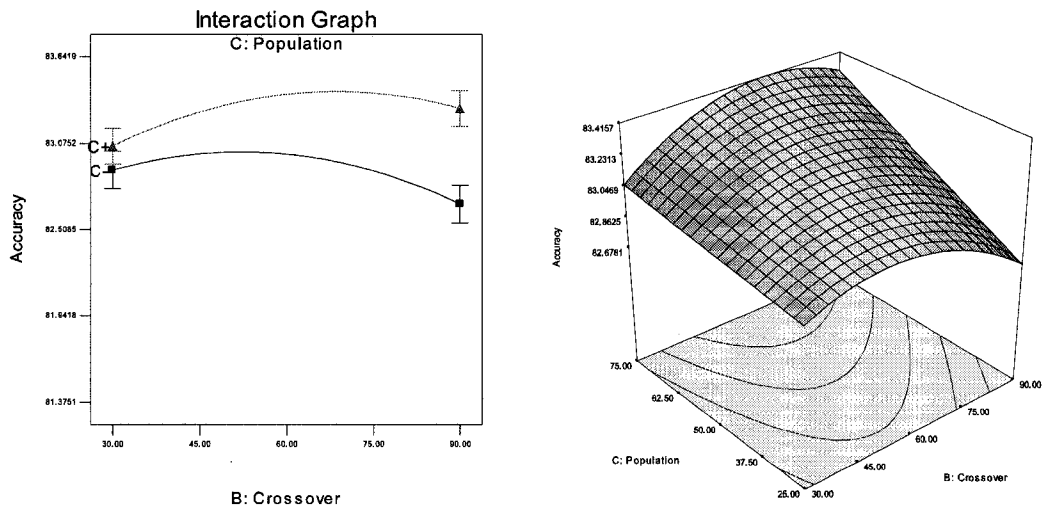
5.3.4 3SVM

The results of the 3SVM experiments for the accuracy response is a significant model. Equations 5.11 and 5.12 are the model equations in terms of coded and actual factors,



(a) (b)

Figure 5.8: (a) One factor plot for A for 2SVM y_1 (B = 60, C = 50, D = 20). (b) One factor plot for D for 2SVM y_1 (A = 2, B = 60, C = 50).



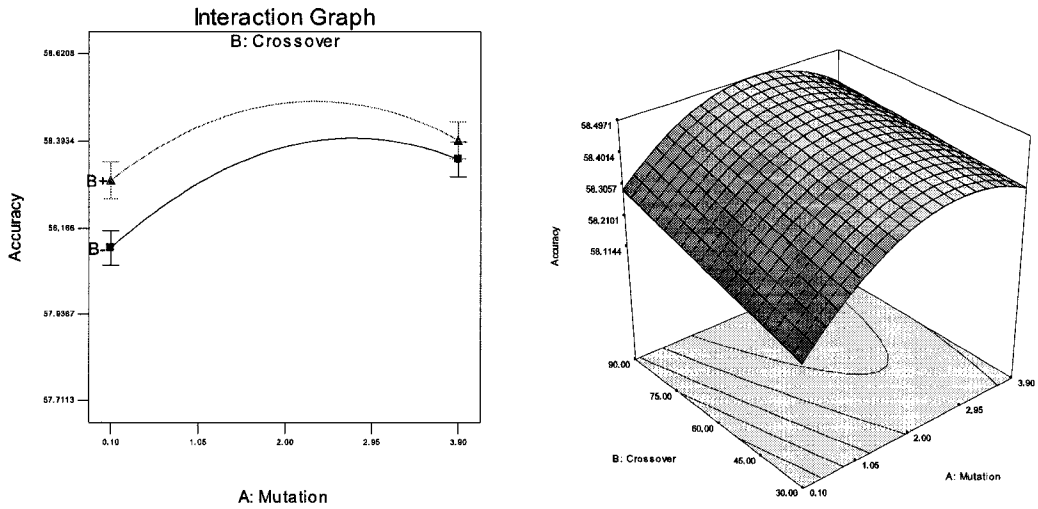
(a) (b)

Figure 5.9: (a) 2SVM y_1 interaction graph for BC (A = 2, D = 20). (b) The corresponding 3D surface graph.

respectively. The factor effects are summarized graphically in Figures 5.10 to 5.12.

$$\hat{y}_1 = 58.44 + 0.084A + 0.057B + 0.077C + 0.097D - 0.15A^2 - 0.032AB - 0.042AC + 0.058AD \tag{5.11}$$

$$\hat{y}_1 = 57.68 + 23p_m + 0.302p_c + 4.88 \times 10^{-3}n_p + 3.64 \times 10^{-3}n_g - 410p_m^2 - 5.64p_m p_c - 0.0893p_m n_p + 0.3028p_m n_g \tag{5.12}$$

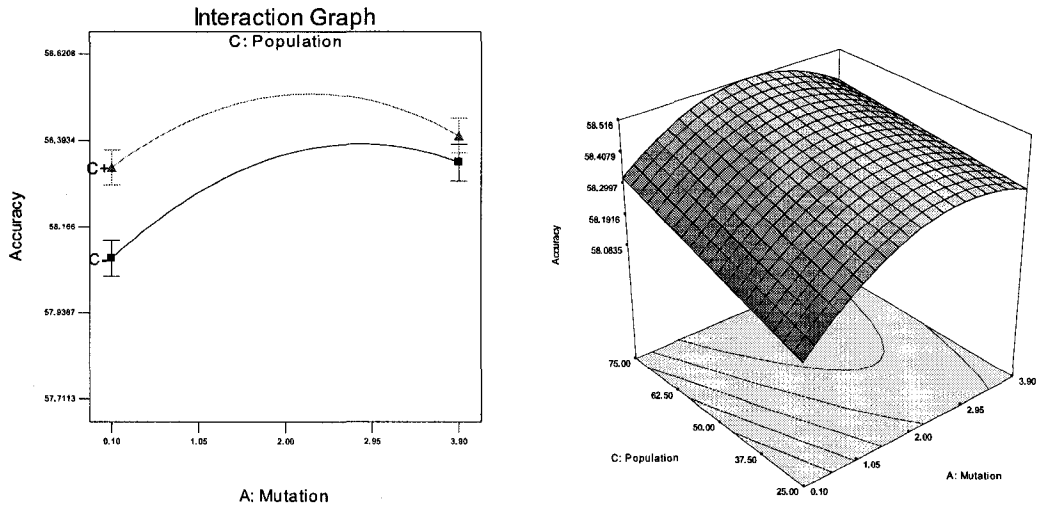


(a) (b)

Figure 5.10: (a) 3SVM y_1 interaction graph for AB (C = 50, D = 20). (b) The corresponding 3D surface graph.

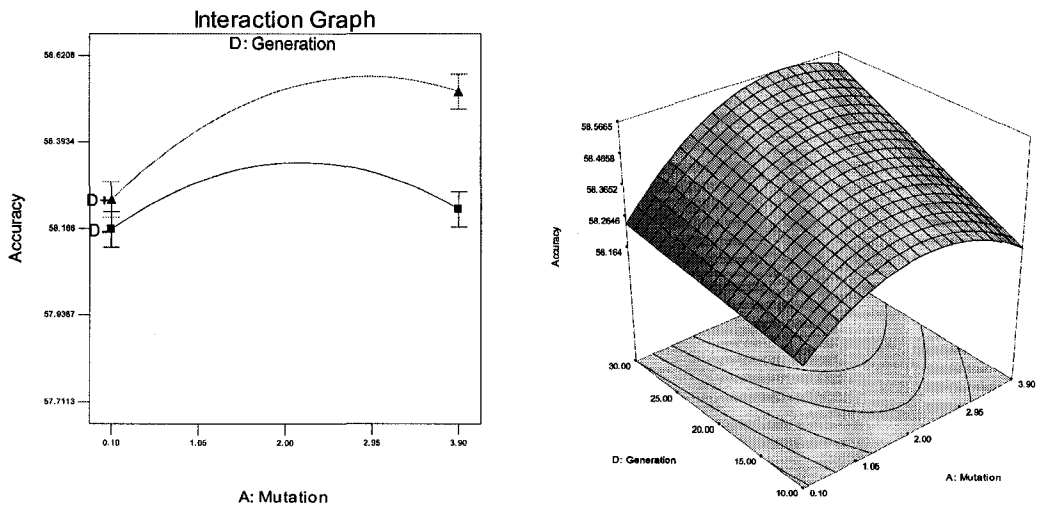
5.3.5 Summary Discussion

For both classifier types, there is only a small difference in best and worst accuracy response: 2SVM $\approx$ 1.5%, 3SVM $\approx$ 0.75%, 2QD $\approx$ 4.4%, and 3QD $\approx$ 4.5%. This might suggest that the GA is fairly robust; regardless of GA parameter settings, within the design space considered, the responses are somewhat similar. This is especially true when high generation and high population values are used. This is consistent with results reported by



(a) (b)

Figure 5.11: (a) 3SVM y_1 interaction graph for AC (B = 60, D = 20). (b) The corresponding 3D surface graph.



(a) (b)

Figure 5.12: (a) 3SVM y_1 interaction graph for AD (B = 60, C = 50). (b) The corresponding 3D surface graph.

[103]. It is hypothesized that the smaller differences in SVMs are a result of these classifiers performing well (high accuracy) even without FSS, which is what the GA is performing. As a consequence the GA cannot further optimize the SVM (provide a better accuracy) by selecting different features sets. This is consistent with the results of others (e.g. [59]) who have suggested FSS for SVMs is not necessary. The results might be much different if SVM parameters were also optimized by the GA.

A general trend in all models is an increase in accuracy with increasing population size and generation size (e.g. Figure 5.4, Figure 5.9, Figure 5.11, Figure 5.12). This can be explained in evolutionary terms: high values for population means a bigger gene pool and therefore more gene diversity to sample from; more generations allows more time to filter out the bad genes and concentrate the good genes. It is suspected that this increase in accuracy will continue with increasing population and generation, but will rapidly reach a critical point where there will be no further significant increase in accuracy regardless of population and generation size. That point may have been reached in the chosen design space for 2QD because a peak is seen at about $n_g = 23$ (Figure 5.1).

This increase and hypothesized leveling off makes sense when considering how a GA functions over time, where time is determined by population and generation size. The GA starts with random, generally poor solutions. It then evolves to progressively better solutions, first at a fast rate, then gradually slower until it converges on the best solution possible for the GA. At this point executing the GA longer will generally not change the solution. The graph of such a progress is similar to a log function. Figure 5.13 shows an example from an actual GA ran in this work. The general trend found in the models will likely follow a similar pattern. This suggests the response surface method (RSM) of steepest ascent [118, 117] can be applied to both investigate this phenomenon and to select the optimal GA parameters over a much larger space than considered in this work.

Each accuracy model is noticeably different in the single factors and interactions that are

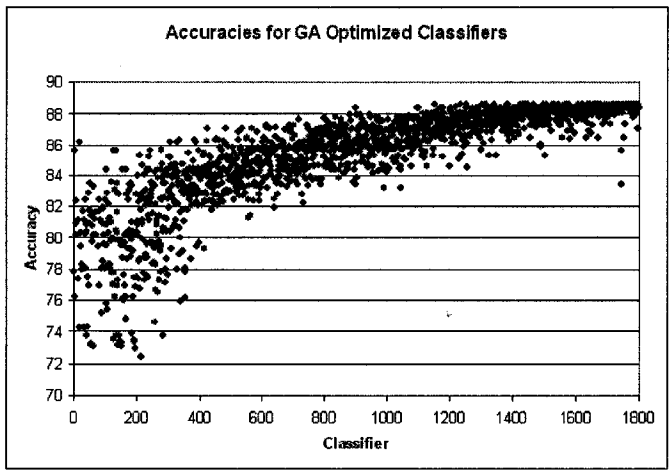


Figure 5.13: Plot of classifier accuracies for each GA iteration illustrating asymptotic behaviour over time.

significant and the coefficients attached to these. The only commonality is that all factors (A, B, C, and D) are included in the model either as a significant main effects or as part of a two factor interaction. The later suggests all factors (GA parameters) are important in some way to calibrating a GA. The former suggests a single DOE derived model for all GAs used in classifier optimization cannot be developed. Rather, the optimum GA parameter values are problem specific. Hence, a model should be developed for each new problem and the optimal parameter values selected. Factors that make one problem different from another might include classifier type, number of classes, classifier parameter settings, and possibly the data set. This is consistent with results obtained by others (e.g. [114, 121, 122]) who conducting similar studies on GAs used for optimizing job-shop schedules in the manufacturing industry (see Section 3.8).

If optimal results are expected from a GA, then blindly using De Jong's "standard" parameter settings is not recommended, especially when optimizing QD classifiers. In general, De Jong's parameters can be used as a starting point and DOE, or some similar technique, should be used to find the best parameter values for the given problem. Nevertheless, the results are consistent with De Jong's and more recently published recommen-

dations (e.g. [95]) in that a high crossover and low mutation rate should be used—consider Figures 5.2, 5.3, 5.6, 5.10, 5.11, and 5.12. This can be explained in evolutionary terms. High crossover means more mating will occur, further diversifying the gene pool, but with good genes. A high mutation provides more chances to evolve away from a local minimum. However, a mutation level set too high results in too much randomness in the evolution. As a consequence it takes the GA much longer to find a good solution, or the GA terminates before the best solution is found.

None of the accuracy models are excellent, as can be deduced from the summary statistics. This might suggest that for the classifier, data, classifier configuration, and design space considered, the accuracy response is very irregular. As a result the linear, quadratic, or 2FI models cannot be adequately fitted. If a smaller design space was used, or the design was shifted perhaps the models would fit better. In general the SVM type classifier produced better models; the best model is the 3SVM, followed by the 2SVM, with the 2QD and 3QD models tied for last. This is most likely because the response surface for this classifier GA combination is smoother. We cannot generalize and say the same result would be produced with a new data set or classifier configuration.

5.4 Calculating the Optimal GA Parameters

The models (equations) derived in the proceeding sections can be used to find the optimal GA parameter values for each classifier type. There are several common methods for finding the optimal [117, 118]: (1) visually examine the graphs in Section 5.3 and estimate the best values; (2) overlay contour plots of each response surface; (3) use the downhill simplex method in conjunction with a desirability function; and (4) write the second order model in matrix notation, take the derivate, set it to zero, and solve for all the factors. However, each of these methods have their own set of issues limiting their use. They are either not precise,

difficult to apply and achieve good results, or can have computation issues that may require advanced techniques to solve.

A simple and effect alternative to the aforementioned methods is to repeatedly evaluate the accuracy functions in a grid search over the model design space and pick the parameter values that correspond to the function that gives the highest accuracy. This method can be employed for two reasons. Firstly, only one response is considered, so a desirability function approach—well suited when multiple responses are considered—is unnecessary. Secondly, the model space is relatively small and the model functions are computationally cheap to evaluate, so evaluating them many times is not an issue.

In fact, a near exhaustive search can be performed. First, two of the four factors (C and D) are discrete (integers) with a (small) finite range. Second, the remaining factors (A and B), which are real, can be easily discretized such that the change in accuracy between one discrete value and the adjacent value is not significantly different. Hence, from a practical point of view, an exhaustive search is performed resulting in equal or better results in a reasonable amount of time compared to other commonly used search methods, such as the downhill simplex method.

Using this method, a grid search was performed by evaluating the model function for each point in the finite set of points $\mathbb{A} \times \mathbb{B} \times \mathbb{C} \times \mathbb{D}$ and recording the accuracy response, where $\mathbb{A}$ is the set of p_m, $\mathbb{B}$ is the set of p_c, $\mathbb{C}$ is the set of n_p, and $\mathbb{D}$ is the set of n_g:

$$\begin{aligned} \mathbb{A} &= \{0.0010, 0.0015, \ldots 0.0390\} \\ \mathbb{B} &= \{0.300, 0.305, \ldots 0.900\} \\ \mathbb{C} &= \{25, 26, \ldots 75\} \\ \mathbb{D} &= \{10, 11, \ldots 30\} \end{aligned} \tag{5.13}$$

$\mathbb{A} \times \mathbb{B} \times \mathbb{C} \times \mathbb{D}$ systematically covers the entire model space in a 4-D grid, resulting in

Table 5.10: Summary of DOE derived optimal GA parameter values for each classifier type.

Classifier Type	p_m	p_c	n_p	n_g
2QD	0.0010	0.900	75	23
3QD	0.0115	0.900	75	30
2SVM	0.0390	0.685	75	30
3SVM	0.0245	0.900	75	30

a near exhaustive search. The optimal set of GA parameter values correspond to the point $(a, b, c, d) \in \mathbb{A} \times \mathbb{B} \times \mathbb{C} \times \mathbb{D}$ that results in the maximum accuracy. The optimal parameter values for each model are summarized in Table 5.10.

Chapter 6

Beluga Whale Classification

This chapter describes the development of several classifiers and their evaluation and optimization using DOE calibrated GAs. The classifiers are used to classify the objects found by image processing algorithms described in Chapter 4. This is steps 3 and 4 in the basic approach described in Section 3.1. The chapter follows the broad steps in developing any classifier. First the feature selection process, feature extraction process, and identification of object classes are described. This is followed by a description of classifier designs, training and testing, and classifier optimization methods employed. The chapter ends with a discussion of the results.

6.1 Feature Selection

Selecting a set of features to distinguish objects is a critical step in developing a classifier [30]. First, an initial set of features must be chosen, and then that set analyzed to find the best subset combination that improves classification. Rather than doing extensive analysis of many different features in the hopes of finding one or more that renders the feature subset selection and classification problem easier, the focus is onenabling classification in the presence of weakly discriminating features. This approach is reasonable given the chal-

lenging conditionspresented (i.e. limited image resolution, low visual distinction between whales and wave crests, etc.). One goal of this work is to use a genetic algorithm (GA) as the single means of feature subset selection; that is, optimize the classifier by automatically selecting the best feature subset with minimal upfront analysis of the initial set. Hence, minimal analysis of the original feature set was conducted. To do otherwise would make the GA approach less automatic then desired.

Nevertheless, the initial set of features was not chosen completely arbitrarily. The features that can be derived intuitively (without major research) are limited since the subject is natural objects (which have a great variation in size, shape, etc.) that are represented by a very few number of pixels. As a result the initial set was chosen from a common set of features often used as the starting point for analysis (i.e. feature reduction). The criteria used to select features are: (1) the feature must be easy and efficient to calculate, (2) the feature must describe some aspect of a whale, and (3) the feature must be unbiased. Including such features in the classifier will make it biased to the training data, hence such features were avoided. The twenty-five features calculated for each object (blob, or target) are shown in Table 6.1 (see also Appendix A). The numbers in brackets are reference numbers used in the remainder of this chapter. Note that some of these features may be redundant or noisy, but if the GA can be used for feature reduction, then it should be able to filter out such features.

6.2 Feature Vector Extraction

Algorithms for extracting the 25 features from the segmented objects were developed and tested. Then, images with representative whale and non-whale objects (such as wave crests and sun glare) were manually selected and subsequently processed by the image processing algorithms described in Chapter 4 to produce labeled binary images of objects. The binary

Table 6.1: The 25 features used in classification.

(1) Area	(14) Roughness
(2) Length	(15) Mean Pixel
(3) Breadth	(16) Pixel Standard Deviation
(4) Elongation	(17) Minimum Pixel
(5) Minimum Ferret Diameter	(18) Maximum Pixel
(6) Maximum Ferret Diameter	(19) Moment Invariant 1
(7) Ferret Elongation	(20) Moment Invariant 2
(8) Mean Ferret Diameter	(21) Moment Invariant 3
(9) Number of Holes	(22) Moment Invariant 4
(10) Number of Chain Pixels	(23) Moment Invariant 5
(11) Perimeter	(24) Moment Invariant 6
(12) Convex Perimeter	(25) Moment Invariant 7
(13) Compactness	

images, together with their corresponding color and gray scale images, were then analyzed by the feature extraction algorithms to produce a feature vector for each object found. A total of 1304 whale and not-whale objects were captured. These are divided into classes as described in the Section 6.3.

6.3 Object Classes

The feature vectors for 1304 objects derived in Section 6.2 were summarized in a spreed sheet and analyzed using MATLAB [28] and Microsoft Excel [132] math and graphing functions. The main purpose was to derive logical object classes and to determine if the calculated features values were reasonable. From this reconnaissance analysis and some classifier experiments it was determined that two methods should be considered: (1) the data grouped into three classes and classified using a 3-class classifier; and (2) the data grouped into two classes and classified using a 2-class classifier.

In the first case the following three classes where created:

1. *certain whales*; hereafter indicated by the subscript c, as in $\mathscr{C}_c$

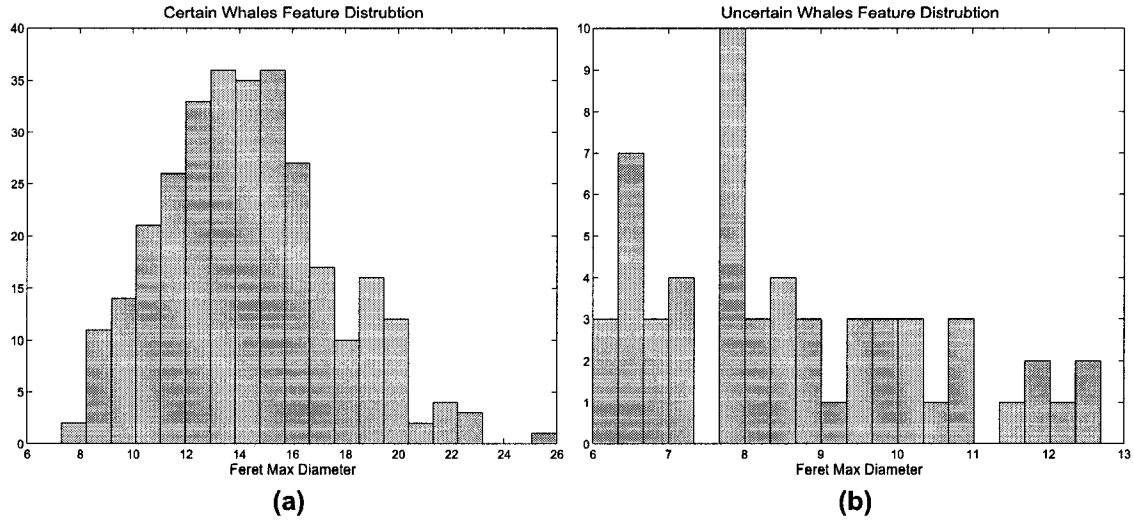


Figure 6.1: Example of normal and non-normal feature value distribution. The left histogram shows a normal distribution of a feature that describes the certain whale class well. The right histogram shows the poor normal distribution for the same feature in the poorly defined uncertain whale class.

2. *uncertain whales*; hereafter indicated by the subscript u, as in $\mathscr{C}_u$

3. *not-whales*; hereafter indicated by the subscript n, as in $\mathscr{C}_n$

Certain whales are whales that have been labeled as certain in the ground truth data by scientific experts. They are good representatives of whales because they have all or most of the features common to clearly identifiable whales. In particular, they are located near the surface, white in color, medium to large size, and elongated. In almost all cases, these are adults. Uncertain whales are whales that have been labeled uncertain in the ground truth data, mainly because they lack characteristics of typical whales. These are typically small in size, located at great depth beneath the surface, light blue-green color, and not elongated. Most young whales are assigned to this class, although by no means does this class consists solely of young. The third, class not-whales, consists of all objects that do not fall into the aforementioned classes.

It was hypothesized that these three classes would best encapsulate the natural grouping of the objects (based on clustered feature values), encapsulate what was observed from

Table 6.2: Summary of the data set partitioned into classes and groups.

Grouping	Class	No. of Samples
3-class	$\mathscr{C}_c$	306
	$\mathscr{C}_u$	57
	$\mathscr{C}_n$	941
2-class	$\mathscr{C}_w$	363
	$\mathscr{C}_n$	941
total samples		1304

manual examination of images, and conform to the readers manual classification. However, it turns out that the uncertain class is somewhat poorly defined: many (9 out of 25) of its features have poor normal distributions and the feature value distributions overlap significantly with other classes (e.g. Figure 6.1). This is likely because the sample size for this class is small (Table 6.2) and whales are naturally occurring objects, and as such have a huge variation of feature values as their patterns change between different animals and different environmental conditions (Section 4.2).

As a result of the proceeding discussion, it was deemed necessary to consider a 2-class case consisting of the following classes:

1. *whales*; hereafter indicated by the subscript w, as in $\mathscr{C}_w$

2. *not-whales*; hereafter indicated by the subscript n, as in $\mathscr{C}_n$

The whale class is formed by combining certain and uncertain classes described above. The not-whale class is the same as in the 3-class case. Table 6.2 shows how the 1304 data objects are grouped for each class arrangement.

6.4 Classifier Optimizing GA Implementation

In this work, classifier optimization refers to feature subset selection and optimizing parameter values that increases overall accuracy. The GA designed for classifier optimization follows the basic GA given in [95]. This GA, with modifications as necessary, has been successfully applied to many similar problems. This section describes the basic design common to all classifiers considered. The general classifier optimization algorithm is listed in Algorithm 1, below.

Algorithm 1: GA to optimize classifiers

Result: List of chromosomes with corresponding classification accuracy

```
1  begin
2      create initial population, P_0, of n_p randomly generated chromosomes
3      evaluate fitness of each chromosome in P_0 using f(c)
4      for g = 1 to n_g do
5          set C to n_p/2 pairs of chromosomes selected from P_{g-1} based on fitness
6          create P_g by crossing over chromosome pairs in C using p_c
7          mutate each chromosome in P_g using p_m
8          evaluate fitness of each chromosome in P_g using f(c)
9      end
10 end
```

The first step in applying the GA is to select good parameter values for n_p, p_m and p_c. This is detailed in Chapter 5. In particular, see Table 5.10, where the specific parameter values for the four classifier types are listed.

In this GA, a chromosome consisting of a string of 0s and 1s encode the selected features for a classifier instance generated in the GA. For certain SVM classifiers, the SVM parameter values are also encoded. This is detailed more throughly in Section 6.5. The features are enumerated as given in Table 6.1 and, from left to right, are labeled 1 if the feature is included in the classifier instance and 0 if not. For example, the following is a chromosome with features 7 and 9 omitted from the classifier: 111111010111111111111111.

Generation g_i, $i = 1, \cdots, n_g$, of n_p chromosomes is created from g_{i-1} by selecting $n_p/2$ pairs (c_i, c_j) of g_{i-1} parent chromosomes and mating these to produce two offspring

(c'_i, c'_j) —the initial generation, g_0, of n_p chromosomes is created using a uniform random sequence of 0s and 1s. The parent chromosomes selected depends on fitness levels; the higher the chromosome fitness, the higher the probability it will be selected to mate. To achieve this the roulette wheel method of selection is used (Section F.2). The mating of (c_i, c_j) is accomplished using single point crossover (Section F.3), where the crossover point is uniform randomly selected. Whether (c_i, c_j) mates or simply migrates to the next generation unchanged is governed by the crossover rate, p_c. Generation g_i of chromosomes then enters a mutation stage. Here, each bit of each chromosome has the probability p_m of being mutated (inverted) (Section F.4). Finally, the chromosomes are evaluated. For each chromosome in g_i, a classifier is built using the feature subset and classifier parameters encoded in the chromosome. The generated classifier is then trained and tested using 10-fold cross validation (Section 6.6). The testing results are evaluated using the fitness function $f(c)$ that produces a fitness number f associated with the chromosome c. This process repeats for n_g generations.

In this research the main concern is overall classifier accuracy. In most published works with a similar concern (e.g. [108, 110, 112]), $f(c)$ is usually equal to the accuracy. However, using accuracy as a measure of classifier performance on unbalanced (biased sample) data is potentially misleading since a high accuracy can be achieved by classifying more objects to the class with the most samples. Since the data in this work is significantly unbalanced, the following method is used to calculate a "balanced accuracy" (or "sample normalized hit rate") and use it as the fitness function:

$$f = \frac{\sum_{i=1}^{n_{\mathscr{C}}} h_i}{n_{\mathscr{C}}} \tag{6.1}$$

where $n_{\mathscr{C}}$ is the number of classes and h_i is the classification (hit) rate for $\mathscr{C}_i$.

6.5 Classifier Design

It is important to find the best classifier possible for the problem at hand, but all possible classifiers and their configurations cannot be examined. In this work an attempt is made to find the best classifier configuration from 2 and 3 class Bayesian minimum error rate quadratic discriminates (Section 3.2) and 2 and 3 class support vector machines (SVMs) (Section 3.3). This section describes these classifiers and their configurations as they are implemented in this work—theory is not covered.

Some classifier naming conventions need to be defined up front. Names consisting of all capital letters refer to the classifier type. Mixed lower case and upper case names refer to particular classifier configurations. A proceeding number on either indicates the number of classes being discriminated by the classifier. Note that classifier types are groups of similar classifiers, hence these cannot be optimized; only configured classifiers are optimized. Hence, for example, 2QD refers to the 2-class quadratic discriminate type classifiers, 3SVM refers to the 3-class SVM type classifiers, and 3Svm refers to a 3-class SVM classifier configured using default values and optimized using a GA. Table 6.3 summarizes all the classifier configurations used. Details are explained in the following sections.

6.5.1 2-Class Quadratic Discriminate (2QD)

Recall (Section 3.2) that the definition for a Bayesian minimum error rate quadratic discriminate function for $\mathscr{C}_i$ is

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) - \frac{d_i}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(\mathscr{C}_i) \tag{6.2}$$

In the 2QD case, $i \in \{w, n\}$ (Section 6.3). Using the samples for $\mathscr{C}_i$ the mean feature vector $\boldsymbol{\mu}_i$, the covariance matrix $\boldsymbol{\Sigma}_i$, and its determinate $|\boldsymbol{\Sigma}_i|$, is calculated. The term $(d_i/2)\ln(2\pi)$ is a constant because the dimensionality of all classes are the same: $d = 25$. For discrim-

Table 6.3: Summary of classifier configurations. * represent values to be optimized by the GA. Other values are precalculated as described in subsections of Section 6.5.

Classifier	Classifier Parameters				
	$P(\mathscr{C}_w)$	$P(\mathscr{C}_n)$			
2Qd	0.28	0.72			
	$P(\mathscr{C}_c)$	$P(\mathscr{C}_u)$	$P(\mathscr{C}_n)$		
3Qd	0.23	0.044	0.72		
	C	γ	w_w	w_n	
2Svm	1	$1/d$	1	1	
2SvmCg	2048	0.03125	1	1	
2SvmCgWi	8192	0.03125	2.6	1	
2SvmGaCg	*	*	1	1	
2SvmGaCgWi	*	*	2.6	1	
2SvmGaCgGaWi	*	*	*	*	
	C	γ	w_c	w_u	w_n
3Svm	1	$1/d$	1	1	1
3SvmCg	32768	0.007813	1	1	1
3SvmCgWi	2	8	3.08	16.6	1
3SvmGaCg	*	*	1	1	1
3SvmGaCgWi	*	*	3.08	16.6	1
3SvmGaCgGaWi	*	*	*	*	*

ination between classes we are interested only in the relative size of $g_i(\mathbf{x})$, not the actual values; hence, the additive constant is superfluous. $P(\mathscr{C}_i)$ is the prior probability of class i. Since the true, real world prior probability of each class is unknown, it was estimated as

$$P(\mathscr{C}_w) = n_w/n = 363/1304 = 0.28 \tag{6.3}$$

$$P(\mathscr{C}_n) = n_n/n = 941/1304 = 0.72 \tag{6.4}$$

where n_i is the total samples in the data set belonging to $\mathscr{C}_i$ and n is the total number of samples in the data set. However, using this method to determine $P(\mathscr{C}_i)$ assumes that new unknown data sets will be obtained under similar conditions (e.g. location, weather conditions, altitude, and time of year). Hence the final discriminate functions are

$$g_w(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_w)^T\boldsymbol{\Sigma}_w^{-1}(\mathbf{x}-\boldsymbol{\mu}_w) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_w| + \ln P(0.28) \tag{6.5}$$

$$g_n(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_n)^T\boldsymbol{\Sigma}_n^{-1}(\mathbf{x}-\boldsymbol{\mu}_n) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_n| + \ln P(0.72) \tag{6.6}$$

Using the above equations, an object whose feature vector is $\mathbf{x}$ is classified as $\mathscr{C}_n$ if $g_n > g_w$, otherwise it's classified as $\mathscr{C}_w$. A classifier configured using the above description is known as the 2Qd classifier.

6.5.2 3-Class Quadratic Discriminate (3QD)

The 3QD classifier type is similar to the 2QD classifier type in the previous section except there are three classes: $\mathscr{C}_c$, $\mathscr{C}_u$, and $\mathscr{C}_n$. The prior probabilities are

$$P(\mathscr{C}_c) = n_c/n = 363/1304 = 0.23 \tag{6.7}$$

$$P(\mathscr{C}_u) = n_u/n = 57/1304 = 0.04 \tag{6.8}$$

$$P(\mathscr{C}_n) = n_n/n = 941/1304 = 0.72 \tag{6.9}$$

The discriminate functions are

$$g_c(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)^T\boldsymbol{\Sigma}_c^{-1}(\mathbf{x}-\boldsymbol{\mu}_c) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_c| + \ln P(0.23) \tag{6.10}$$

$$g_u(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_u)^T\boldsymbol{\Sigma}_u^{-1}(\mathbf{x}-\boldsymbol{\mu}_u) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_u| + \ln P(0.04) \tag{6.11}$$

$$g_n(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_n)^T\boldsymbol{\Sigma}_n^{-1}(\mathbf{x}-\boldsymbol{\mu}_n) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_n| + \ln P(0.72) \tag{6.12}$$

Using these equations, an object whose feature vector is $\mathbf{x}$ is classified as $\mathscr{C}_n$ if $\max(g_u, g_c, g_n) = g_n$, $\mathscr{C}_u$ if $\max(g_u, g_c, g_n) = g_u$, and $\mathscr{C}_n$ if $\max(g_u, g_c, g_n) = g_n$. If $g_i = g_j$ in $\max(g_i, g_j)$, g_i is taken as the maximum value. A classifier configured using the above description is known as the 3Qd classifier.

6.5.3 2 Class SVM (2SVM)

The formulation chosen for the SVM classifiers is the classical C-support vector classification (C-SVC) [59], or more generally C-SVM. The decision function is (Section 3.3.1)

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} y_i\alpha_i K(\mathbf{x}, \mathbf{x}_i) + b\right) \tag{6.13}$$

The kernel chosen, for reasons discussed in Section 3.3, is the RBF kernel:

$$K(\mathbf{x}, \mathbf{x}_i) = e^{-\gamma\|\mathbf{x}-\mathbf{x}_i\|^2} \tag{6.14}$$

As in the 2QD classifier, the two classes are whale and not-whale. These are labeled $+1$ and -1 respectively, as is required for SVMs. Thus, for an object with feature vector $\mathbf{x}$, it is classified as a whale if $g(\mathbf{x}) = +1$ and a not-whale if $g(\mathbf{x}) = -1$.

To optimize 2SVMs it is usually not sufficient to rely on FSS only. Using the RBF kernel means the parameters C and γ must be chosen *a priori*. This can be done by simply

setting them to default values, using the SVM parameter grid search technique, or, as proposed in this work, using a GA. Additionally, the training data for the whale and not-whale classes are unbalanced, which can result in the classifier favouring the class with more training samples. To optimize the best classifier, several 2SVM configurations are considered (Table 6.3); these are detailed in the following subsections.

Before training the proposed classifiers the training data was scaled. Scaling is necessary for two main reasons [65]. First, it prevents features with a large range of values dominating those with small ranges. Second, it avoids potential numerical difficulties in calculating the inner product of feature vectors. To achieve this, each feature value for feature i in the training data was linearly scaled to the range $[-1, 1]$. Before testing, the scaling factor calculated for feature i in the training data is applied to each value for feature i in the testing data.

6.5.3.1 2Svm

First, and the most basic 2-class SVM, is the 2Svm classifier. This classifier uses default values for C and γ: $C = 1, \gamma = 1/d$, where $d = 25$. Also, no attempt to modify the classifier for unbalanced data is made.

6.5.3.2 2SvmCg

The second configuration, 2SvmCg, uses the SVM parameter grid search technique [65] to select C and γ values (Section 3.3.3). In this method a $\mathbb{X} \times \mathbb{Y}$ grid is setup, where $\mathbb{X} = [-5, 15] \subset \mathbb{Z}$ and $\mathbb{Y} = [-11, 2] \subset \mathbb{Z}$. At each point $(x, y) \in \mathbb{X} \times \mathbb{Y}$ a SVM is configured with $C = 2^x$ and $\gamma = 2^y$. The SVM is trained and tested using 10-fold cross validation on the entire data set with all features included. C and γ at the point with the highest accuracy is chosen for the final model. The results are shown graphically in Figure 6.2a and tabulated in Table 6.3.

6.5.3.3 2SvmCgWi

The third configuration, 2SvmCgWi, takes into account the unbalanced training data by extending the classical C-SVC to include C_- and C_+ for classes -1 and $+1$, respectively. C_- and C_+ are implemented using the notion of weights. In this method weight multipliers, w_- and w_+, are used to calculate C_- and C_+, respectively $C_- = Cw_-$ and $C_+ = Cw_+$. For the 2SVMs, the weights for the whale and not-whale classes ($\mathscr{C}_w$, and $\mathscr{C}_n$) are designated as w_w and w_n, respectively.

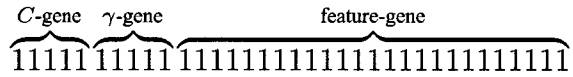
No results are published on how to best chose weights for each class. In this work, when w_i is precalculated for $\mathscr{C}_i$, it is set inversely proportional to the *a priori* probability of class occurrence, normalized so the largest class has a weight of 1. The probability of $\mathscr{C}_i$ is estimated from the fraction of $\mathscr{C}_i$ samples to the total samples. Hence, w_i is calculated as

$$w_i = \frac{\left(\frac{n}{n_i}\right)}{\left(\frac{n}{n_{\max}}\right)} = \frac{n_{max}}{n_i} \tag{6.15}$$

where n is the total number of samples in the data set, n_i is the number of samples in $\mathscr{C}_i$, and $n_{\max}$ is the number of samples in the class with the largest number of samples. Thus, using the values from Table 6.2, the two classes whale and not-whale have weights $w_w = 2.6$ and $w_n = 1$, respectively. C and γ are chosen using the grid search method described in Section 6.5.3.2, but C_- and C_+ are used when the SVM is trained. The results are shown in Figure 6.2b and Table 6.3.

6.5.3.4 2SvmGaCg

The 2SvmGaCg configuration is similar to 2SvmCg, except the GA is used to optimize C, γ, and the feature set. To do so, the GA chromosome is divided into three genes: C-gene, γ-gene, and the feature-gene. C and γ are encoded into the chromosome as 5-bit genes and the feature-gene remains unchanged at 25-bits:



So a more fair comparison can be made with C and γ calculated from the SVM parameter grid search, the range of C and γ and the method of calculation in the GA is similar to the grid search. As recommended by [65, 67], $C = 2^c$ and $\gamma = 2^g$, where $c \in \mathbb{C} = [-5, 15] \subset \mathbb{Z}$ and $g \in \mathbb{G} = [-15, 3] \subset \mathbb{Z}$. However, the exponents c and g are encoded into the C-gene and γ-gene, respectively. A l-bit gene results in 2^l discrete values. These are taken to represent integers in the range $[0, 2^l - 1]$, using standard binary to integer conversion. The integer converted values from C and γ genes, call them x and y, are then scaled to ranges of $\mathbb{C}$ and $\mathbb{G}$ as follows

$$c = \frac{x}{2^l - 1}(\max(\mathbb{C}) - \min(\mathbb{C})) + \min(\mathbb{C}) \tag{6.16}$$

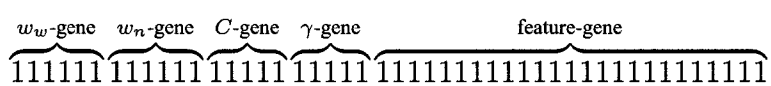
$$g = \frac{y}{2^l - 1}(\max(\mathbb{G}) - \min(\mathbb{G})) + \min(\mathbb{G}) \tag{6.17}$$

6.5.3.5 2SvmGaCgWi

The 2SvmGaCgWi classifier is configured in the same manner as 2SvmGaCg (C and γ optimized by the GA), except the weights w_w and w_n are precalculated and used as detailed for 2SvmCgWi.

6.5.3.6 2SvmGaCgGaWi

For 2SvmGaCgGaWi, the SVM parameters, the weights, and the feature set are optimized using the GA. The methods used are similar to that described for 2SvmGaCg, but in this case the chromosome is divided into five genes: C-gene, γ-gene, w_w-gene, w_n-gene, and the feature-gene. C, γ, and the feature-gene are encoded and decoded as done for 2SvmGaCg. w_u and w_n are encoded as 6-bit genes. The result is



The l-bit weight gene gives 2^l discrete values. Using binary to integer conversion, these values fall in the range $[0, 2^l - 1]$. For the w_i-gene, $i \in \{w, n\}$, the integer x derived from the gene is scaled to fall in the range $\mathbb{W} = [0.1, 5] \subset \mathbb{R}$ and used as the weight for w_i. The equation is

$$w_i = \frac{x}{2^l - 1}(\max(\mathbb{W}) - \min(\mathbb{W})) + \min(\mathbb{W}) \tag{6.18}$$

6.5.4 3 Class SVM (3SVM)

To implement the 3-class SVM type classifiers, the above 2SVM methods are extended to three classes: certain whale ($\mathscr{C}_c$), uncertain whale ($\mathscr{C}_u$), and not-whale ($\mathscr{C}_n$). To extend the C-SVM to 3 classes, the one-against-one approach is used (Section 3.3.2). For k classes, $k(k-1)/2$ binary SVMs (hyperplanes) are created such that there is one SVM trained (hyperplane constructed) on data from classes $\mathscr{C}_i$ and $\mathscr{C}_j$ for each possible combinatorial pairs (i, j), $i \neq j$, $i, j < k$. For each pair of classes, a binary SVM optimization problem is solved [63]. Once the classifiers are created, the max-wins strategy [56] is used for final assignment of $\mathbf{x}$ to a class. The basic algorithm is: if $g_n(\mathbf{x})$, $n = 1 \cdots k(k-1)/2$, classifies $\mathbf{x}$ into $\mathscr{C}_i$, the vote for $\mathscr{C}_i$ is incremented by 1, otherwise $\mathscr{C}_j$ is increased by 1. Finally, $\mathbf{x}$ is assigned to the class which has the most votes. In case of a tie, $\mathbf{x}$ is assigned to $\mathscr{C}_1$ for $\mathscr{C}_1, \mathscr{C}_2, \cdots \mathscr{C}_k$ being classified; $\mathscr{C}_1 = \mathscr{C}_u$ in this work.

6.5.4.1 3Svm

The 3Svm classifier is configured in the same manner as the 2Svm, but extended to 3 classes, as described in Section 6.5.4.

6.5.4.2 3SvmCg

This classifier is similar to the 2SvmCg classifier: C and γ are selected using the SVM parameter grid search, but with each SVM configured for three classes. The results are

shown in Figure 6.2c and Table 6.3.

6.5.4.3 3SvmCgWi

As in with 2SVM classifiers, the 3SVM classifiers have unbalanced training data. To test the effect of this, a 3-class configuration, 3SvmCgWi, using C_- and C_+ was also developed. This is an extension of 2SvmCgWi. In this method, a single C is chosen and three weights for each of the three classes is chosen and multiplied against C, forming C_i, $i \in \{c, u, n\}$. The weights are calculated using Equation 6.15, resulting in the weights $w_c = 3.08$, $w_u = 16.6$, and $w_n = 1$. Hence, as calculated in Section 6.5.3.3, $C_c = Cw_c$, $C_u = Cw_u$, and $C_n = Cw_n = C$, where C is chosen using the SVM parameter grid search (Figure 6.2d and Table 6.3). Using the one-against-one multiclass approach a total of three 2-class classifiers are trained with parameters $(C_-, C_+) = (C_c, C_u), (C_c, C_n), (C_u, C_n)$.

6.5.4.4 3SvmGaCg

3SvmGaCg is configured and implemented in the same manner as 2SvmGaCg, except it is extended to three classes.

6.5.4.5 3SvmGaCgWi

The 3SvmGaCgWi classifier is configured in the same manner as 3SvmGaCg, except the weights w_c, w_u, and w_n are precalculated and used as detailed for the 3SvmCgWi classifier.

6.5.4.6 3SvmGaCgGaWi

The 3SvmGaCgGaWi classifier is configured and implemented in the same manner as 2SvmGaCgGaWi, except it has been extended for three classes. There are now three weight genes: w_c-gene, w_u-gene, and w_n-gene, one for each of the three classes. These three genes

are encoded in the same manner as 2SvmGaCgGaWi, forming a chromosome with the following structure:

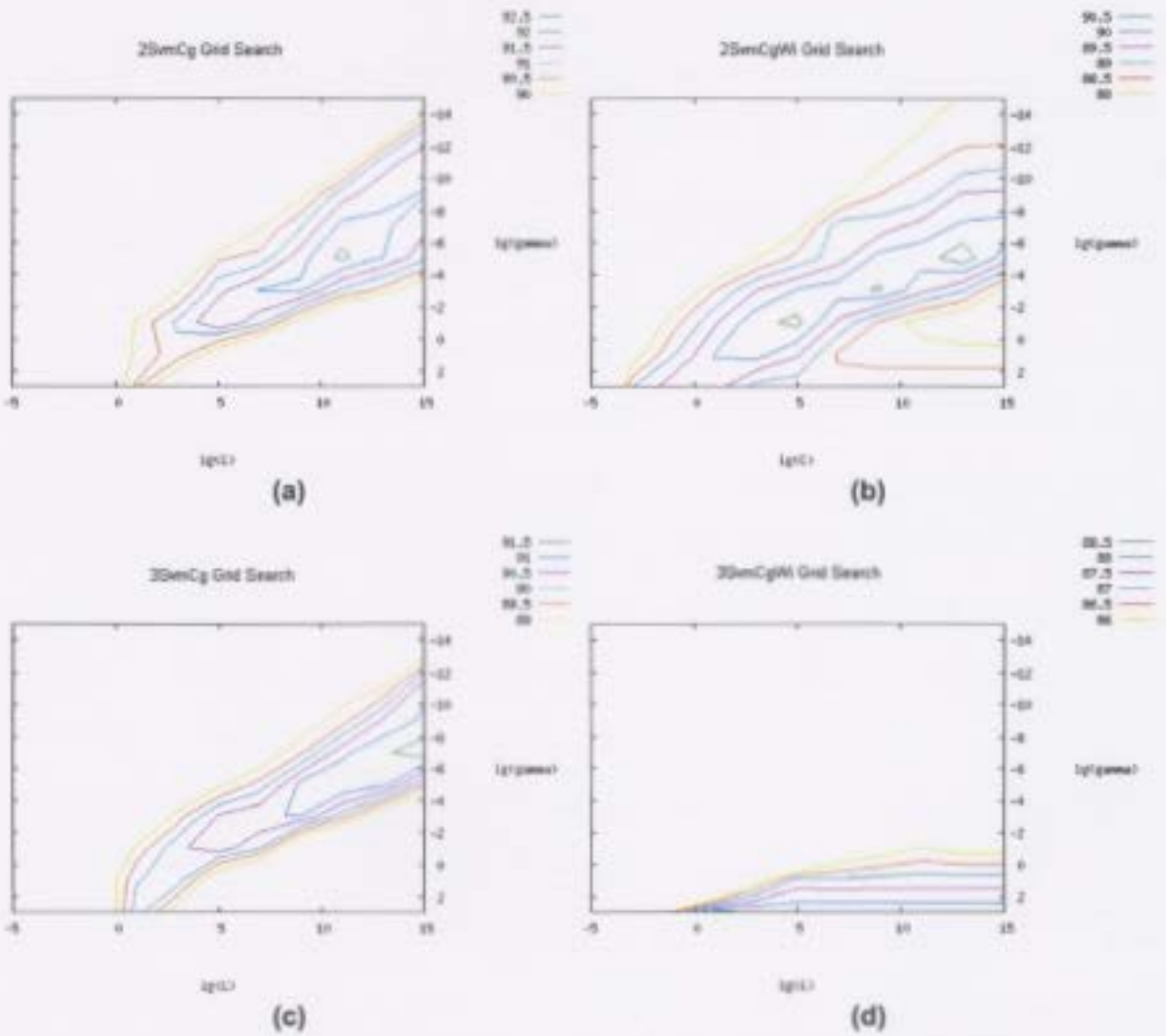


Figure 6.2: C and γ grid searches for 2 and 3-class SVMs. The estimated optimal C and γ are (a) $C = 2048$ and $\gamma = 0.03125$, (b) $C = 8192$ and $\gamma = 0.03125$, (c) $C = 32768$ and $\gamma = 0.0078125$, and (d) $C = 2$ and $\gamma = 8$.

2SVM		LOO		k-fold	
C	γ	1 eval.	GA-SVM	1 eval.	GA-SVM
1	$1/d = 1/25$	302	28,296	0.232	5,148
2048	0.03125	1858	174,240	1.42	32,040

Table 6.4: Example of timing results using LOO and k-fold cross validation evaluation methods. The 2SVM column shows the 2-class SVM configurations. The LOO column shows evaluation times when using the leave-one-out method on a single classifier instance (1 eval.) and when using a GA to optimize the SVM (GA-SVM). The k-fold columns shows evaluation times when using k-fold cross validation with $k = 10$. All times are in seconds. The GA has $n_g = 30$ and $n_p = 75$, as is commonly used in this work.

6.6 Training and Testing Methodology

For classifiers that have a slow training (or testing) time, it is important to use an evaluation method that is relatively fast. It is equally important for the chosen method to accurately estimate the true performance of the classifier. This balance is important in the present work because the SVM classifiers have a relatively slow training time. The exact time required can vary greatly depending on the data set and the SVM parameter values chosen. Additionally, using GAs to optimize the classifiers means that many thousands of classifiers must be trained and tested.

Arguably the most accurate evaluation method for classifiers built using small data sets is the leave-one-out method (Section 3.5). However, using this method on the data set in this work with a 2-class SVM optimized using a GA is impractical (Table 6.4). As a result it was decided to use k-fold cross validation (Section 3.5) with $k = 10$ as the method of evaluation. This method promises reasonably accurate estimates of classifier performance without a prohibitive computational performance hit.

Using k-fold cross validation, each class of samples, $\mathscr{C}_c$, $\mathscr{C}_u$, and $\mathscr{C}_n$, in the initial data set was randomly split into ten equal sized sets. For classes that could not be equally divided into ten equal sets, the remaining samples were added to the last (tenth) set. To ensure a fair comparison between 2-class and 3-class configurations, the data was split once into ten

sets and the same sets were used for all classifiers. For the 2-class classifiers, set i of $\mathscr{C}_w$ is a merger of set i of $\mathscr{C}_u$ and $\mathscr{C}_c$. This ensures set i of $\mathscr{C}_w$ is not biased to certain or uncertain whales—which might occur if new sets where randomly split again—since it contains the same ratio of certain/uncertain whales as all the other nine sets. The not-whale sets remain unchanged for all 2 and 3-class classifiers.

Using the rules of 10-fold cross validation, a classifier is trained and tested ten times to evaluate it's performance. During the i^{th} iteration, $i = 1, 2, \cdots, 10$, the i^{th} set is used to test the classifier and the remaining nine sets are combined and used for training. During testing, each object in set i is classified (assigned) to one class. This result is recorded in a confusion matrix for set i. After running the ten training/testing episodes, the ten confusion matrices are added together and the final confusion matrix is used to evaluate the classifier's performance.

6.7 Results

The results of running GA optimizations on the 14 classifiers described in Section 6.5 are presented in this section. Table 6.5 summarizes the experiments conducted, including the results of SVM parameter optimization. Figures 6.3 to 6.16 shows trends of GA optimization over time for each classifier. These trends are compared side-by-side in Figure 6.17. Table 6.6 shows the best chromosome patterns found by each GA. Figure 6.18 summarizes the classification results in a series of confusion matrices. Finally, Table 6.7 summarizes the performance measures for each GA optimized classifier. It should be noted that when comparing GA optimized results, only the balanced accuracy (a and $a^{\dagger}$) can be compared fairly because this is the value being optimized by the GA. Following are definitions of heading abbreviations and symbols used in Table 6.7 and the remaining of this chapter.

a – Balanced accuracy. Calculated using Equation 6.1.

$a^{\dagger}$ – The balanced accuracy before running the GA to optimize the classifier; all features are included and SVM parameters are not selected by the GA.

h_c – Certain whale classification (hit) rate expressed as a percentage. This is only calculated for the 3-class classifiers.

h_u – Uncertain whale classification rate expressed as a percentage. This is only calculated for the 3-class classifiers.

h_n – Not-whale classification rate expressed as a percentage.

h_w – Whale classification rate expressed as a percentage. For the 2-class classifier it is calculated as normal. For the 3-class classifier, a target is considered a whale if it classified as either certain or uncertain, and then the hit rate is calculated as a 2-class hit rate. In essence, we are analyzing the 3-class classifier as a 2-class classifier. This measure is useful when accuracy is not as important as the ability to distinguish a whale target, regardless of class, from a not-whale target.

f_p – False positive rate expressed as a percentage. For the 2-class case, it is calculated as normal. For the 3-class case a target is considered a whale if it classified as uncertain or certain, as done for h_w. This measure complements h_w.

n_F – Number of selected features for the classifier.

t – Time to perform the optimization in seconds. The GAs were executed on a computer with a Pentium 4, 3.0GHz CPU, and 1GB of RAM.

Table 6.5: Summary of GA optimized classifier experiments. Values marked with a * were optimized by the GA. Other values were precalculated as described in Section 6.5 and summarized in Table 6.3.

Classifier	GA Parameters				Classifier Parameters				
	p_m	p_c	n_p	n_g	$P(\mathscr{C}_w)$	$P(\mathscr{C}_n)$			
2Qd	0.001	0.9	75	23	0.28	0.72			
					$P(\mathscr{C}_c)$	$P(\mathscr{C}_u)$	$P(\mathscr{C}_n)$		
3Qd	0.0115	0.9	75	30	0.23	0.044	0.72		
					C	γ	w_w	w_n	
2Svm	0.039	0.685	75	30	1	1/d	1	1	
2SvmCg	0.039	0.685	75	30	2048	0.03125	1	1	
2SvmCgWi	0.039	0.685	75	30	8192	0.03125	2.6	1	
2SvmGaCg	0.039	0.685	75	30	32768*	0.01911*	1	1	
2SvmGaCgWi	0.039	0.685	75	30	32768*	0.01911*	2.6	1	
2SvmGaCgGaWi	0.039	0.685	75	30	374.387*	0.06391*	4.922*	4.144*	
					C	γ	w_c	w_u	w_n
3Svm	0.0245	0.9	75	30	1	1/d	1	1	1
3SvmCg	0.0245	0.9	75	30	32768	0.007813	1	1	1
3SvmCgWi	0.0245	0.9	75	30	2	8	3.08	16.6	1
3SvmGaCg	0.0245	0.9	75	30	8566.65*	3.577*	1	1	1
3SvmGaCgWi	0.0245	0.9	75	30	915.686*	0.1429*	3.08	16.6	1
3SvmGaCgGaWi	0.0245	0.9	75	30	2239*	0.1429*	0.5667*	3.678*	0.1778*

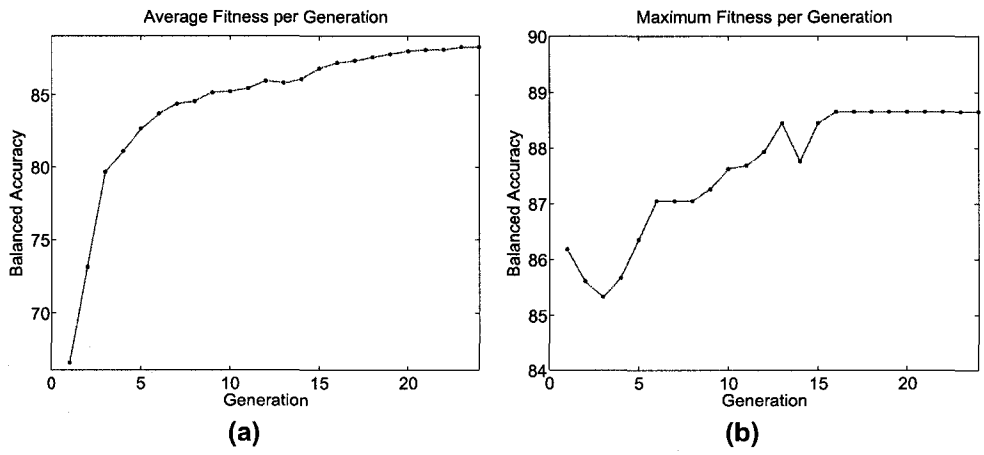


Figure 6.3: 2Qd classifier GA optimization trends.

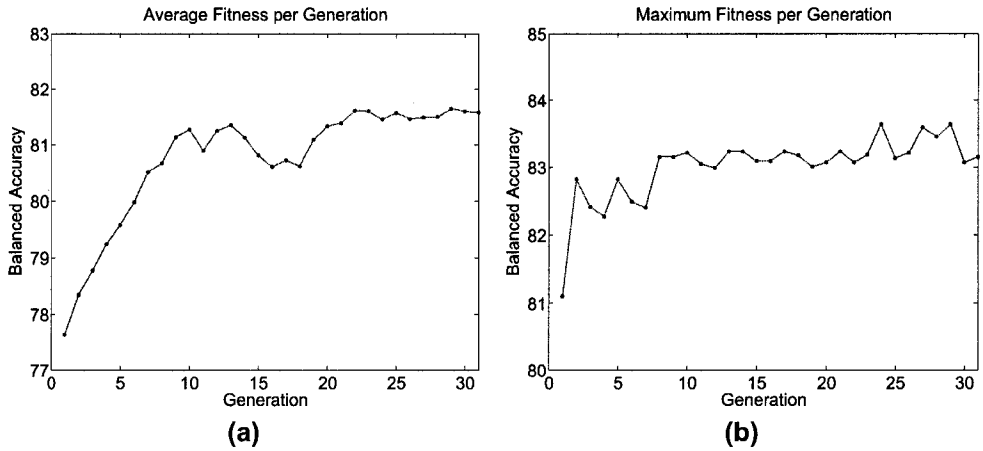


Figure 6.4: 2Svm classifier GA optimization trends.

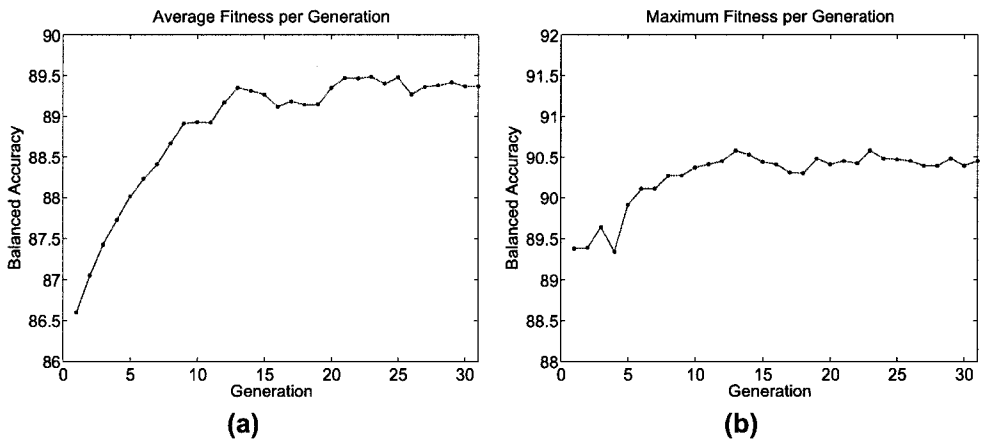


Figure 6.5: 2SvmCg classifier GA optimization trends.

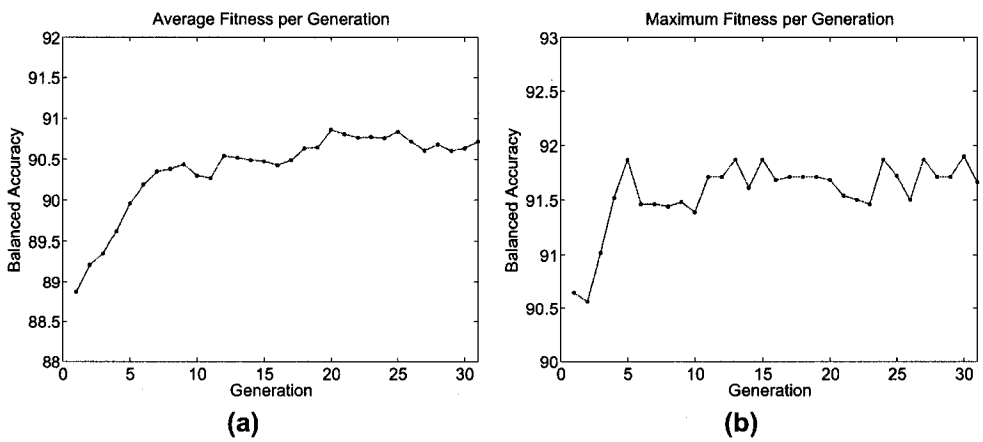


Figure 6.6: 2SvmCgWi classifier GA optimization trends.

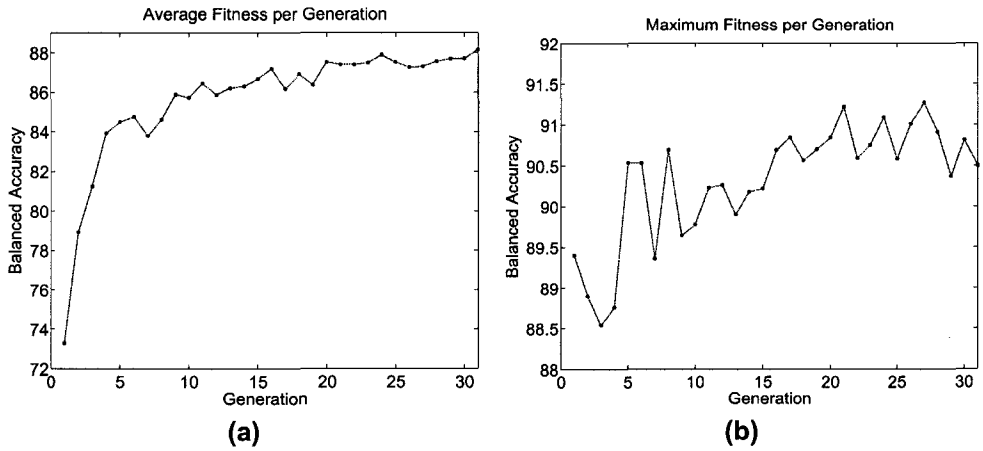


Figure 6.7: 2SvmGaCg classifier GA optimization trends.

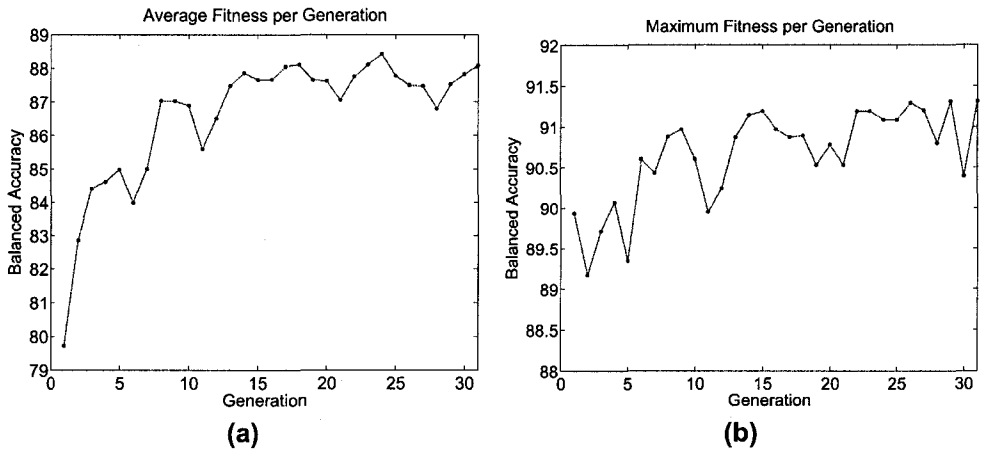


Figure 6.8: 2SvmGaCgWi classifier GA optimization trends.

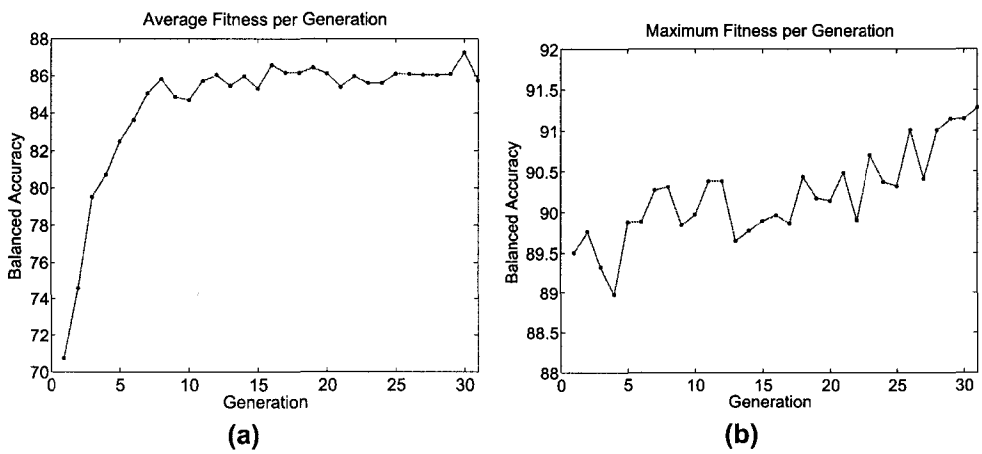


Figure 6.9: 2SvmGaCgGaWi classifier GA optimization trends.

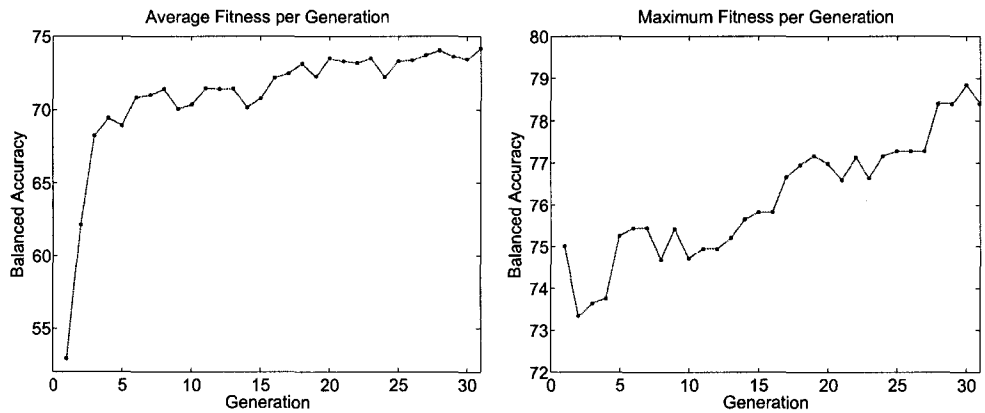


Figure 6.10: 3Qd classifier GA optimization trends.

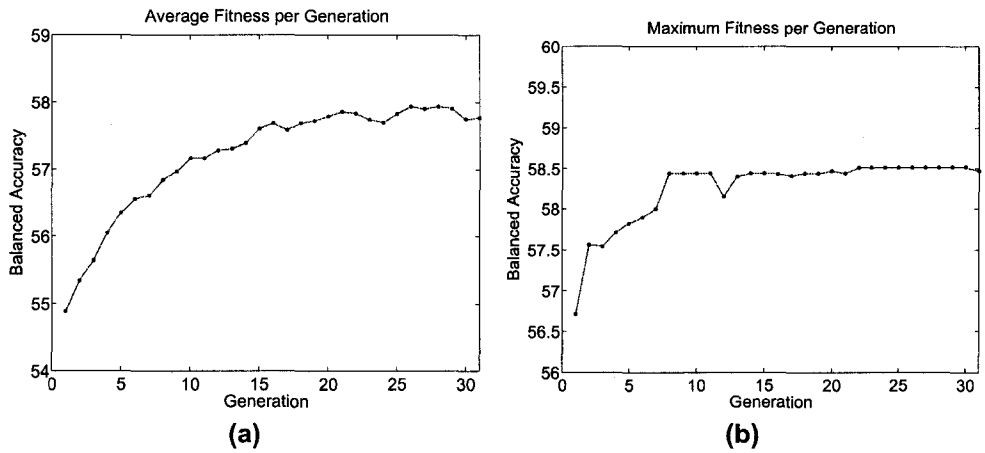


Figure 6.11: 3Svm classifier GA optimization trends.

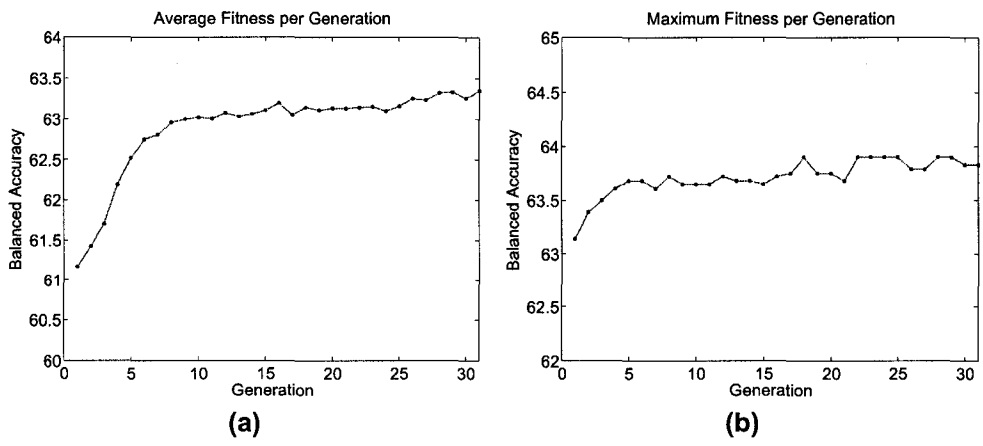


Figure 6.12: 3SvmCg classifier GA optimization trends.

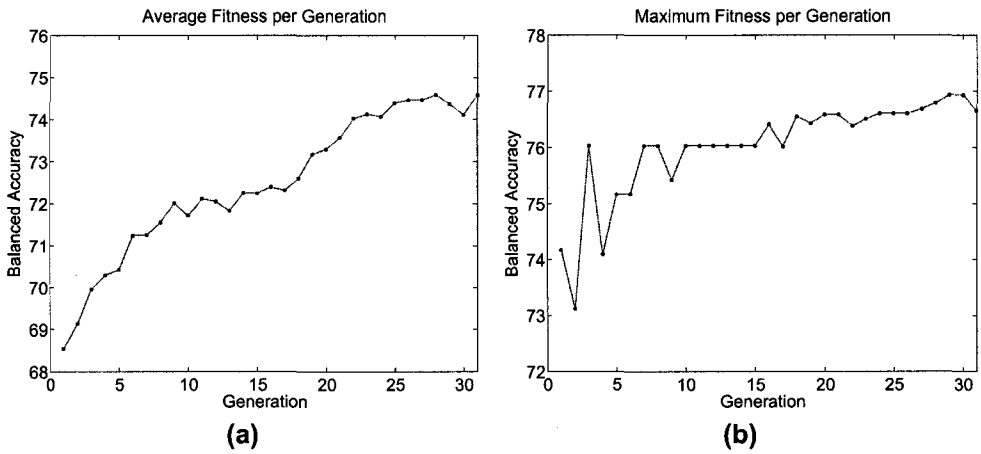


Figure 6.13: 3SvmCgWi classifier GA optimization trends.

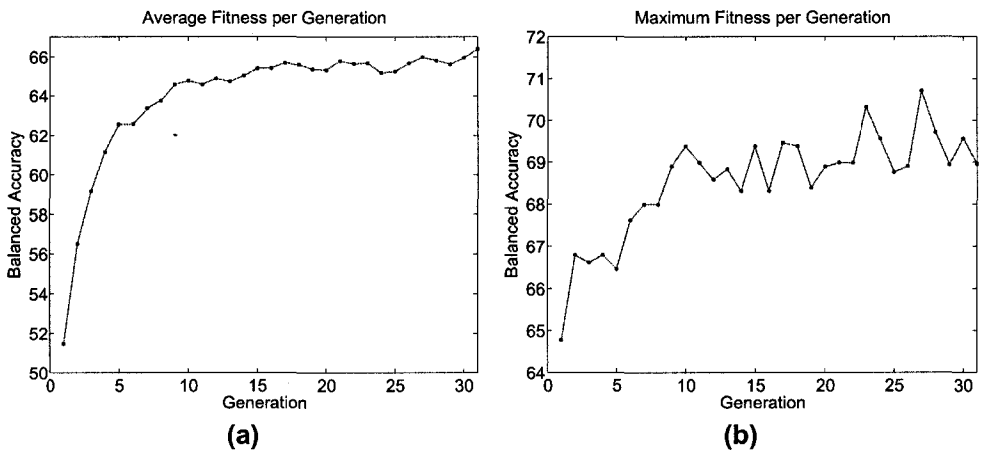


Figure 6.14: 3SvmGaCg classifier GA optimization trends.

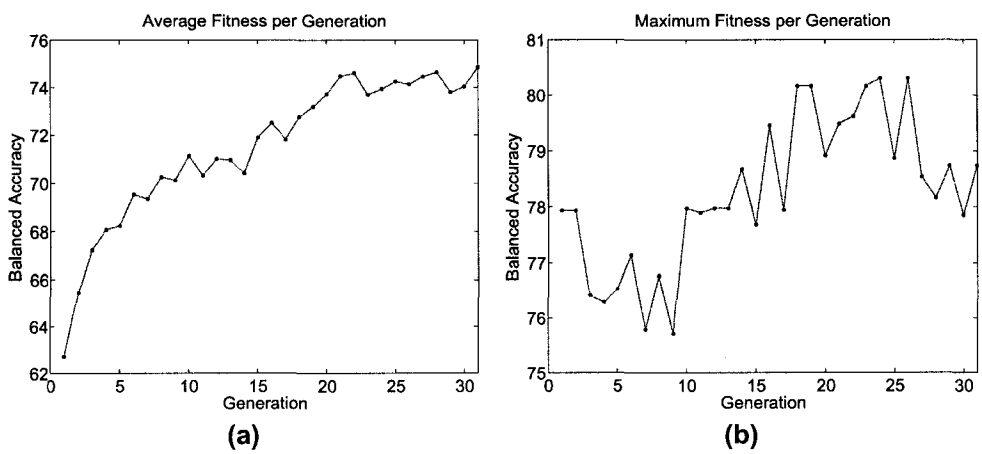


Figure 6.15: 3SvmGaCgWi classifier GA optimization trends.

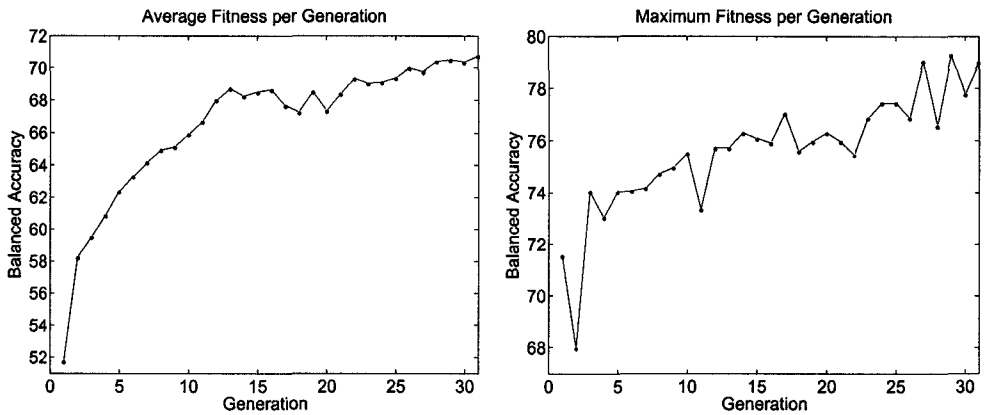


Figure 6.16: 3SvmGaCgGaWi classifier GA optimization trends.

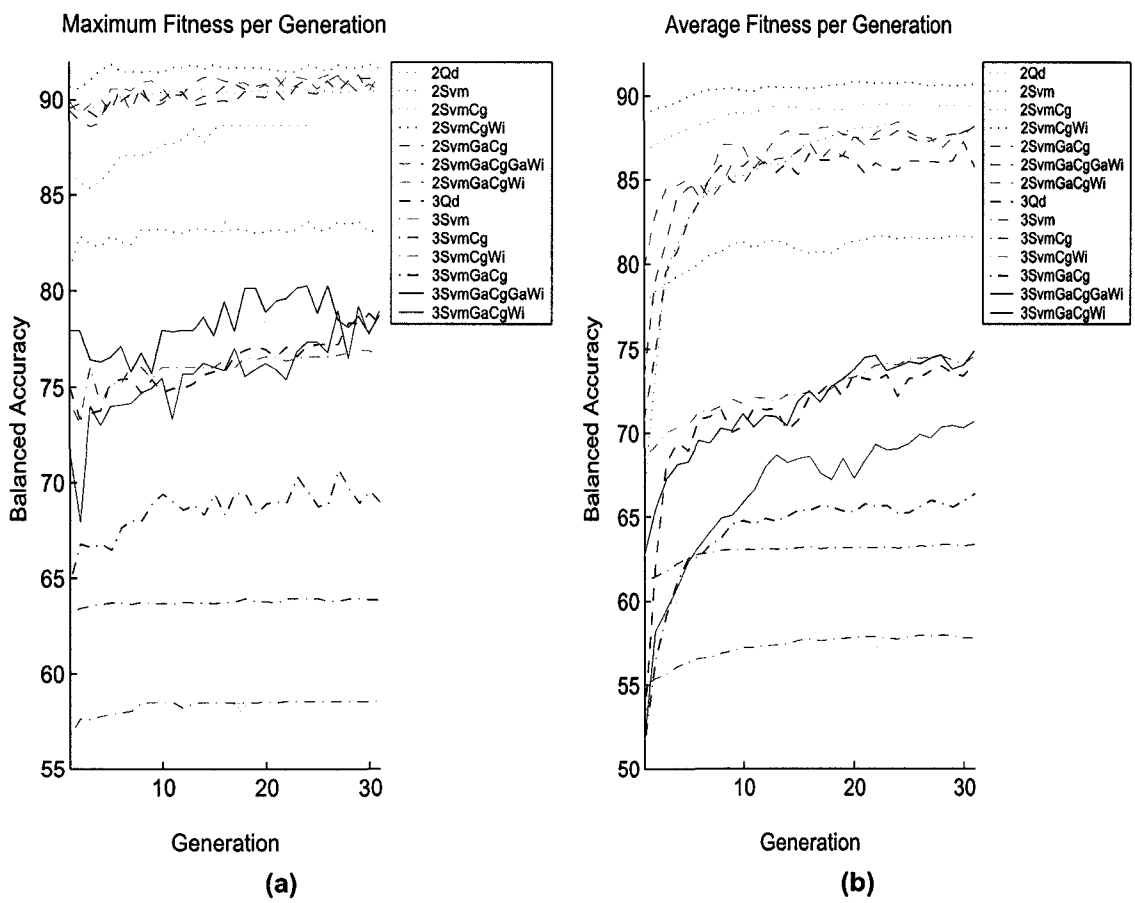


Figure 6.17: Summary of maximum (a) and average fitness (b) per GA generation for each classifier.

Table 6.6: Best chromosomes found by the GA for each classifier. Genes that encode classifier parameters are separated by |; see Section 6.5 for details.

Classifier	Chromosome
2Qd	000100000000011111101000
3Qd	101010000000100101100101
2Svm	001110000001000100110010
2SvmCg	110001101110100111101101
2SvmCgWi	101000000111011111101010
2SvmGaCg	11111|10000|100111101101101111111100
2SvmGaCgWi	11111|10000|111101111010101111101010
2SvmGaCgGaWi	111110|110100|10101|10011|000111101110000111101000
3Svm	001110100000110010110010
3SvmCg	001111100110111111111111
3SvmCgWi	001101101011000001100000
3SvmGaCg	11100|11101|100011011100011111100010
3SvmGaCgWi	10111|10101|001101101101100101110001
3SvmGaCgGaWi	000110|101110|000001|11001|10101|001111111001110010110001

Table 6.7: Summary of GA optimization results. Heading definitions are described at the end of Section 6.7.

Classifier	$a^\dagger$	a	h_c	h_u	h_n	h_w	f_p	n_F	t
2Qd	50.00	88.65	NA	NA	88.31	88.98	11.69	8	172
3Qd	33.33	78.84	90.52	82.46	63.55	96.42	36.45	9	380
2Svm	79.51	83.64	NA	NA	93.73	93.73	6.27	8	3767
2SvmCg	90.00	90.58	NA	NA	95.22	85.95	4.78	16	12737
2SvmCgWi	90.32	91.90	NA	NA	92.35	91.46	7.65	13	78165
2SvmGaCg	79.32	91.27	NA	NA	95.22	87.33	4.78	18	140501
2SvmGaCgWi	85.30	91.31	NA	NA	91.71	90.91	8.29	18	81397
2SvmGaCgGaWi	79.51	91.28	NA	NA	94.69	87.88	5.31	13	76619
3Svm	55.62	58.51	81.70	0	93.84	69.70	6.16	10	3783
3SvmCg	63.50	63.90	95.10	0	96.60	82.64	3.40	20	113021
3SvmCgWi	67.05	76.93	88.89	70.18	71.73	96.14	28.27	10	4028
3SvmGaCg	55.62	70.70	87.91	33.33	90.86	83.20	9.14	14	29335
3SvmGaCgWi	71.40	80.30	93.46	73.68	73.75	96.97	26.25	14	49383
3SvmGaCgGaWi	55.62	79.23	92.81	73.68	71.20	96.69	28.80	15	28762

	$\mathscr{C}_w$	$\mathscr{C}_n$
$\mathscr{C}_w$	323	40
$\mathscr{C}_n$	110	831

(a) 2Qd

	$\mathscr{C}_c$	$\mathscr{C}_u$	$\mathscr{C}_n$
$\mathscr{C}_c$	277	20	9
$\mathscr{C}_u$	6	47	4
$\mathscr{C}_n$	66	277	598

(b) 3Qd

	$\mathscr{C}_w$	$\mathscr{C}_n$
$\mathscr{C}_w$	267	96
$\mathscr{C}_n$	59	882

(c) 2Svm

	$\mathscr{C}_w$	$\mathscr{C}_n$
$\mathscr{C}_w$	312	51
$\mathscr{C}_n$	45	896

(d) 2SvmCg

	$\mathscr{C}_w$	$\mathscr{C}_n$
$\mathscr{C}_w$	332	31
$\mathscr{C}_n$	72	869

(e) 2SvmCgWi

	$\mathscr{C}_w$	$\mathscr{C}_n$
$\mathscr{C}_w$	317	46
$\mathscr{C}_n$	45	896

(f) 2SvmGaCg

	$\mathscr{C}_w$	$\mathscr{C}_n$
$\mathscr{C}_w$	330	33
$\mathscr{C}_n$	78	863

(g) 2Svm-GaCgWi

	$\mathscr{C}_w$	$\mathscr{C}_n$
$\mathscr{C}_w$	319	44
$\mathscr{C}_n$	50	891

(h) 2SvmGaCg-GaWi

	$\mathscr{C}_c$	$\mathscr{C}_u$	$\mathscr{C}_n$
$\mathscr{C}_c$	250	0	56
$\mathscr{C}_u$	3	0	54
$\mathscr{C}_n$	58	0	883

(i) 3Svm

	$\mathscr{C}_c$	$\mathscr{C}_u$	$\mathscr{C}_n$
$\mathscr{C}_c$	291	0	15
$\mathscr{C}_u$	9	0	48
$\mathscr{C}_n$	32	0	909

(j) 3SvmCg

	$\mathscr{C}_c$	$\mathscr{C}_u$	$\mathscr{C}_n$
$\mathscr{C}_c$	272	30	4
$\mathscr{C}_u$	7	40	10
$\mathscr{C}_n$	49	217	675

(k) 3SvmCgWi

	$\mathscr{C}_c$	$\mathscr{C}_u$	$\mathscr{C}_n$
$\mathscr{C}_c$	269	6	31
$\mathscr{C}_u$	8	19	30
$\mathscr{C}_n$	50	36	855

(l) 3SvmGaCg

	$\mathscr{C}_c$	$\mathscr{C}_u$	$\mathscr{C}_n$
$\mathscr{C}_c$	286	16	4
$\mathscr{C}_u$	8	42	7
$\mathscr{C}_n$	60	187	694

(m) 3SvmGaCgWi

	$\mathscr{C}_c$	$\mathscr{C}_u$	$\mathscr{C}_n$
$\mathscr{C}_c$	284	18	4
$\mathscr{C}_u$	7	42	8
$\mathscr{C}_n$	63	208	670

(n) 3SvmGaCgGaWi

Figure 6.18: Confusion matrices for each classifier configuration.

6.8 Discussion

In the following discussion performance refers to balanced accuracy (a, Section 6.7), unless stated otherwise. This is because the GA was designed to optimize a without consideration for any other metric, such as t, f_p, and h_w. To be fair, classifiers can only be compared on the basis on which they were optimized.

Overall, the SVM classifier outperforms the QD classifier, provided extensive SVM parameter calibration is performed. For the 2-class case, the SVM classifier has the best performance (a) when C and γ are calibrated; whether or not w_i is set is not that significant. C and γ can be set using the grid search or the GA; both methods result in a a better than the corresponding QD classifier. For the 3-class case, the SVM has comparable or better performance only when all 3 SVM parameters, C, γ, and w_i, are calibrated. Otherwise, the 3QD is better.

The 3-class configuration seems to be a much more complex problem. This is seen by the fact that both classifier types have a dramatic decrease in a when the classification problem is 3-class verses 2-class (a difference of 9.81 for the QD and a difference of 11.6 to 33.69 for the SVM). This is most likely due to the poor training set (low number and poor representative samples) for the uncertain whale class coupled with the fact that its feature values overlap significantly with those of the certain whale class.

Additionally, the 3SVMs require much more tweaking of parameters, especially weights, when dealing with 3-class data. This is partly because the data set is more severely unbalanced and the problem is more complex, but might also be because the SVM is by design a 2-class classifier being coerced into a 3-class classifier. This is clearly demonstrated by the difference in a within 2SVMs and within 3SVMs. In the 2SVMs, calibrating C and γ resulted in a best a gain of 7.63, then calibrating weights resulted in another gain of 0.63; but in the 3-class case, calibrating C and γ resulted in a 12.19 gain while calibrated weights

results in an additional gain of 9.6. Additionally, the method of calibrating C, γ, and w_i in the 3-class case has a much more pronounced effect than in the 2-class. In the latter case a is about the same at 91 ± 1 for all methods of choosing C, γ, and w_i. In the 3-class case there is up to a 16.4 difference between methods.

These results demonstrate the importance of calibrating C, γ, and (sometimes) w_i, for SVM classifiers, whether using a GA or a SVM parameter grid search. Comparing a with and without calibrating these parameters we see a maximum increase of 8.26 for 2SVM and 21.79 for 3SVM. This is consistent with results reported in literature (e.g. [65, 67, 107, 108]).

The results also demonstrate the value of using a GA to optimize classifiers. First, compare a of each classifier configuration before and after using the GA to select a feature subset (Table 6.7); the results show an increase in a with a dramatic reduction in the number of features. For QDs there is a minimum of 38.7% to maximum of 45.5% increase in a with 17 fewer features. For SVMs there is a minimum of 0.4% to maximum of 9.9% increase in a with 9 to 17 fewer features (These are results for GA-SVMs that did not optimize SVM parameters; only FSS was performed. With parameter optimization as well, there is up 23.6% increase in a with 9 to 12 fewer features). The results of FSS is much more dramatic for the QD type classifiers than SVMs. In fact, even without FSS, the SVMs perform reasonably well. This suggests that the SVM is not as sensitive to the initial feature set. However, in all cases there is an increase in a with FSS, regardless of C, γ, and w_i, hence one cannot conclude that FSS is not necessary (as some have suggested). In addition to the obvious benefits of increased accuracy, the value of feature reduction has been discussed in Section 3.4.

In addition to feature reduction, the results show that the GA can be easily extended to optimize a SVM by calibrating the key SVM parameters, C, γ, and w_i. When the GA is used to select C and γ the performance (as measured by a) is approximately the same

(for 2SVMs) or significantly better (for 3SVMs) than using the traditional method of grid search. Similarly, the GA can select weights that result an a that is as good as or better than selecting weights by hand. The benefit of the GA is that all parameters, including the feature subset, are selected automatically and simultaneously. This allows the GA to consider both individual and interaction effects of the three parameters and feature subsets in parallel. Regardless of the method of calibration, the results clearly show that SVM parameter calibration must be performed to successfully apply SVMs. This is clearly demonstrated by a minimum of 6.9% up to 21.8% increase in a when performing SVM calibration versus using default values.

The main disadvantage of the GA is that optimization time maybe large. This is mainly a concern for SVM classifiers, as is dramatically illustrated in the 2SvmGaCg and 3SvmCg case. The QD classifiers are optimized 10 to 200 times faster than the SVMs. This is mainly because training the SVM, which amounts to using a computationally complex optimization algorithm to solve a quadratic programming problem with the number of variables equal to the number of training samples (e.g. [133, 134]), is much more computationally complex than training QD classifiers, which is done by calculating basic statistics, such as the mean and covariance. The optimization time depends largely on the size of the data and the SVM parameter values. In particular, larger values of C results in longer training time for SVMs because the SVM tries harder to find, usually a more complex, discriminate surface to achieve increased accuracy.

When choosing the classifier to integrate into the final software solution the following was considered as the most important factors: (1) high a, (2) high h_w, and (3) low f_p. To this end, any one of the 2SVM classifiers with C, γ, and w_i calibrated is suitable. However, the 2SvmCgWi is chosen because it has the highest a, reasonably high h_w and h_n, and a reasonably low f_p. All 3-class classifiers are ruled out because of the much lower a (10% or more) compared to the 2-class classifiers. This means the final software cannot classify

whales as uncertain: a target is a whale or it is not. If uncertain whale classification is required, it can be accommodated by estimating the confidence of classifying a target as a whale. If the confidence is below a predetermined level, the whale is labeled uncertain. A simple measure of confidence is the distance an object represented by x is from the decision boundary. This is left for future work.

We now enter into a discussion of the details of GA optimization. First it can be seen that the GA does not result in the same feature subset for each classifier. This indicates there is no single subset of best features that apply to all configurations of 2-class and 3-class classifiers. This is mostly due to the interactions of other variables with feature subsets, highlighting the importance of not treating one variable (such as feature subset) in isolation of other variables (such as SVM parameters). As stated above, exploring combinations of variables simultaneously is a strength of the GA approach. It should also be noted that, although not obvious from the results, the best chromosome selected by the GA can be effected by the initial generation of chromosomes; that is, the starting point. In order to fairly compare results, all GAs used in this work were started with the same initial generation. Subsequent experiments with different starting points resulted in slightly different final chromosomes, and a different optimization path taken, but with no significant difference in the final fitness value (a, the optimization goal). This suggests that for a given classifier there can be more than one good, near optimal solution (feature subset and SVM parameter values). However, because the optimized result (a) is always nearly identical, it also suggests that the GA adequately searches the solution space for the best solution. This stands in contrast to some non-randomized techniques, such as SFFS and BSS (Section 3.4), that typically start at the same point and end with the same solution.

A general trend seen in all classifiers is the initial poor fitness, followed be a rapid increase in average fitness, followed by a slow increase in average fitness and, in most cases a leveling off (Figure 6.17). This is the expected, normal trend in GA optimization.

When the fitness tends to level off over several generations the GA is considered to have converged on its best solution. Running the GA longer will not significantly improve the results. In this work, all trends did level off to some degree (e.g. Figure 6.11), but arguably not all converged. In some cases (e.g. Figure 6.15) there remained an increasing trend when the GA terminated. This is a result of setting a hard limit on the number of generations. There are two ways to deal with this problem. First, if the DOE-GA technique detailed in Chapter 5 is used, then the method of steepest ascent in RSM should be applied to find the best operating conditions for the GA. In particular, it is recommend to follow the ascent in the direction of increasing number of generations. This would ensure there is enough generations for the GA to converge. A second method is to dynamically halt the GA using some algorithm that determines when the GA has converged.

Chapter 7

The Final Software System

This chapter briefly describes the software application, formally titled Marine Mammal Detector (MMD), that was hinted at in Section 1.2. The purpose of this software is to encapsulate the image processing and chosen classifier algorithms described in the proceeding chapters and present these in a user friendly, intuitive way. In addition, MMD provides support for most of the activities currently being done using pen and paper or through manually data entry (Section 2.3). The chapter starts with an overview of the software development methodology employed. Following that is a discussion of the software itself, including its basic design and main feature set.

7.1 Software Development Methodology

The development paradigm followed is a modification of the waterfall life cycle (model). Whereas the traditional, or "pure", waterfall [135] is very sequential with no provision for iteration, the modified approach augments the waterfall to allow for iteration between stages. It is illustrated in Figure 7.1. This approach has the advantage of having discrete, well defined stages, but allows knowledge about the problem gained in subsequent stages to be fed back into previous stages. Note that the final stage in most life cycles is the

maintenance stage. This has not been included in Figure 7.1 because it is out of the scope of this present work. The following subsections gives an overview of the development stages conducted in this current work. The definitions for these stages are not included; for more information see, for example, [136].

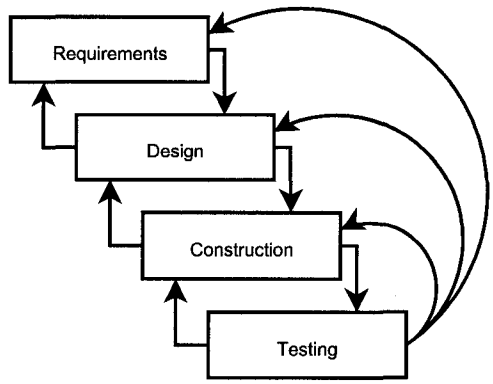


Figure 7.1: Software development model used.

7.1.1 Requirements Specification

Before the software was developed, a software requirements specification (SRS) document was created. The purpose of this document was to capture in concise and precise descriptions what the software is required to do; that is, the features of the software. An overriding concern here was to specify software functionality that captures the end-users current processes and activities in an intuitive way. The requirements gathering and documentation processes followed recommendations by [137] and the IEEE standard 830-1998 [138].

7.1.2 Design

The design follows state of the art object oriented techniques, including generic programming methods, as implemented in an imperative programming language. This is the logical choice given that most common programming languages have support for this style of

programming. Also, object oriented technique lends itself to a natural design because it attempts to model in the program code objects that exist in the real world.

One of the main design objectives is to allow the software to be used to aid marine mammal population assessments of mammals other than beluga whales. Hence the software was designed to allow a clear separation between beluga whale concepts (e.g. detection, classification, species names) and the main application concepts (e.g. GUI components, data base, overlays, tools).

In the process of building any relatively complex system (as is most software), modeling the system before building it is essential. Hence, the basic design of MMD was performed with the aid of the unified modeling language (UML), the standard modeling language for software engineering.

7.1.3 Construction

The implementation was performed in ISO C++. C++ was chosen as the development language because of (1) its high performance relative other commonly used languages, (2) availability of tools and libraries, (3) its great support for object oriented programming techniques, and (4) support for templates that allow a generic form a programming.

All development was done in the Microsoft Visual C++ 6.0 and 8.0 development environment on Windows 2000 and Windows XP operating systems (OS). Because the Microsoft Foundation Classes (MFC) library was used for implementing the GUI components, the software can only run on a Windows 2000 or later OS. This is not considered a limitation at this time since most desktop computers run some variant of the Windows OS.

7.1.4 Testing

Testing performed by the author was limited to white box testing. After coding specific classes, or implementing specific functionality a series of informal tests was used to confirm the system functions as expected and as specified in the SRS. An alpha version of the software was submitted to the target end-users (DFO) for black box testing, trials, and to provide feedback for further development. After fixing bugs and addressing other issues raised during feedback, a beta version has been developed and as of this writing has been submitted to the end-user for further black box testing. Once this phase of testing is over, and the major bugs fixed, a final version will be released.

7.2 Application Description

The MMD application is a standard Windows desktop GUI application (Figure 7.3). Figure 7.2 shows the relationship of the software to its main external entities. The application can be viewed as containing four broad components: (1) image manipulation and annotation, (2) analysis statistics and properties, (3) data storage, and (4) configuration.

7.2.1 Image Manipulation and Annotation

The first component is responsible for displaying aerial images and allows the user to manipulate these to perform analysis (2 and 4 in Figure 7.3). This component is superficially similar to standard imaging software, and offers many similar features, including the ability to scroll, zoom, and annotate the image. It differs in that it offers unique features that scientific personnel may need to perform analysis, including image overlays and specialized annotation objects, as described in more detail in the subsequent paragraphs.

All annotations are done on image overlays. Image overlays are abstractions representing the clear plastic overlays used during traditional manual analysis to cover the pho-

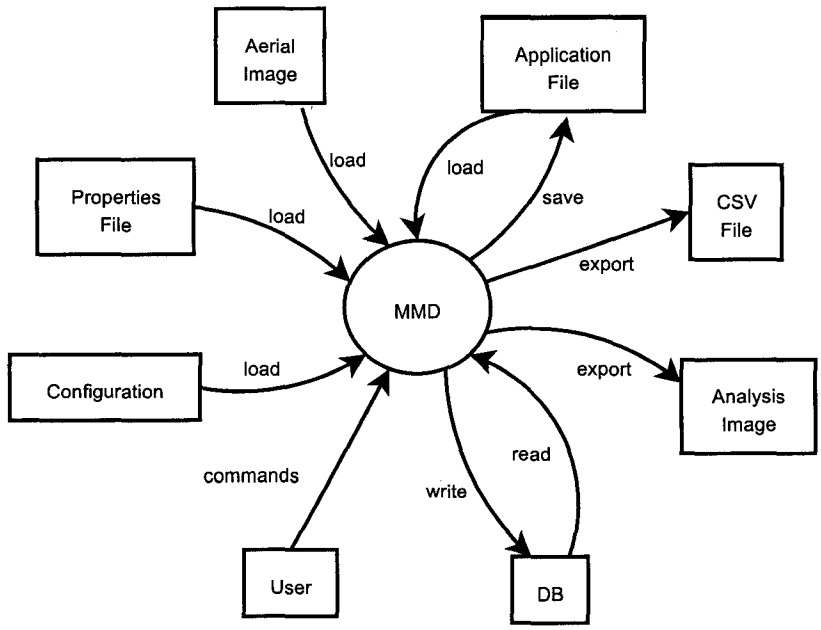


Figure 7.2: MMD context diagram showing the interaction of the software with the main external entities.

tographs. Mimicking physically adding and removing overlays on a photograph, these can be toggled on and off. Their purpose is to allow annotations to be made to (what appears to be) the photograph without modifying the original.

There are three types of overlays: analysis overlays, grid overlays, and unreadable mask overlays (Figure 7.4). Analysis overlays are used to mark on rather than directly marking on the original image. It is on this overlay classification results are visualized. Each identified target is circled in green for a certain whale and yellow for an uncertain whale. Whales are circled automatically after the automated detection and classification occurs. They can also be created, deleted, moved, and resized by the user. Each ellipse also contains properties that can be changed by the user identifying the species inside the region (e.g. adult whale, young). Other annotations include a sticky note that allows the user to record a comment directly on the image (really the overlay), and free-hand drawing tools that allow users to draw on the overlay. One use of these tools is to highlight a region of interest and make

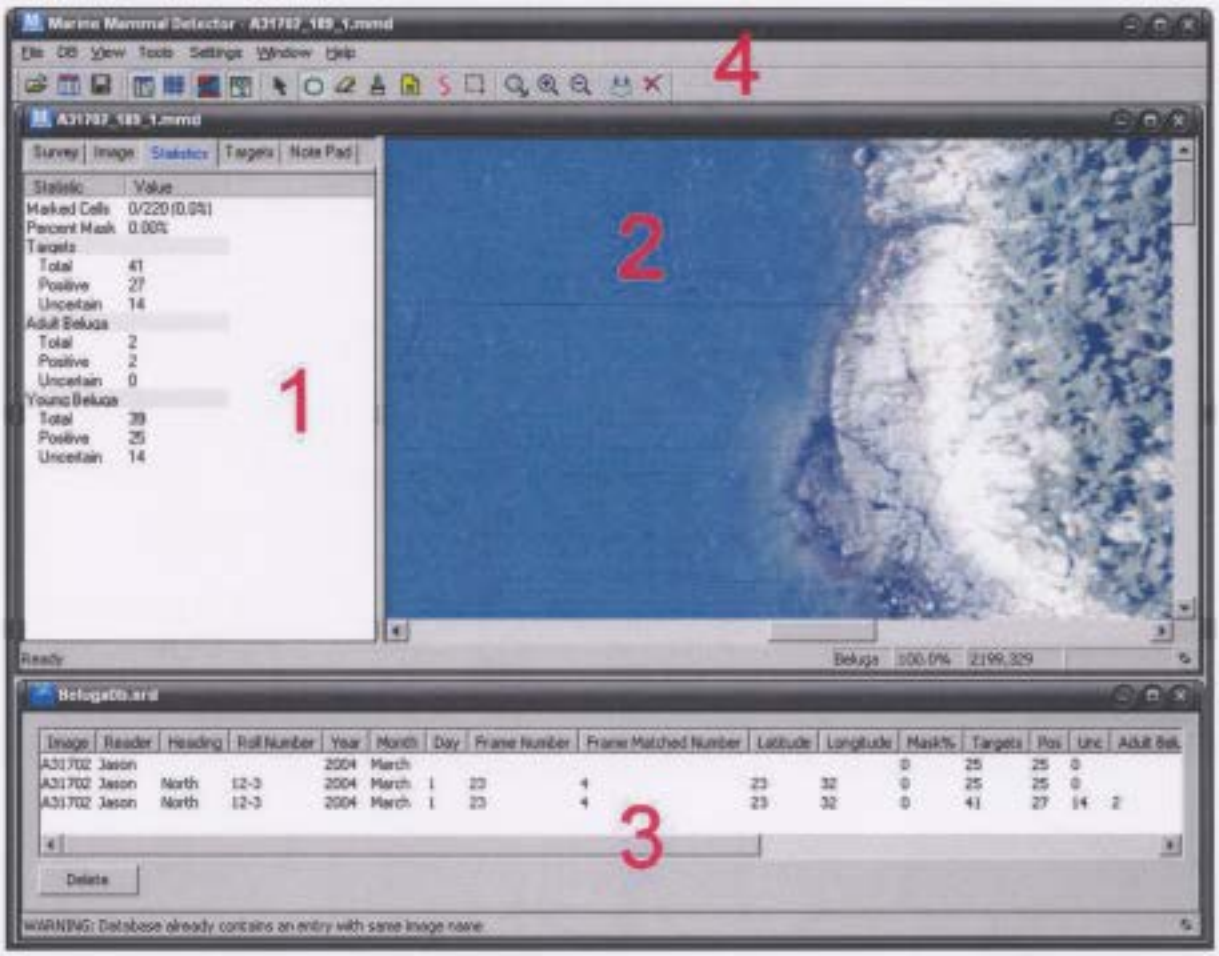


Figure 7.3: Screen capture of MMD showing the major GUI components. The panel labeled 1 is the data view (analysis and statistics). The panel labeled 2 is the image view. The window containing panels 1 and 2 is the main window the user interacts with. The window labeled 3 is the data base window. The tool bar and menu that provides most of the functionality to the user is labeled 4.

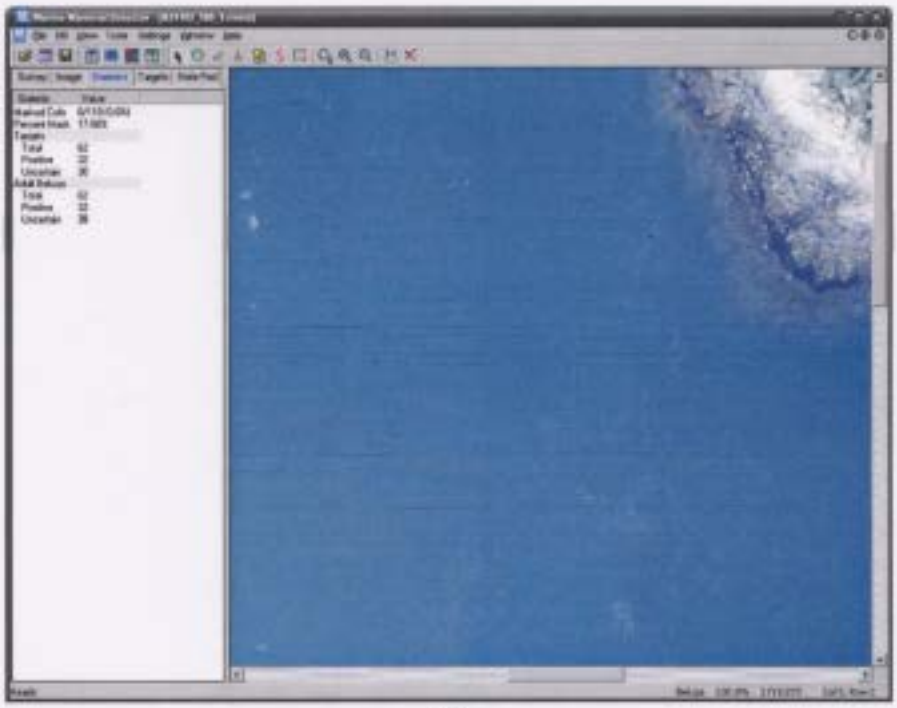
some comment about it.

Grid overlays are used to cover the image with a $n \times n$ grid, where n can be set by the user or is calculated based on a user selected size of a grid cell. The grid serves several purposes. First it keeps track of the users location in the image (e.g. during scrolling) and can be used to reference parts of the image using a row and column pair. Second, the grid records the parts of the image the user has analyzed, either doing manual detection or doing quality control. As the user scans areas of the image they can click on a grid cell to record the fact that they have completed a thorough visual inspection of the area under the cell. Using the grid they can systematically scan the entire image without scanning the same area multiple times while ensuring the entire image is scanned at least once.

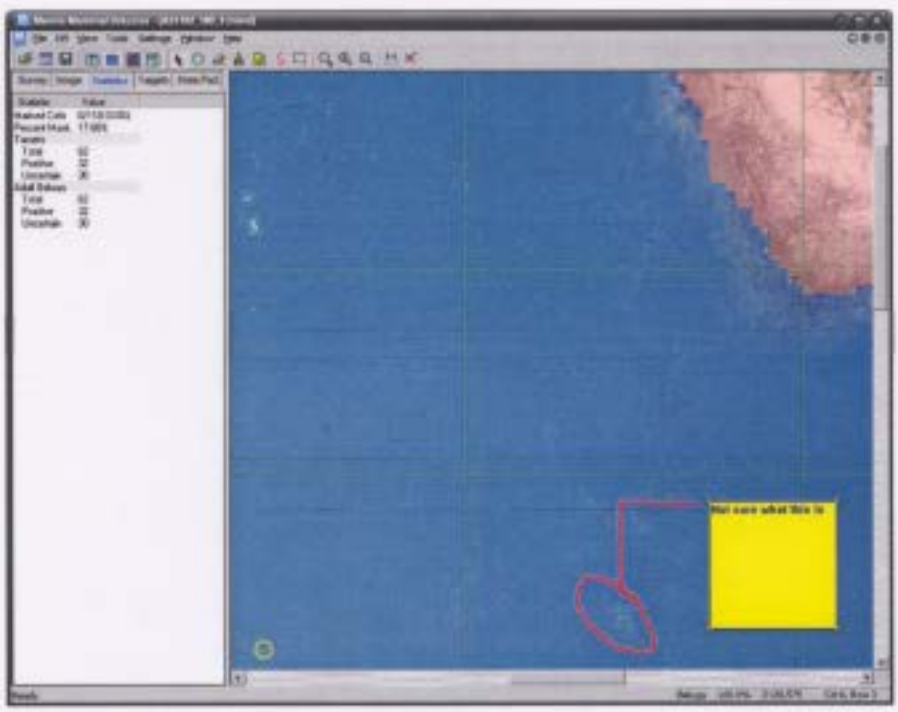
The unreadable mask overlay covers areas of the image where whale detection has not been performed. This mask is created automatically as detailed in Chapter 4. However, the user can manually erase and add to the mask using the mask eraser and painter tools. This allows them to manually inspect regions that where not automatically analyzed and reflect that fact by erasing the mask over the area. Similarly, they can add mask to cover unreadable areas that the automated mask creation algorithm missed.

7.2.2 Statistics and Properties

The properties and statistics component (panel 1 in Figure 7.3) contains data about the image being analyzed and other analysis related book keeping information. There are five types of data maintained and displayed to the user: (1) survey properties, (2) image properties, (3) image statistics, (4) targets, and (5) notes. Survey properties contain information pertaining to a set of images in a survey (e.g. survey year and location of survey). These are entered manually by the user and remain until changed, even between analysis sessions and application restart. Image properties record information specific to the current image being analyzed (e.g. reader name, image number, and analysis date). This information is



(a)



(b)

Figure 7.4: Screen captures of the MMD with example analysis overlays and annotations. (a) The application with all overlays off. (b) The same as (a) but with all overlays on. The grid is the green lines, the mask is the orange region, the green and yellow circles are target indications, the red line is a free-hand marking, and the yellow square is a sticky note opened for editing.

entered manually by the user, once for each new image analyzed. Survey and image property fields are completely configurable by the user; any number of fields can be configured. This allows the end-user to configure properties for a specific survey or mammal type. The fields are then presented to the user every time a new image is analyzed to allow them to enter information about the image.

Image statistics is updated automatically as the image is analyzed using the automated subsystem and via user manipulation (such as marking grid cells, adding a detected whale, and erasing the mask). Information maintained includes the number of positive targets, the number of uncertain targets, the fraction of marked grid cells, percent of the image covered by the unreadable mask (hence the percent of the image not analyzed), and the total number of each type of species.

The targets tab (Figure 7.3) shows a list of detected targets which is updated automatically as targets are manually or automatically detected. The list shows the target status (certain, uncertain) and location in the image. The user can click on the target to automatically scroll the image to center the target in the field of view. Targets can also be deleted directly from this list.

The notes tab provides a location for the reader to make notes. It is essentially a simple text editor.

7.2.3 Data Storage

The data storage component handles the persistent storage of analysis data. Besides the image, there are two types of files associated with each analysis session. The first is a binary application file that contains all information about an analysis session so sessions can be saved and reopened at any time. The data in this file can be exported to two formats. First, textual and numerical data can be exported to an extended comma separated value (CSV) formatted file for further analysis in other applications, such as spread sheets. Second,

snapshots of the current image or selectable subimages, with or without overlays, can be exported as JPEG, TIFF, or BMP images. An example usage of this is viewing the image on a computer that does not have the MMD application installed, or pasting results in a report.

The second data store is a data base survey file that records a summary of the results of each analyzed image. This content of this data base table can be viewed and edited as shown in the window labeled 3 in Figure 7.3. In particular, after each image is analyzed the results can be saved to this file. Once a survey (set of images) is complete, the data can be exported to CSV for further data analysis in such tools as Microsoft Excel and MATLAB.

7.2.4 Configuration

The configuration component handles the application, image processing, and classification configuration. The main configurable parameters are: (1) survey property fields (e.g. location, date, and elevation), together with default values; (2) image property fields together with their default values; (3) grid cell size or number of cells; (4) types of species being searched for; and (5) default species type, size of target region, and status of target for manually created targets.

Since MMD can be extended to mammals other than beluga, there is a separate configuration for each mammal detector module (e.g. beluga detector and seal detector). The configuration loaded depends on the type of detection the user requests. For example, if a seal detection module is requested, then the application uses the seal detector specific configuration parameters.

Chapter 8

Conclusions and Recommendations

8.1 Summary of Results and Conclusions

Automating beluga whale detection and classification in scanned photograph negatives (images) is not a trivial task. Whales are naturally occurring objects that are located in natural scenes, thus objects have a wide range of features values. This is unlike artificial objects in a well controlled environment (e.g. "nuts and bolts" in a manufacturing environment), which have a narrow range of feature values. In addition, there is a wide variation of "noise" features, such as land and wave crests, that must be recognized and removed. Because of these natural variations, any algorithms developed must be robust and adaptable.

The image segmentation algorithms are good for detecting all but the most unique whales. The process of applying a mask increases the speed of detection and reduces the number of non-whales objects segmented, thus also increasing the speed and accuracy of classification. This is achieved by reducing the search area to areas where whales naturally occur or are visually discernible. Importantly, this is achieved without losing large parts of the image where whales are recognizable. The primary adaptive thresholding method and secondary watershed algorithms work well, even though the objects are very small (few

number of pixels). The main problems are segmenting touching whales and whales deep below the surface.

Although rarely used for GAs, and never for GAs in the pattern recognition research area, DOE can be used to calibrate a GA. The process is systematic, based on sound theoretical statistics, and allows a parameter selection model to be created with a minimum number of experiments. This model can then be used to calculate the best parameters for optimal results from the GA (within the design space of the DOE model). This is certainly much more effective than current methods that rely on a set of general guidelines and ad-hoc experimentation. Moreover, it is not possible to optimize these parameters using the OFAT approach commonly employed because of the interaction between GA factors and the other factors in the experiment. A DOE or similar approach must be used. There does not appear to be a general DOE derived model for selecting GA parameters; hence, calibrating the GA is problem specific. This is clearly seen by the different models developed. If a general statement is to be made it's that high population, high generation, medium to high crossover, and low mutation settings produce the best results.

The GA is a good tool for automatically selecting a good feature subset, thus optimizing a given classifier. This is demonstrated for the minimum error rate quadratic discriminate classifier and the support vector machine classifier. For all but the most trivial feature sets, the process of obtaining the best subset is nearly impossible since it is usually unknown which feature combinations will achieve the best results and trying all combinations is computationally intractable. With a GA, an answer can usually be obtained within an hour to a day. In addition, the GA is completely automatic, so the developer can carry out different work while the GA is performing the analysis.

The GA has also been shown useful for optimizing (calibrating) the SVM parameters. The increase in accuracy obtaining when calibrating the SVM parameters, especially C and γ, clearly indicate the necessity of SVM calibration. Although the SVM parameter

grid search method achieved similar results, the GA has the advantage of taking into consideration parameter values and feature subsets simultaneously, thus any interaction effects between feature subsets and parameter combinations are considered. Additionally, the GA presents a unified, automated method to achieve feature reduction and parameter calibration.

The SVM classifiers can perform as good as or better than the traditional QD classifiers on both 2-class and 3-class problems. However, to achieve this enhanced performance, SVM parameter calibration is required. Although FSS does usually improve SVM performance, SVMs can perform well without it. QD classifiers, on the other hand, can perform horribly if FSS is not performed.

Overall, the MMD software system is appropriate for general use. The software is implemented to meet end user requirements as per the SRS. As a result, the features added to the software, even without image processing and classification algorithms, will aid scientific staff in performing population assessments by migrating the user from a traditional pen and paper approach to a computer aided approach. The value added features of automated detection and classification of whales is an enhancement not found in any other existing software product of a similar nature.

8.2 Contributions

Following is a summary of the main contributions of this work.

1. Development of novel marine mammal detection and classification algorithms based on state of the art image processing and pattern recognition methods.

2. Development of a graphical user interface (GUI) for these algorithms. The GUI also includes advanced features that automate or aid the user in performing most of the manual activities that is currently standard practice in photo analysis for population

assessments. The complete system will dramatically increase the productivity of scientist and technicians conducting population assessments.

3. Investigation of the feasibility of using design of experiments (DOE) as an aid in calibrating genetic algorithms (GA) used for classifier optimization.

4. Investigation of support vector machine (SVM) classifier optimization using GAs. Very little work has been conducted in this area and none compared results to the commonly used method of grid search with cross validation for parameter calibration.

8.3 Recommendations

Following are recommendations for future work.

- Improve adjacent and occluded whale separation techniques. The watershed segmentation can be improved to dynamically calculate a minimum variation specific to the ROI of the adjacent whales. In addition, the watershed can recognize the orientation of whales and remove lines that cut across whales. A different approach is to try fitting ellipses to the image to segment individual whales.

- Experiment with other measures of texture for creating the unreadable mask. Besides standard deviation, other statistical approaches [128, 139], may be appropriate, especially if combined.

- It is suspected that better DOE-GA results can be obtained if the method of steepest ascent in the RSM is used. This would allow the experimenter to find the optimum GA operating point even if that optimum is outside of the initially chosen design space.

- It is expected that the uncertain/young whale class can be discriminated better, thus improving the error rate for this class and the overall whale detection rate, by including more data in the training of uncertain whales. On a related note, if 2-class classification remains a better alternative (as it turned out to be in this work), each classified target could have an associated confidence level. If the confidence level is below some predetermined level, the whale is classified as uncertain. Alternatively, all classified targets could have a confidence level, say scaled between 0 and 1, and the user can chose what the uncertain-certain threshold level is.

- Investigate other classifier types, such as neural networks. In addition, a comparison between some feature based approaches (such as those used in this thesis) with template based approaches would be useful.

- A spatial feature should be added to the classifier feature list, or used in the image processing stage to aid in filtering objects. One feature not exploited by the image processing or pattern recognition system is the relative location of an object to other objects and to land. In general whales are isolated or located in small pods and are rarely found adjacent to shorelines. Also, the confidence that an object is a whale decreases as the noise (such as wave crests) surrounding the object increases.

- Since a high overall whale detection rate is more important than accuracy, a fitness function that reflects this could be used in the GA.

- Compare software aided detection and classification counting results to traditional manual counting results. When considering the software results, it might be useful to consider both fully automated results and results obtained after user quality control. The latter is a much more realistic use of the software since all detections must be confirmed by the user. Recall that the software is an aid for counting whales, not a user replacement.

- Presently the image processing and pattern recognition algorithms are developed using a set of images collected at a fixed image resolution and airplane altitude. The algorithms should be modified to allow detection and classification for a finite set of common altitudes and image resolutions.

- Considering adding on-line learning of algorithms. This would allow end-users to adjust the algorithms, in particular the pattern recognition algorithms, to new data sets and/or improve the classifier over time.

- Investigate methods to reduce execution time. A simple method is to refactor the code at bottle necks (e.g. preallocating and reusing image buffers).

References

[1] G. B. Stenson, "Computer-assisted image analysis techniques to support marine mammal population assessments," DFO Science Strategic Fund Project Application Form, 2001.

[2] S. H. Ridgway and R. Harrison, Eds., *Handbook of Marine Mammals*. Academic Press, 1989, vol. 4.

[3] "American cetacean society fact sheet: Beluga whale, *Delphinapterus leucas*," American Cetacean Society, 2005. [Online]. Available: http://www.acsonline.org/factpack/BelugaWhale.htm

[4] R. M. Nowak, *Walker's Marine Mammals of the World*. Baltimore, Maryland: The Johns Hopkins University Press, 2003.

[5] S. K. Katona, V. Rough, and D. Richardson, *A Field Guide to the Whales, Porpoises, and Seals of the Gulf of Maine and eastern Canada : Cape Cod to Newfoundland*, 3rd ed. New York: Scribner, 1983.

[6] R. M. Nowak, *Walker's Mammals of the World*, 5th ed. Baltimore, Maryland: The Johns Hopkins University Press, 1991, vol. 2.

[7] M. Lundberg, "The beluga whale," 2005. [Online]. Available: http://www.explorenorth.com/library/weekly/aa051200a.htm

[8] DFO.2002, "Northern Quebec (Nunavik) beluga (Delphinapterus leucas)," Department Fisheries and Oceans," DFO Sci. Stock Status Rep. E4-01 (2002), 2002.

[9] D. E. Gaskin, "Marine biodiversity monitoring: Monitoring protocol for marine mammals in canadian waters," Department of Zoology, University of Guelph, Tech. Rep., 1996, a report by the Marine Biodiversity Monitoring Committee to the environmental monitoring and assessment network of Environment Canada. [Online]. Available: http://www.eman-rese.ca/eman/ecotools/protocols/marine/mammals/intro.html

[10] S. Lawrence Beluga Recovery Team, "St. Lawrence beluga recovery plan," Department of Fisheries and Oceans and World Wildlife Fund Canada, Tech. Rep., Dec 1995.

[11] J.-F. Gosselin, V. Lesage, and A. Robillard, "Population index estimate for the beluga of the St Lawrence River estuary," Fisheris and Oceans Canada, Maurice-Lamontagne Institute, Tech. Rep. Research Document 2001/049, 2000.

[12] G. B. Stenson, L.-P. Rivest, M. O. Hammill, J. F. Gosselin, and B. Sjare, "Estimating pup production of harp seals, *Pagophilus Groenlandicus*, in the northwest Atlantic," *Marine Mammal Science*, vol. 19, no. 1, pp. 141–160, Jan 2003.

[13] G. B. Stenson, R. A. Myers, I.-H. Ni, and W. G. Warren, "Pup production and population growth of hooded seals (*Cystophora cristata*) near Newfoundland, Canada," *Canadian Journal of Fisheries and Aquatic Sciences*, vol. 54, no. 1, pp. 209–216, 1997.

[14] G. B. Stenson, M. O. Hammill, M. C. S. Kingsley, B. Sjare, W. G. Warren, and R. A. Myers, "Is there evidence of increased pup production in northwest Atlantic harp seals, *Pagophilus Groenlandicus*?" *ICES Journal of Marine Science*, vol. 59, pp. 81–92, 2002.

[15] G. B. Stenson, R. A. Myers, I.-H. Ni, W. G. Warren, and M. C. S. Kingsley, "Pup production of harp seals, *Phoca groenlandica*, in the northwet Atlantic," *Canadian Journal of Fisheries and Aquatic Sciences*, vol. 50, no. 11, pp. 2429–2439, 1993.

[16] "MiscroGOP Software," 2006. [Online]. Available: http://www.contextvision.com

[17] "ImageTool," University of Texas Health ahd Science, 2006. [Online]. Available: http://www.ddsdx.uthscsa.edu/dig/itdesc.html

[18] "GIMP," 2006. [Online]. Available: http://www.gimp.org

[19] "Adobe Photoshop," 2006. [Online]. Available: www.adobe.com/products/photoshop/main.html

[20] Matrox, "Matrix Imaging Library (MIL)," 2005. [Online]. Available: http://www.matrox.com/imaging/products/mil/home.cfm

[21] A. S. Laliberte and W. J. Ripple, "Automated wildlife counts from remotely sensed imagery," *Wildlife Society Bulletin*, vol. 31, pp. 362–371, 2003.

[22] D. S. Gilmer, J. A. Brass, L. L. Strong, and D. H. Card, "Goose counts from aerial photographs using an optical digitizer," *Wildlife Society Bulletin*, vol. 16, pp. 204–206, 1988.

[23] D. Bajzak, "Automated waterfowl census," in *Proceedings of Symposium on Remotes Sensing and Photo Interpretation*, A. H. Aldred, P. Gimbarzevsky, and B. Silverside, Eds. Banff, Alberta: International Society of Photogrammetry, 1974, pp. 137–145.

[24] D. Bajzak and J. F. Piatt, "Computer-aided procedure for counting waterfowl on aerial photographs," *Wildlife Society Bulletin*, vol. 18, pp. 125–129, 1990.

[25] D. J. Cunningham, W. H. Anderson, and R. M. Anthony, "An image processing program for automated counting," *Wildlife Society Bulletin*, vol. 24, pp. 345–347, 1996.

[26] "ERDAS IMAGINE," ERDAS, Inc. Atlanta, GA, 2006. [Online]. Available: http://gi.leica-geosystems.com/LGISub1x33x0.aspx

[27] P. N. Trathan, "Image analysis of color aerial photography to estimate penguin population size," *Wildlife Society Bulletin*, vol. 32, no. 2, pp. 332–343, 2004.

[28] The MathWorks, "MATLAB," 2005. [Online]. Available: http://www.mathworks.com

[29] R. Gosine, L. Gamage, and A. Trites, "Image processing techniques for automatic counting of sea-lions from aerial images: Results from a feasibility study," in *995 IASTED International Conference on Modeling and Simulation*, Colombo, Sri Lanka, 1995.

[30] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. John Wiley & Sons, 2001.

[31] F. A. Gougeon, "Comparison of possible multispectral classification schemes for tree crown individually delineated on high spatial resolution MEIS images," *Canadian Journal of Remote Sensing*, vol. 21, pp. 1–9, 1995.

[32] P. Meyer, K. Staenz, and K. I. Itten, "Semi-automated procedures for tree species identification in high spatial resolution data from digitized colour infrared-aerial photography," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 51, pp. 5–16, 1996.

[33] L. J. Quackenbush, P. F. Hopkins, and G. J. Kinn, "Developing forestry products from high resolution digital aerial imagery," *Photogrammetric Engineering and Remote Sensing*, vol. 66, pp. 1337–1346, 2000.

[34] C. A. Maltin, S. M. Hay, M. I. Delday, G. E. Lobley, and P. J. Reeds, "The action of the beta-antagonist clenbuterol on protein metabolism in inverted and denervated phasic muscles," *Biochemistry Journal*, vol. 261, pp. 965–971, 1989.

[35] C. A. Glasbey, G. W. Hogan, and J. E. Daryshire, "Image analysis and three-dimensional modeling of pores in soil aggregates," *Journal of Soil Science*, vol. 42, pp. 479–488, 1991.

[36] R. O. Duda, "Pattern recognition for HCI," http://www.engr.sjsu.edu/~knapp/HCIRODPR/PR_home.htm, 1997.

[37] R. Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches*. New York: John Wiley & Sons, 1992.

[38] M. Stefik, *Introduction to Knowledge Systems*. San Francisco: Morgan Kaufmann, 1995.

[39] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. New York: Academic Press, 1990.

[40] E. Alpaydin, *Introduction to Machine Learning*. Cambridge, MA: MIT Press, 2004.

[41] T. Bayes, "An essay towards solving a problem in the doctrine of chances," *Philosophical Transactions of the Royal Society (London)*, vol. 53, pp. 370–418, 1763.

[42] P. S. Laplace, *Théorie Analytique des Probilitiés*. Paris, France: Courcier, 1812.

[43] C. K. Chow, "An optimum character recognition system using decision functions," *IRE Transactions*, pp. 247–254, 1957.

[44] J. M. Bernardo and A. F. M. Smith, *Bayesian Theory*. New York: Wiley, 1996.

[45] M. Hearst, B. Sch olkopf, S. Dumais, E. Osuna, and J. Platt, "Trends and controversies - support vector machines," *IEEE Intelligent Systems*, vol. 13, no. 4, pp. 18–28, 1998.

[46] B. E. Boser, I. M. Guyon, , and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, D. Haussler, Ed. ACM Press, 1992, pp. 144–152.

[47] V. Vapnik and A. Chervonenkis, *Theory of Pattern Recognition*. Nauka, 1974, in Russian.

[48] V. Vapnik, *Estimation of Dependences Based on Empirical Data*. Nauka, 1979, english version: Springer Verlag, New York, 1982.

[49] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[50] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[51] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

[52] H. Byun and S.-W. Lee, "A survey of pattern recognition applications of support vector machines," *International Journal of Pattern Recognition*, vol. 17, no. 3, pp. 459–486, 2003.

[53] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 131–159, 2002.

[54] K. P. Bennett and C. Campbell, "Support vector machines: Hype or hallelujah?" *SIGKDD Explorations*, vol. 2, no. 2, pp. 1–13, 2000.

[55] L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. le Cun, U. Muller, E. Sackinger, P. Simard, , and V. Vapnik, "Comparison of classifier methods: A case study in handwriting digit recognition," in *International Conference on Pattern Recognition*. IEEE Computer Society Press, 1994, pp. 77–87.

[56] J. Friedman, "Another approach to polychotomous classification," Department of Statistics, Stanford University, Tech. Rep., 1996.

[57] U. Kreßel, "Pairwise classification and support vector machines," in *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 255–268.

[58] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin DAGs for multiclass classification," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, , and K.-R. Mueller, Eds. Cambridge, MA: MIT Press, 2000, pp. 547–553.

[59] V. N. Vapnik, *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

[60] J. Weston and C. Watkins, "Multi-class support vector machines," Department of Computer Science, University of London, Tech. Rep. CSD-TR-98-04, 1998.

[61] ——, "Support vector machines for multiclass pattern recognition," in *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, April 1999.

[62] K. Crammer and Y. Singer, "On the learnability and design of output codes for multiclass problems," in *Computational Learing Theory*, 2000, pp. 35–46.

[63] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, March 2002.

[64] S. Abe, *Support Vector Machines for Pattern Classification*. New York: Springer-Verlag, 2003.

[65] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Department of Computer Science and Information Engineering, National Taiwan University, Tech. Rep., 2003.

[66] S. S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with gaussian kernel," *Neural Computation*, vol. 15, pp. 1667–1689, 2003.

[67] H. T. Lin and C. J. Lin, "A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods," Department of Computer Science and Information Engineering, National Taiwan University, Tech. Rep., 2003, submitted to Neural Computation.

[68] C.-C. Chang and C.-J. Lin, "LIBSVM FAQ," 2004. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm/faq.html

[69] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, "The entire regularization path for the support vector machine," *Machine Learning Research*, vol. 5, pp. 1391–1415, 2004.

[70] R. Bellman, *Adaptive Control Process: A Guided Tour*. Princeton University Press, 1961.

[71] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbour" meaningful?" in *Proceedings of the 7th International Conference on Data Theory*, ser. Lecture Notes in Computer Science, vol. 1540. Springer-Verlag, 1999, pp. 217–235.

[72] M. Kapper, "The curse of dimensionality," in *5th Online World Conference of Soft Computing in Industrial Applications*, 2000.

[73] P. F. Evangelista, "Taming the curse of dimensionality in kernels and novelty detection," in *Advances in Soft Computing*, J. Kacprzyk, Ed. Berlin: Springer Verlag, 2006.

[74] D. W. Aha and R. L. Bankert, "A comparative evaluation of sequential feature selection algorithm," in *Learning from Data : Artificial Intelligence and Statistics*, ser. Lecture Notes in Statistics, D. Fisher and H.-J. Lenz, Eds. New York: Springer, 1996, vol. 112, ch. 19.

[75] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1–2, pp. 273–324, 1997, secial issue on relevance.

[76] M. A. Hall and L. A. Smith, "Feature subset selection: A correlation based filter approach," in *Proceedings of the Fourth International Conference on Neural Information Processing and Intelligent Information Systems*, N. Kasabov, Ed., 1997, pp. 855–858.

[77] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for SVMs," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds. Cambridge, MA: MIT Press, 2000, pp. 668–674.

[78] J. Doak, "An evaluation of feature selection methods and their application to computer security," Dept. of Computer Science, Univ. of California at Davis, Tech. Rep. CSE-92-18, 1992.

[79] Z. Sun, G. Bebis, and R. Miller, "Boosting object detection using feature selection," in *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance*. IEEE, 2003.

[80] H. Vafaie and K. De Jong, "Robust feature selection algorithms," in *Proceedings of the Fifth Conference on Tools for Artificial Intelligence*. Boston, MA: IEEE Computer Society Press, 1993, pp. 356–363.

[81] D. Skalak, "Prototype and feature selection by sampling and random mutation hil climbing algorithms," in *Proceedings of the Eleventh International Machine Learning Conference*. New Brunswick, NJ: Morgan Kaufmann, 1994, pp. 293–301.

[82] J. Kittler, "Feature selection and extraction," in *Handbook of Pattern Recognition and Image Processing*, T. Y. Young and K. S. Fu, Eds. Orlando, Forida: Academic Press, 1986, pp. 59–83.

[83] W. Siedlecki and J. Sklansky, "On automatic feature selection," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 2, no. 2, pp. 197–220, 1988.

[84] A. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 153–158, February 1997.

[85] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, no. 3, pp. 131–156, 1997.

[86] F. J. Ferri, P. Pudil, M. Hatef, and J. Kittler, "Comparative study of techniques for large-scale feature selection," in *Pattern Recognition in Practice IV*, E. S. Gelsema and L. N. Kanal, Eds. Elsevier Science, 1994, pp. 403–413.

[87] I.-S. Oh, J.-S. Lee, and B.-R. Moon, "Hybrid genetic algorithms for feature selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, November 2004.

[88] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection," *Pattern Recognition Letters*, vol. 10, pp. 335–347, 1989.

[89] F. Z. Brill, D. E. Brown, and W. N. Martin, "Fast genetic selection of feature for neural network classifiers," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 324–328, March 1992.

[90] J. H. Yang and V. Honavar, "Feature subset selection using genetic algorithm," *IEEE Intelligent Systems*, vol. 13, no. 2, pp. 44–49, 1998.

[91] K. I. Kuncheva and L. C. Jain, "Nearest neighbor classifier: Simultaneous editing and feature selection," *Pattern Recognition Letters*, vol. 20, pp. 1149–1156, 1999.

[92] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, and A. K. Jain, "Dimensionality reduction using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 2, pp. 164–171, July 2000.

[93] M. Kudo and J. Sklansky, "Comparison of algorithms that select features for pattern recognition," *Pattern Recognition*, vol. 33, no. 1, pp. 24–41, 2000.

[94] S. J. Raudys and A. K. Jain, "Small sample size effects in statistical pattern recognition: Recommendations for practitioners," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, March 1991.

[95] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[96] L. Davis and M. Steenstrup, "Genetic algorithms and simulated annealing: An overview," in *Genetic Algorithms and Simulated Annealing*, L. Davis, Ed. Pitman Publishing, 1987.

[97] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[98] R. Biesbroek, "GA tutorial," ESA, 1999. [Online]. Available: http://www.estec.esa.nl/outreach/gatutor/

[99] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan, Ann Arbor, 1975.

[100] R. B. Hollstien, "Artificial genetic adaptation in computer control systems," Ph.D. dissertation, University of Michigan, 1971.

[101] D. Beasley, D. R. Bull, and R. R. Martin, "An overview of genetic algorithms: Part 1, fundamentals," *University Computing*, vol. 15, no. 2, pp. 58–69, 1993.

[102] ——, "An overview of genetic algorithms: Part 2, research topics," *University Computing*, vol. 15, no. 4, pp. 170–181, 1993.

[103] J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, no. 1, pp. 122–128, 1986.

[104] J. Grefenstette and J. Baker, "How genetic algorithms work: A critical look at implicit parallelism," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, J. Schaffer, Ed., 1989.

[105] L. Davis, *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991.

[106] E. Alba and C. Cotta, "Tutorial on evolutionary optimization," Networking and Emerging Optimization, University of Málaga, Spain, 2001. [Online]. Available: http://neo.lcc.uma.es/TutorialEA/semEC/main.html

[107] H. Fröhlick, "Feature selection for support vector machines by means of genetic algorithms," Master's thesis, University of Marburg, 2002.

[108] ——, "Feature selection for support vector machines by means of genetic algorithms," in *IEEE International Conference on Tools with Artificial Intelligence*, Nov 2003.

[109] J. Sepúlveda-Sanchis, G. Camps-Vallis, E. Soria-Olivas, Salcedo-Sanz, C. Bousoño-Calzón, G. Sanz-Romero, and J. M. de la Iglesia, "Support vector machines and genetic algorithms for detecting unstable angina," in *Computer in Cardiology*, vol. 29. IEEE, 2002, pp. 413–416.

[110] K. Lee, Y. Chung, and H. Byun, "Svm-based face verification with feature set of small size," *Electronic Letters*, vol. 38, no. 15, July 2002.

[111] E. Ye and S. Cho, "GA-SVM wrapper approach for feature subset selection in keystroke dynamics identity verification," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 3. IEEE, July 2003, pp. 20–24.

[112] M. Schröder, M. Bogdan, W. Rosenstiel, T. Hinterberger, and N. Birbaumer, "Automated eeg feature selection for brain computer interfaces," in *Proceedings of the IEEE Engineering in Medicine and Biology Society Conference on Neural Engineering*. IEEE, March 2003, pp. 626–629.

[113] L. Davis, "Adapting operational problems in ga," in *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989.

[114] T. Bagchi and K. Deb, "Calibration of GA parameters: The design of experiments approach," *Computer Science and Informatics*, vol. 26, no. 3, 1996.

[115] W. G. Cochran and G. M. Cox, *Experimental Designs*, 2nd ed. Wiley, New York, 1957.

[116] C. R. Hicks and K. V. J. Turner, *Fundamental Concepts in the Design of Experiments*. Oxford University Press, Inc., 1999.

[117] D. C. Montgomery, *Design and Analysis of Experiments*, 5th ed. New York: John Wiley, 2001.

[118] R. H. Myers and D. C. Montgomery, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, 2nd ed. New York: John Wiley & Sons, 2002.

[119] L. M. Lye, "Some applications of statistical design of experiment methodology in civil engineering," in *Annual Conference of Canadian Society for Civil Engineering*, June 2003.

[120] D. J. Stewardson and R. I. Whitfield, "A demonstration of the utility of fractional experimental design for finding optimal genetic algorithm parameter settings," *Journal of the Operational Research Society*, vol. 55, no. 2, pp. 132–138, 2004.

[121] P. Poncharoen, D. J. S. C. Hicks, and P. M. Braiden, "Applying designed experiments to optimise the performance of genetic algorithms used for scheduling complex products in the capital goods industry," *Journal of Appled Statistics*, vol. 28, no. 3–4, pp. 441–455, 2001.

[122] O. Ghrayeb and H. Phojanamongkolkij, "A study of optimizing the performance of genetic algorithms using design of experiments in job-shop scheduling application," *International Journal of Industrial Engineering*, vol. 12, no. 1, pp. 37–44, 2003.

[123] B. M. Kim, Y. B. Kim, and C. Oh, "A study on the convergence of genetic algorithms," *Computers and Industrial Engineering*, vol. 33, no. 3, pp. 581–588, 1997.

[124] O. Ghrayeb, "A sensitivity analysis: Effect of genetic algorithm's parameters on solution quality," in *Proceedings of Industrial Engineering Research Conference*, May 2001.

[125] D. B. Parkinson, "Robust design employing a genetic algorithm," *Quality and Reliability Engineering International*, vol. 16, no. 3, pp. 201–208, 2000.

[126] S. Y. Yuen, H. S. Lam, and C. K. Fong, "A novel robust statistical design of the repeated genetic algorithm," in *Computer Analysis of Images and Patterns: 9th International Conference*, ser. Lecture Notes in Computer Science, W. Skarbek, Ed., vol. 2124. Springer-Verlag GmbH, 2001.

[127] J.-F. Gosselin, V. Lesage, and A. Robillard, "Population index estimate for the beluga of the st. lawrence river estuary in 2000," Fisheries and Oceans Canada," Res. Doc. 2001/049, 2001.

[128] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Prentice-Hall, 2002.

[129] D. C. Montgomery and E. A. Peck, *Introduction to Linear Regression Analysis*. New York: John Wiley & Sons, 1992.

[130] R. H. Myers, *Classical and Modern Regression with Applications*, 2nd ed. Boston: Duxbury Press, 1990.

[131] A. Law and W. D. Kelton, *Simulation Modeling and Analysis*, 2nd ed. McGraw-Hill, New York, 1991.

[132] Microsoft Office Excel 2003, "Microsoft Corporation," 2003. [Online]. Available: http://www.microsoft.com

[133] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing*. IEEE, September 1997.

[134] C.-C. Chang, C.-W. Hsu, and C.-J. Lin, "The analysis of decomposition methods for support vector machines," *IEEE Transactions on Neural Networks*, vol. 11, no. 4, 2000.

[135] W. W. Royce, "Managing the development of large software systems: Concepts and techniques," in *IEEE Western Electronic Show and Convention*, 1970, pp. 1–9, reprinted in *Proceedings of the Ninth International Conference on Software Engineering*, Pittsburgh, PA, USA, ACM Press, 1989, pp. 328–338.

[136] I. Sommerville, *Software Engineering*, 5th ed. Addison-Wesley, 1995.

[137] K. E. Wiegers, *Software Requirements*. Redmond, Washington: Microsoft Press, 1999.

[138] I. C. Society, *IEEE Recommended Practice for Software Requirements Specifications*, Institute of Electrical and Electronics Engineers, Inc., New York, 1998.

[139] A. K. Jain, *Fundamentals of Digital Image Processing*. Prentice-Hall, 1989.

[140] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Transaction on Information Theory*, pp. 179–187, Feb 1962.

[141] J. Serra, *Image Analysis and Mathematical Morphology*, 2nd ed. New York: Academic Press, 1988.

[142] S. Beucher and F. Meyer, "The morphological approach of segmentation: The watershed transformation," in *Mathematical Morphology in Image Processing*, E. Dougherty, Ed. New York: Marcel Dekker, 1992.

[143] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, June 1991.

[144] L. Vincent, "Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms," *IEEE Transactions on Image Processing*, vol. 2, no. 2, April 1993.

[145] S. Haykin, *Neural Networks – A Comprehensive Foundation*. New Jersey, USA: Prentice Hall, 1999.

[146] Y. B. Dibike, S. V. D. Solomatine, and M. B. Abbott, "Model induction with support vector machines: Introduction and applications," *ASCE Journal of Computing in Civil Engineering*, vol. 15, no. 3, pp. 208–216, July 2001.

[147] B. Schlkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, 1st ed. MIT Press, 2001.

[148] N. Cristianini and J. Shawne-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[149] M. A. Aizerman, E. M. Braverman, and L. I. Rozoner, "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.

Appendix A

Features

These are the twenty-five features used to represent segmented whale targets in feature vector **x**. The number in brackets correspond to the position of the feature-gene of a GA chromosome, reading left to right.

Area (1) This is the number pixels in the blob.

Length (2) This is the length of the blob calculated using area, a, and perimeter, p, under the assumption that $p = 2(l + b)$ and $a = l \times b$, where l = length, and b = breadth. It is used to approximate the maximum diameter of uniformly elongated blobs.

Breadth (3) This is the breadth (or width) of the blob. It is calculated in the same manner as length. It is used to approximate the minimum diameter uniformly elongated blobs.

Elongation (4) This is the ratio of length, l, to breadth, b: l/b.

Minimum Ferret Diameter (5) A Ferret diameter is calculated by measuring the diameter of the blob at a certain angle measured from the horizontal. For minimum Ferret diameter, n Ferret diameters are measured at $n/180°$ angles and the minimum diameter is selected. It is similar to breadth, but works best for non-elongated, relatively compact blobs.

Maximum Ferret Diameter (6) This feature is calculated in the same way as Ferret minimum diameter, but the maximum diameter is selected. It is similar to length.

Ferret Elongation (7) This is the ratio of maximum Ferret diameter, F, to minimum Ferret diameter, f: F/f.

Mean Ferret Diameter (8) This is the average Ferret diameter for all n angles.

Number of Holes (9) This is the number of holes in a blob. A hole is defined as a connected set of background pixels completely surrounded by the blob (foreground) pixels.

Number of Chain Pixels (10) This is the number of pixels counted by tracing clock-wise (or counter-clock-wise) along each boundary in the blob (outer boundary and boundaries along holes inside). Note that it is possible that some pixels are counted more than once, for example, along regions of the blob that are one pixel wide.

Perimeter (11) This is the total length in pixels of edges in a blob, including edges of any holes. Staircase effects along diagonals and curves are accounted for by measuring the length of the inside corner of two adjacent but offset pixels as 1.414 rather than 2.0.

Convex Perimeter (12) This is the length of the perimeter of the convex hull of a blob.

Compactness (13) This is a measure of how close pixels are packed together. It is defined as $p^2/(4\pi a)$, where p is the perimeter and a is the area. The minimum value is 1.0 for a perfect circle and increases as the shape becomes more convoluted (deviates from a circle).

Roughness (14) This is a measure of the roughness of a blob's surface. It is defined as p/c, where p is the perimeter and c is the convex perimeter. Thus smooth convex blobs have a value of 1.0 (since $p = c$) and rough blobs have values greater then 1.0.

Mean Pixel (15) This is the average pixel intensity of the blob: $(\Sigma p_i)/a$, where p_i = pixel intensity of the i^{th} pixel and a = blob area.

Pixel Standard Deviation (16) This is the standard deviation of pixel intensities in the blob:

$$\sqrt{\frac{\Sigma p_i^2 - (\Sigma p_i)^2/n}{n}} \tag{A.1}$$

where p = pixel intensity of the i^{th} pixel and n = number of blob pixels.

Minimum Pixel (17) This is the minimum pixel intensity the blob.

Maximum Pixel (18) This is the maximum pixel intensity in the blob.

Moment Invariant 1 (19) This is defined as:

$$\phi_1 = \eta_{20} + \eta_{02} \tag{A.2}$$

where n_{pq} is the normalized central moment (see Section A.1 for more details).

Moment Invariant 2 (20) This is defined as:

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{A.3}$$

Moment Invariant 3 (21) This is defined as:

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{A.4}$$

Moment Invariant 4 (22) This is defined as:

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{A.5}$$

Moment Invariant 5 (23) This is defined as:

$$\begin{aligned} \phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\ &\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \end{aligned} \tag{A.6}$$

Moment Invariant 6 (24) This is defined as:

$$\phi_6 = (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{A.7}$$

Moment Invariant 7 (25) This is defined as:

$$\begin{aligned} \phi_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\ &\quad + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \end{aligned} \tag{A.8}$$

A.1 Hu Moments

The seven moment invariants, $\phi_i, i = 1 \cdots 7$, above are often referred to as Hu moments [140]. These types of moments are invariant to translation, rotation, and scale of the object. The Hu moments are derived from the second and third normalized central moments [128]. A moment of order $(p + q)$ is defined as

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y) \tag{A.9}$$

where $p, q = 0, 1, 2, \ldots$ and $f(x, y)$ is the gray scale value at location (x, y). A central moment of order $(p + q)$ is defined as

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y) \tag{A.10}$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}} \text{ and } \bar{y} = \frac{m_{01}}{m_{00}}$$

Finally, a normalized central moment η_{pq} is defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\rho}} \tag{A.11}$$

where

$$\rho = \frac{p+q}{2} + 1$$

Appendix B

CCD: the Central Composite Design

This appendix presents a brief overview of the central composite design (CDD). For a more detailed treatment see [117].

A factorial design runs experiments on all possible combinations of levels, which is in strong contrast to OFAT (one factor at a time), which modifies levels one factor at a time. A common and very efficient (minimum number of runs) factorial design is the 2^k factorial, where k factors are varied over two levels, high ($+1$) and low (-1). For $k = 2$, the design forms a square with experiments run at each corner; for $k = 3$, the design forms a cube with experiments run on each corner (Figure B.1).

The actual level values are chosen by the designer, but should be far enough apart to allow some variation in response and decrease the change that noise will overwhelm the response, but not too far apart so a model cannot be adequately fit. Usually some standard operating conditions of the process being studied acts as the center level (level 0) and levels closer to the min and max range are the -1 and $+1$ levels, respectively.

The 2^k design is most often used and forms the basis of more complex designs. In addition, the experimenter can run experiments using this design first, and augment it later if necessary to run more advanced designs. This allows the experiment to be run in stages, which is particularly important for executing a sequential strategy to minimize the number of runs to fit a model [117].

Adding one or more runs at the center point of the levels (center of the cube for $k = 3$) allows identification of curvature in the response and estimates of error to be made. Furthermore, if curvature may be significant, runs at axial points (points on the coordinate axes) can also be added to form a center composite design (CCD; Figure B.2a). This allows fitting of second-order models. If the axial points are added at distance $\alpha \neq \pm 1$ from the center, each factor must be varied over five levels. Alternatively, if the points are chosen at $\alpha = 1$, then we have a face-centered CDD (Figure B.2b) that has most of the benefits of the CCD, but only three levels are required. This is particularly useful if it is not desirable or impossible to have levels at $\alpha > 1$. In general, the (face-centered) CCD is a very efficient design, especially for fitting second-order models with $k \geq 3$ factors. It is also one of the most important designs in the process of optimization.

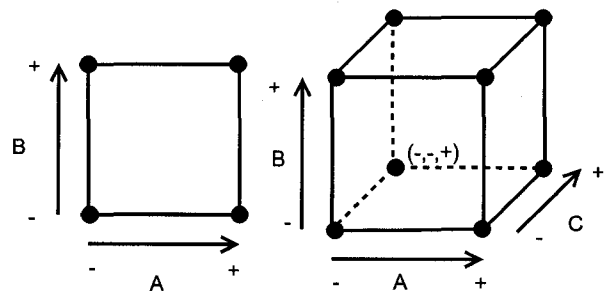


Figure B.1: Illustration of 2^2 and 2^3 DOE designs. Circles represent points at which the experiment is run. Factors are labeled A, B, and C. $-$ represents a low level and $+$ represents a high level. So, at point $(-,-,+)$, the experiment is run with A and B at low levels and C at the high level.

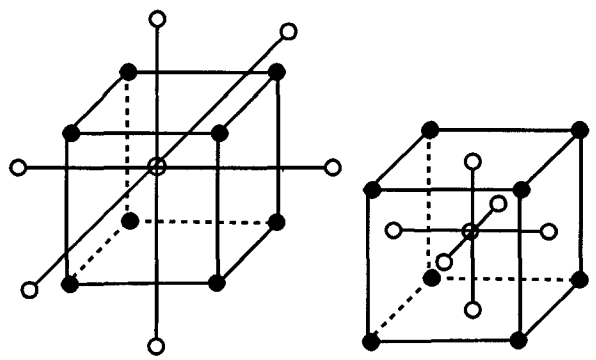


Figure B.2: Illustration of 2^3 CCD and 2^3 face-centered CCD. The center point is level $(0,0,0)$.

Appendix C

DOE Analysis for Model Assumptions

C.1 Introduction

When building response models using DOE, [118] states that "it is always necessary to (1) examine the fitted model to ensure that it provides an adequate approximation of the true system, and (2) verify that none of the least squares regression (ANOVA) assumptions are violated". These assumptions are that the residuals, $e_i = y_i - \hat{y}_1$, are normally and independently distributed with mean zero and that they have a constant variance. If these assumptions hold, the residuals will contain no obvious patterns. The easiest method to check these assumptions is through graphical analysis [117]. This is the approach taken in this work. What follows is the results of analysis of residuals for model assumptions (normality of residuals, constant variance, and independence) for each of the models developed in Chapter 5.

C.1.1 2QD

Figures C.1a to C.5b shows the results of analysis of residuals for model assumptions (normality of residuals, constant variance, and independence). Figure C.1a shows a good straight line fit. This indicates the model's residuals are normal. There is no systematic patterns in Figure C.1b, indicating a constant variance. There is also a random pattern in Figure C.2a, indicating that all systematic run/time related effects are accounted for in the model, and Figures C.2b to C.4a do not show any systematic pattern, suggesting there is no systematic contribution of an independent factor that is not accounted for by the model. This indicates the independence assumption is satisfied. No significant outliers are present in Figures C.4b, C.5a, and C.5b.

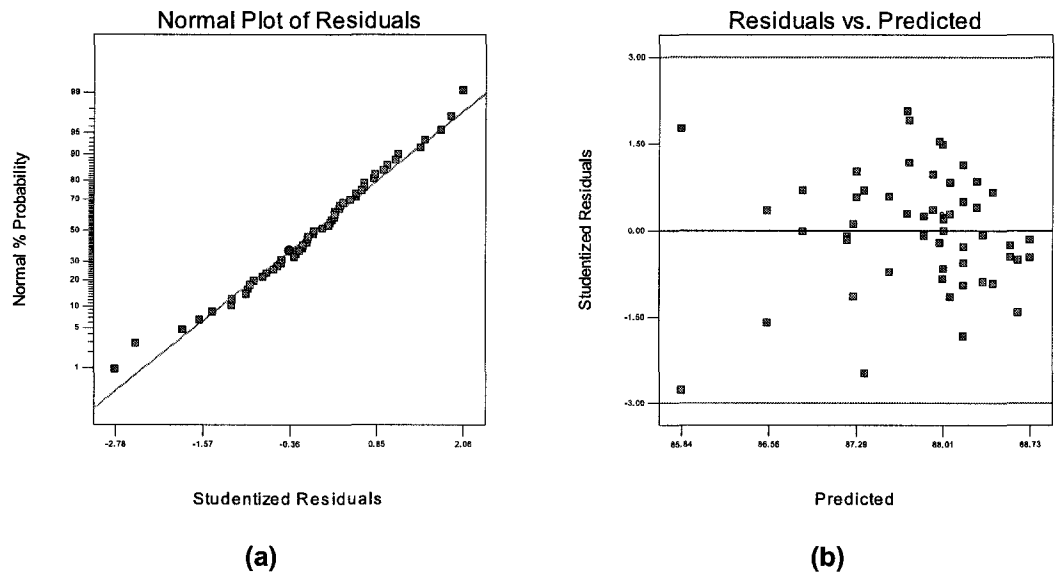


Figure C.1: (a) Normal probability plot of 2QD y_1 residuals. (b) Plot of 2QD y_1 residuals versus $\hat{y}_1$.

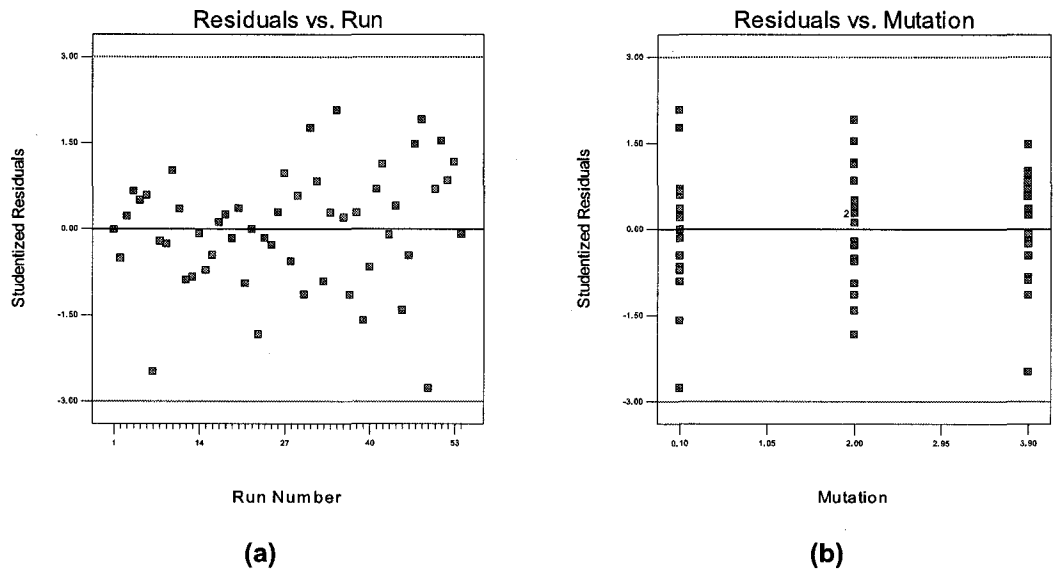
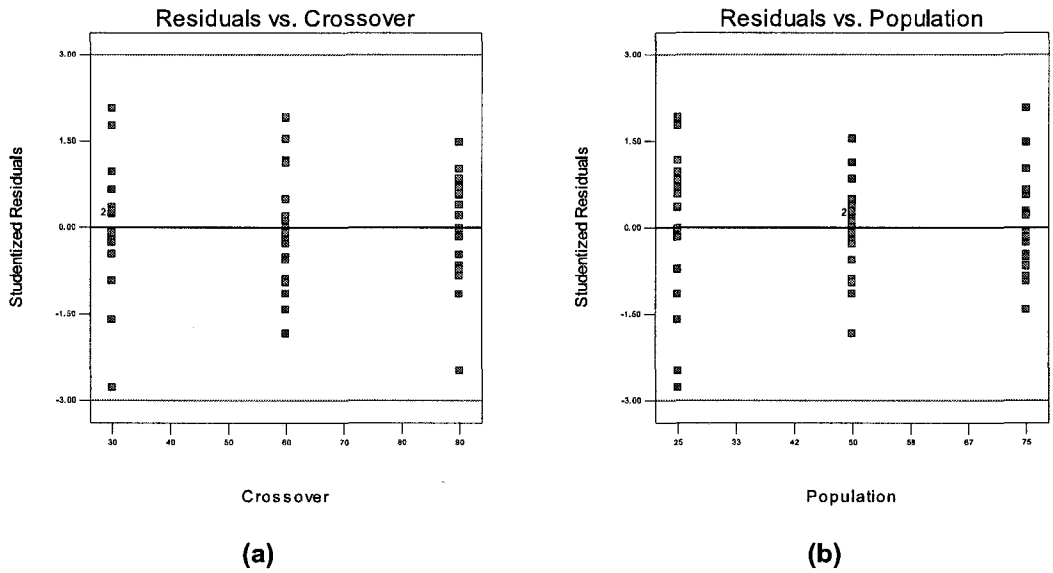
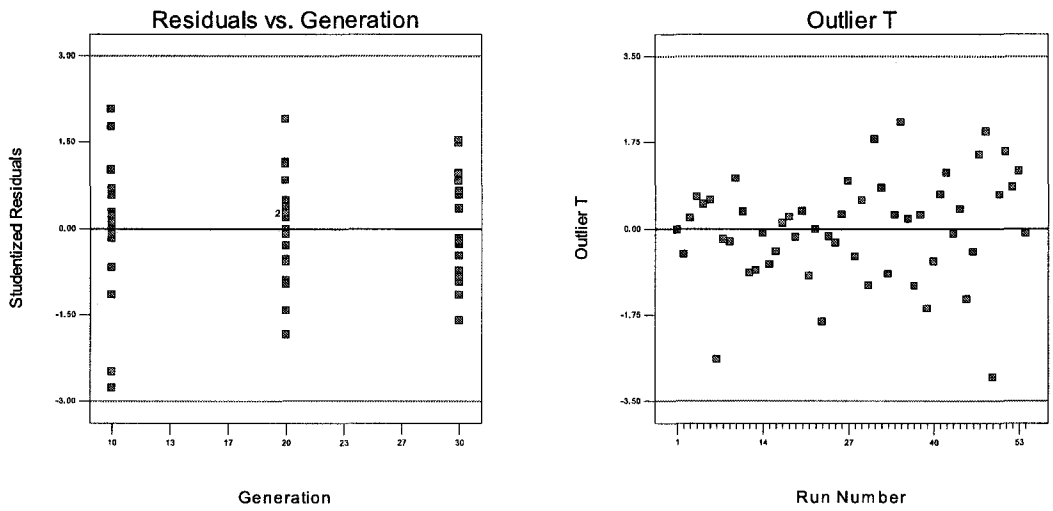


Figure C.2: (a) Plot of 2QD y_1 residuals versus run. (b) Plot of 2QD y_1 residuals versus factor A.



(a) (b)

Figure C.3: (a) Plot of 2QD y_1 residuals versus factor B. (b) Plot of 2QD y_1 residuals versus factor C.



(a) (b)

Figure C.4: (a) Plot of 2QD y_1 residuals versus factor D. (b) Outlier T plot to check for 2QD y_1 outliers.

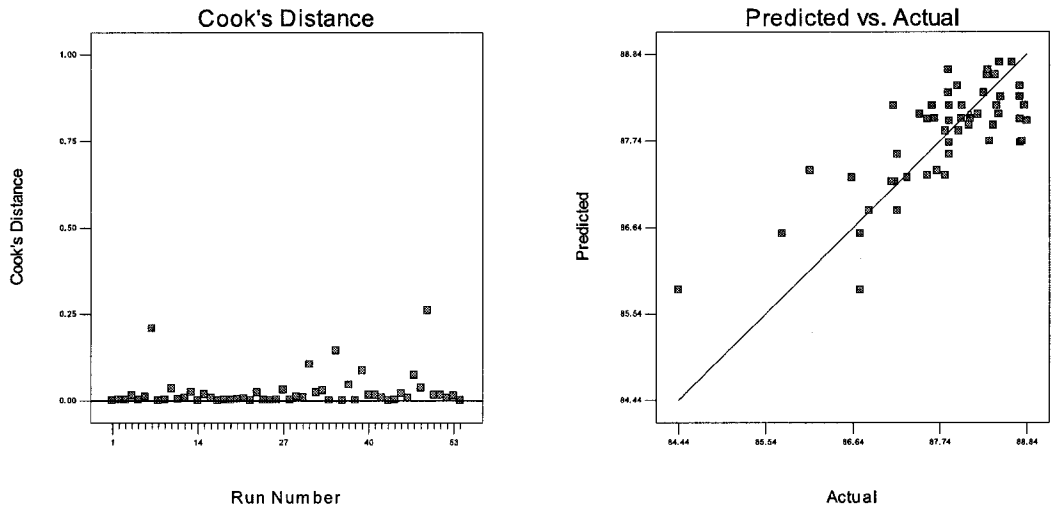


Figure C.5: (a) Cook's distance plot for 2QD y_1. (b) Plot of actual 2QD y_1 versus $\hat{y}_1$.

C.1.2 3QD

Figures C.6a to C.10b shows the results of analysis of residuals for model assumptions. Figure C.6a shows a reasonable straight line fit. This indicates the model's residuals are normal. There is no systematic pattern in Figure C.6b, indicating a constant variance. There is also a random pattern in Figure C.7a, indicating that all systematic run/time related effects are accounted for in the model. The lack of a pattern in Figures C.7b to C.9a suggests there is no systematic contribution of an independent factor that is not accounted for by the model. No significant outliers are present in Figures C.9b, C.10a and C.10b.

C.1.3 2SVM

Figures C.11a to C.15b shows the results of analysis of residuals for model assumptions. Figure C.11a shows a good straight line fit indicating the model's residuals are normally distributed. There is no systematic pattern in Figure C.11b, indicating a constant variance. There is also a random pattern in Figure C.12a, indicating that all systematic run/time related effects are accounted for in the model. Figures C.12b to C.14a do not show any systematic pattern, indicating there is no systematic contribution of an independent factor that is not accounted for by the model. Figures C.14b, C.15a and C.15b shows there are no significant outliers.

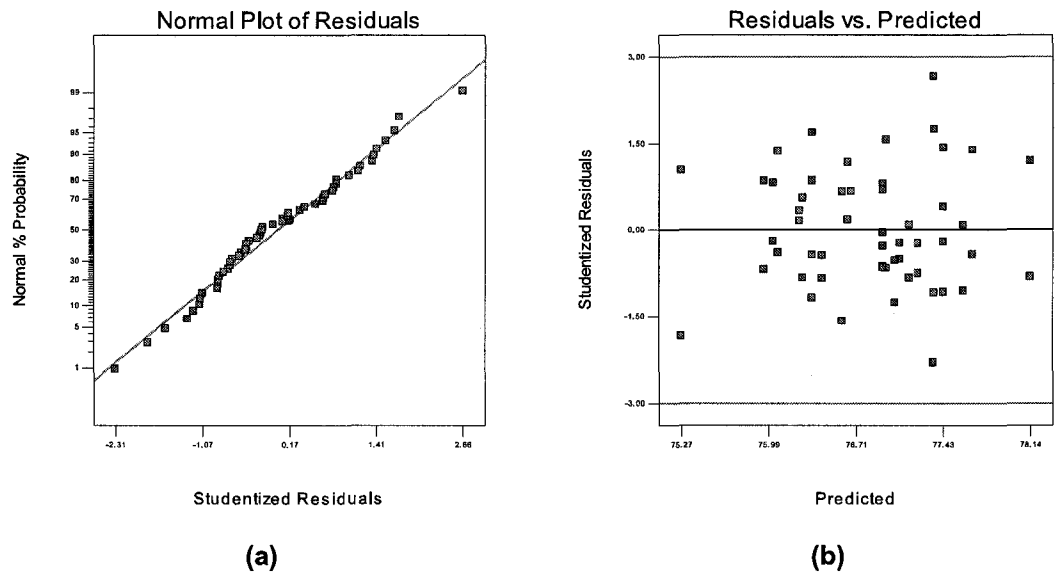


Figure C.6: (a) Normal probability plot of 3QD y_1 residuals. (b) Plot of 3QD y_1 residuals versus $\hat{y}_1$.

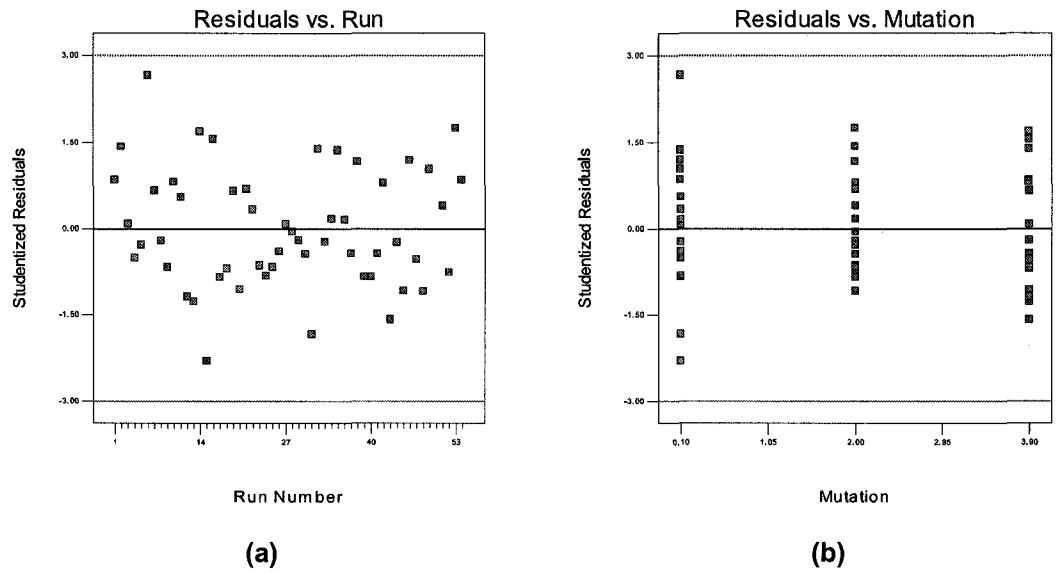


Figure C.7: (a) Plot of 3QD y_1 residuals versus run. (b) Plot of 3QD y_1 residuals versus factor A.

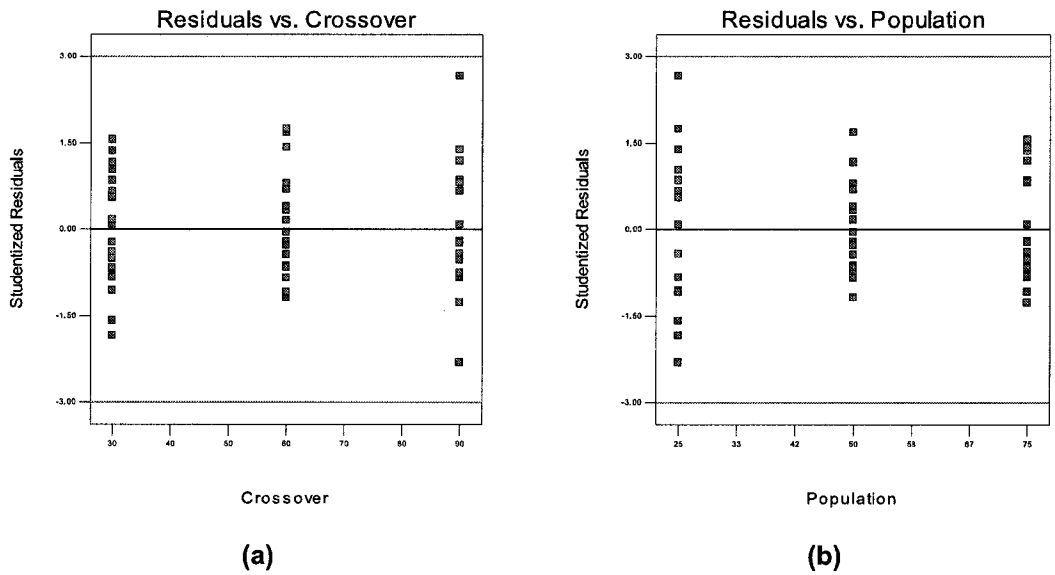


Figure C.8: (a) Plot of 3QD y_1 residuals versus factor B. (b) Plot of 3QD y_1 residuals versus factor C.

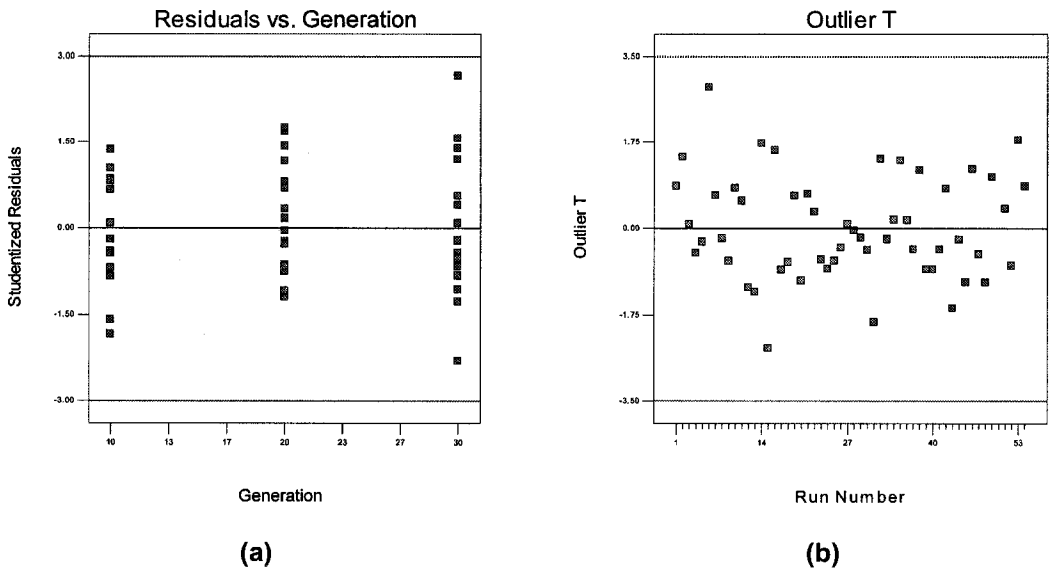


Figure C.9: (a) Plot of 3QD y_1 residuals versus factor D. (b) Outlier T plot to check for 3QD y_1 outliers.

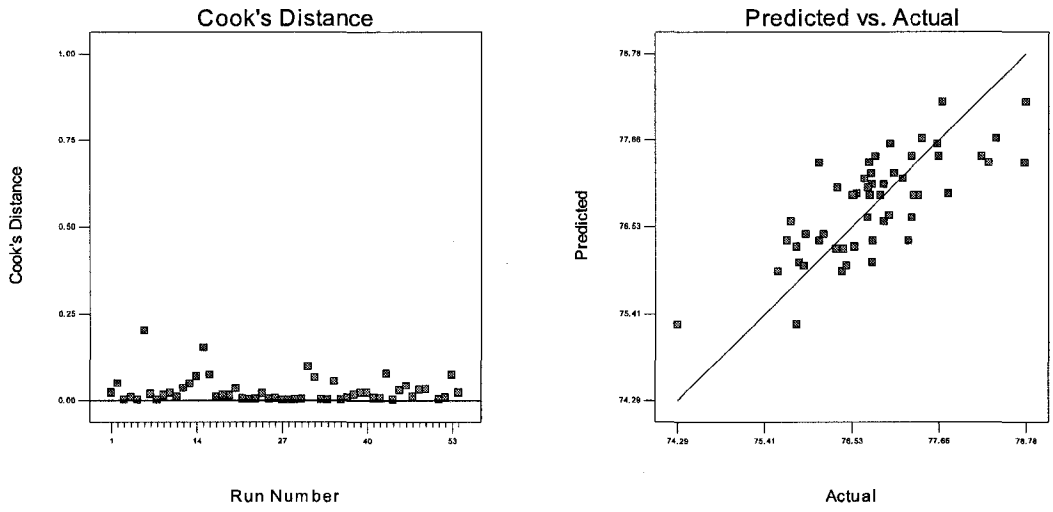


Figure C.10: (a) Cook's distance plot for 3QD y_1. (b) Plot of actual 3QD y_1 versus $\hat{y}_1$.

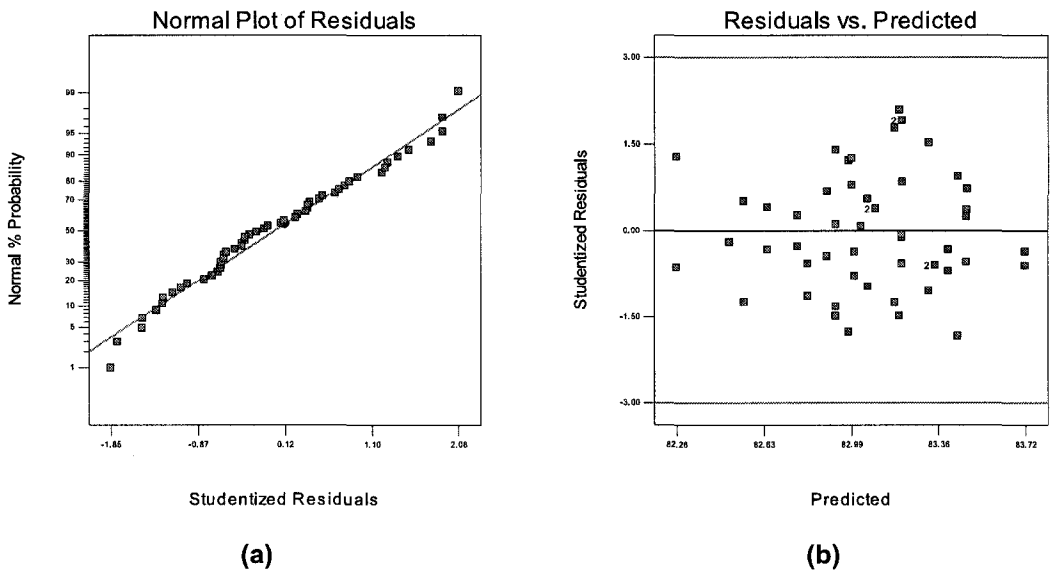
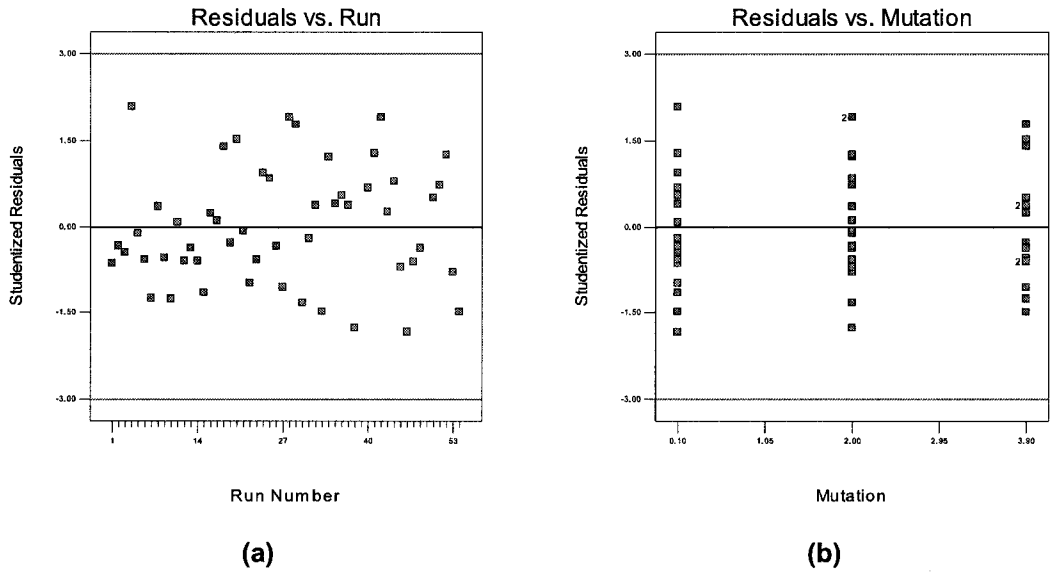
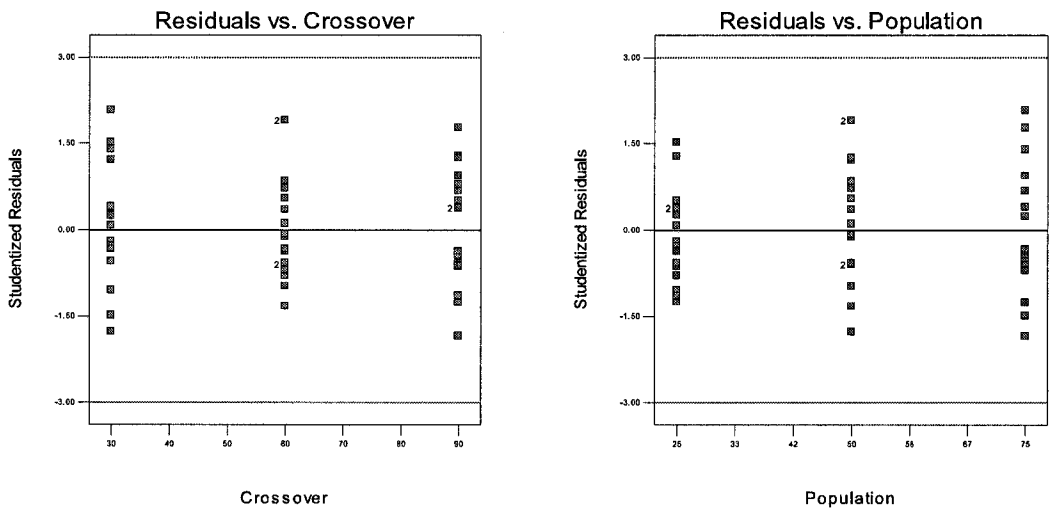


Figure C.11: (a) Normal probability plot of 2SVM y_1 residuals. (b) Plot of 2SVM y_1 residuals versus $\hat{y}_1$.



(a) (b)

Figure C.12: (a) Plot of 2SVM y_1 residuals versus run. (b) Plot of 2SVM y_1 residuals versus factor A.



(a) (b)

Figure C.13: (a) Plot of 2SVM y_1 residuals versus factor B. (b) Plot of 2SVM y_1 residuals versus factor C.

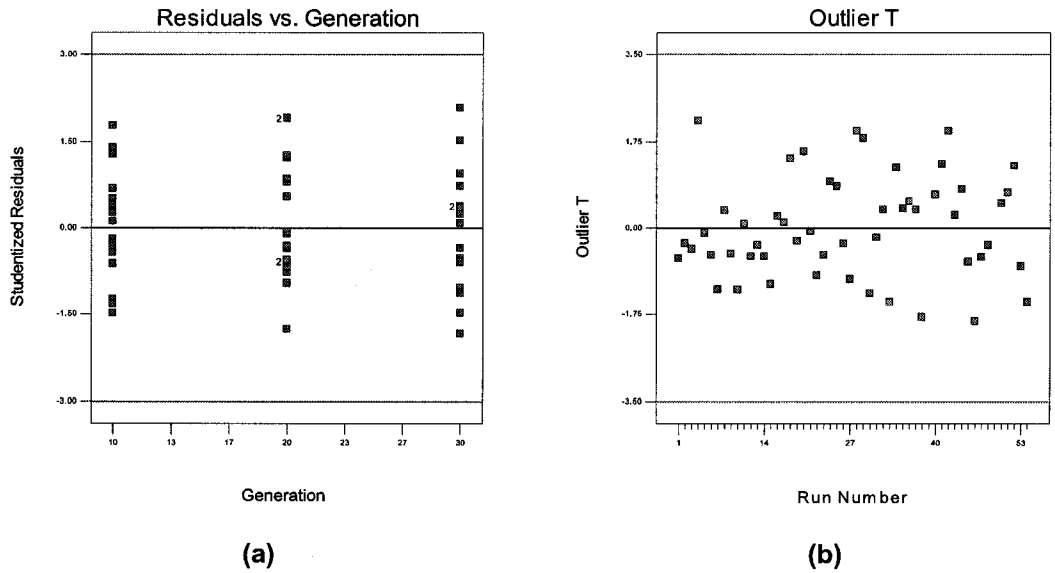


Figure C.14: (a) Plot of 2SVM y_1 residuals versus factor D. (b) Outlier T plot to check for 2SVM y_1 outliers.

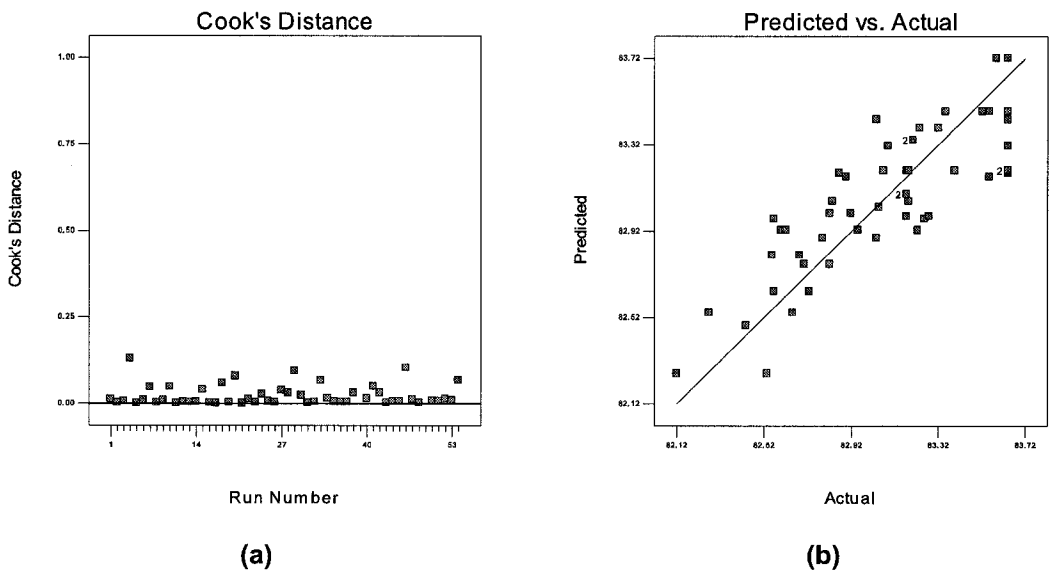


Figure C.15: (a) Cook's distance plot for 2SVM y_1. (b) Plot of actual 2SVM y_1 versus $\hat{y}_1$.

C.1.4 3SVM

Figures C.16a to C.20b shows the results of analysis of residuals for model assumptions. The model's residuals are normally distributed as indicated by the good straight line fit shown in Figure C.16a. There is no systematic pattern in Figure C.16b, indicating a constant variance. There is also a random pattern in Figure C.17a, indicating that all systematic run/time related effects are accounted for in the model. Figures C.17b to C.19a do not show any systematic pattern suggesting there is no systematic contribution of an independent factor that is not accounted for by the model. Figures C.19b, C.20a and C.20b does not indicate any outliers.

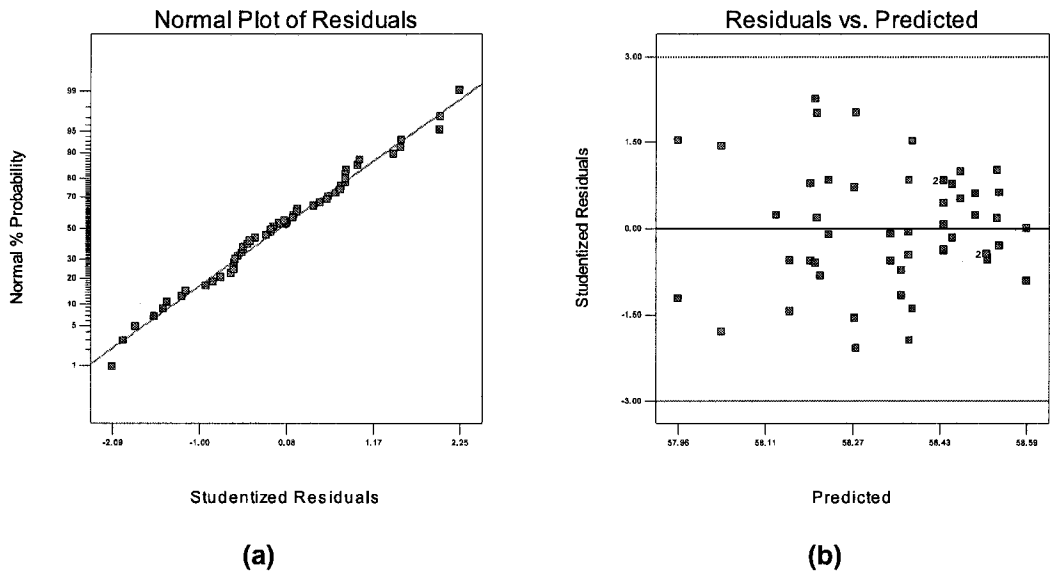


Figure C.16: (a) Normal probability plot of 3SVM y_1 residuals. (b) Plot of 3SVM y_1 residuals versus $\hat{y}_1$.

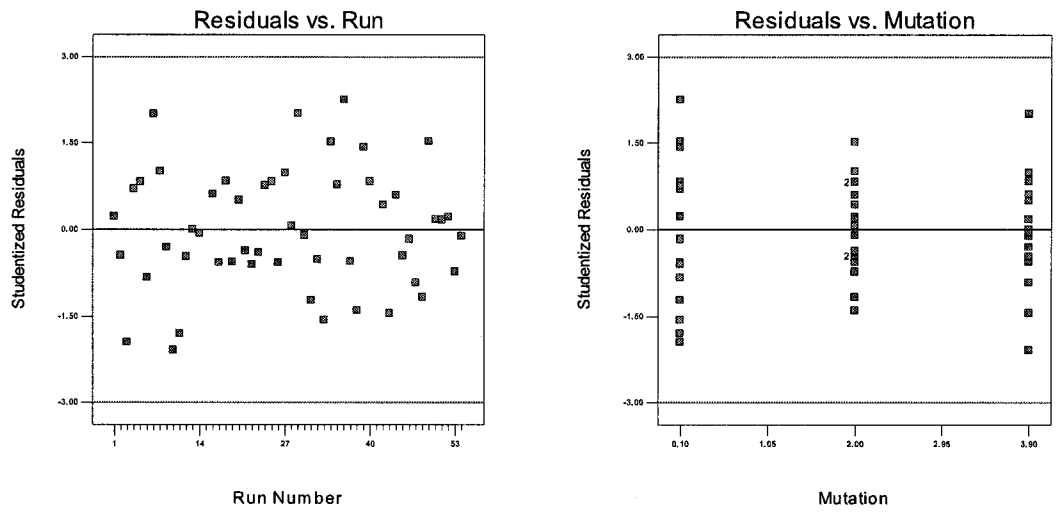


Figure C.17: (a) Plot of 3SVM y_1 residuals versus run. (b) Plot of 3SVM y_1 residuals versus factor A.

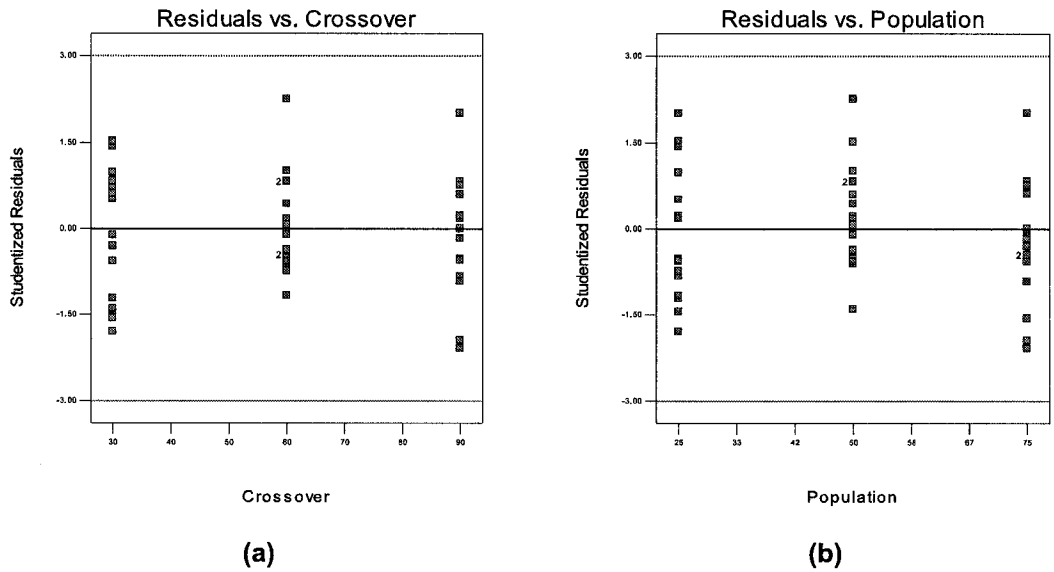
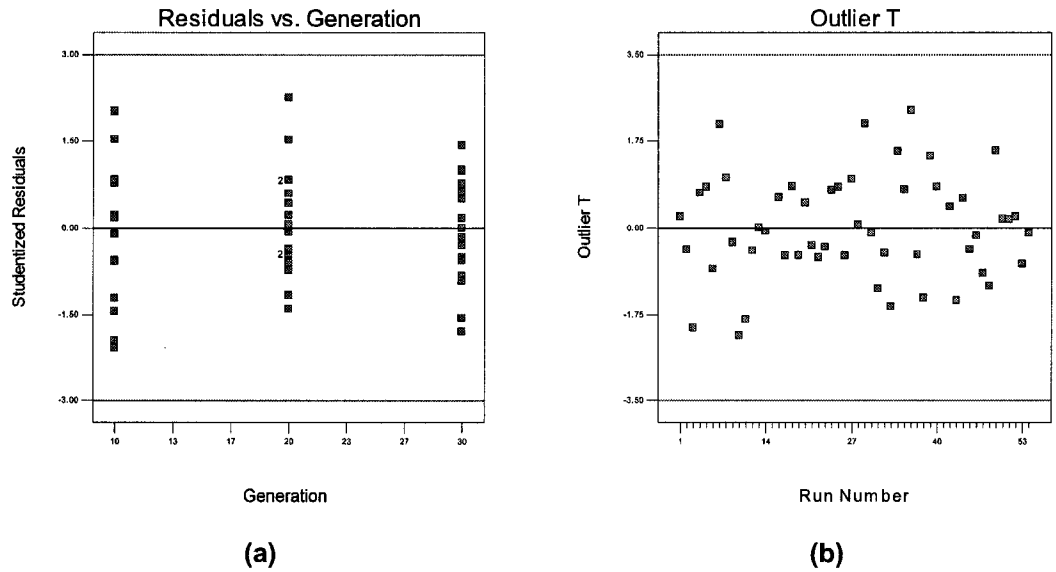
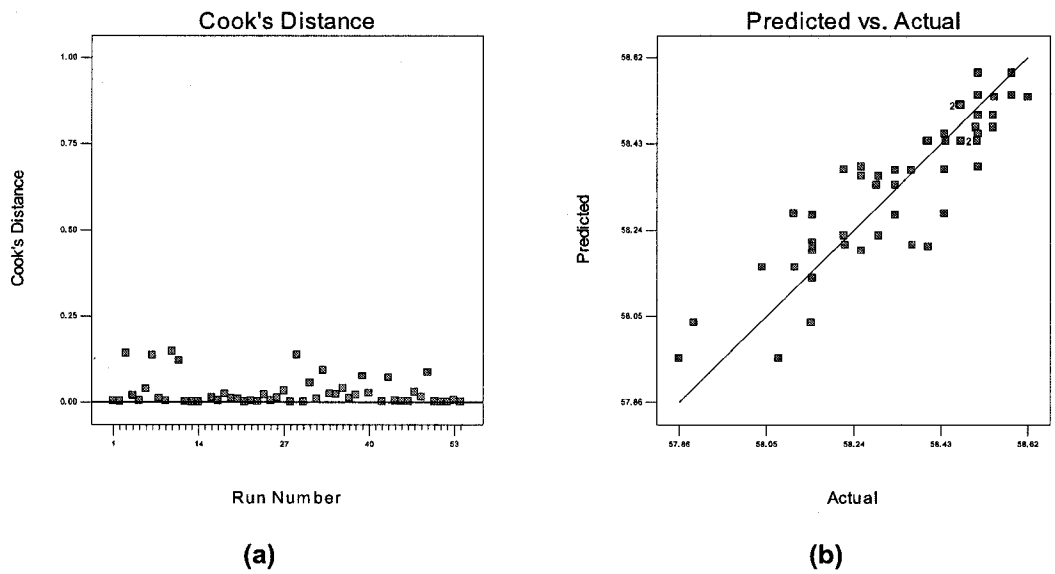


Figure C.18: (a) Plot of 3SVM y_1 residuals versus factor B. (b) Plot of 3SVM y_1 residuals versus factor C.



(a) (b)

Figure C.19: (a) Plot of 3SVM y_1 residuals versus factor D. (b) Outlier T plot to check for 3SVM y_1 outliers.



(a) (b)

Figure C.20: (a) Cook's distance plot for 3SVM y_1. (b) Plot of actual 3SVM y_1 versus $\hat{y}_1$.

Appendix D

Watershed Segmentation

Region based segmentation attempts to segment regions directly rather than relying on finding edges or thresholds. One common region-based segmentation algorithm is known as watershed segmentation. Watershed segmentation is best explained [128] by envisioning a gray scale image as a 3-D topography (map) where the gray levels represent elevations. In such a view, areas of constant gray level represent plains, areas of low gray levels relative to the surrounding area represent catchment basins (regional minima), and regions of high gray levels relative to the surrounding levels are peaks (ridges, hills, or mountains). As in the real world, water dropped anywhere onto the surface of a catchment basin, even on its sides, will flow into that basin. All points where water will flow into the basin is know as the watershed for that basin. On the other hand, if the water is dropped on the crest of some peak or plain, the water is equally likely to flow into any of the adjacent catchment basins. Such areas are known as watershed lines because they separate the adjacent watersheds.

Now suppose it uniformly rained over the entire topography. The water level in each of the basins will start to rise expanding the area covered by water. At some point, when the water level reaches the height of the surroundings peaks, the water in a basin will began to overflow into the adjacent basins. To prevent this a dam is constructed along the peak or in the middle of a plain between the effected watersheds such that it is impossible for the watersheds to merge. Eventually the water in all the watersheds will reach a level where only the dams remain above water, completely enclosing each watershed. Hence the entire topography is segmented into regions of water separated by dams. In image segmentation terms, the watersheds filled with water represent the segmented regions and the dams represent the lines separating the regions.

The above description is illustrated in Figure D.1, which shows a cross section of the local image topography around two objects. The objects are represented by basins, which correspond to low gray levels. Water level lines are shown at four intervals during the flooding stage: at water level 1, no dam is constructed; at water level 2, dam 2 is constructed; at water level 3, dam 1 is constructed; and at water level 4, dam 3 is constructed. Note that dam 3 is constructed in the center of a plain between two basins while the other dams are constructed on peaks. When the algorithm is complete (flooding is done), the dams represent watershed lines that separate the segmented regions as shown.

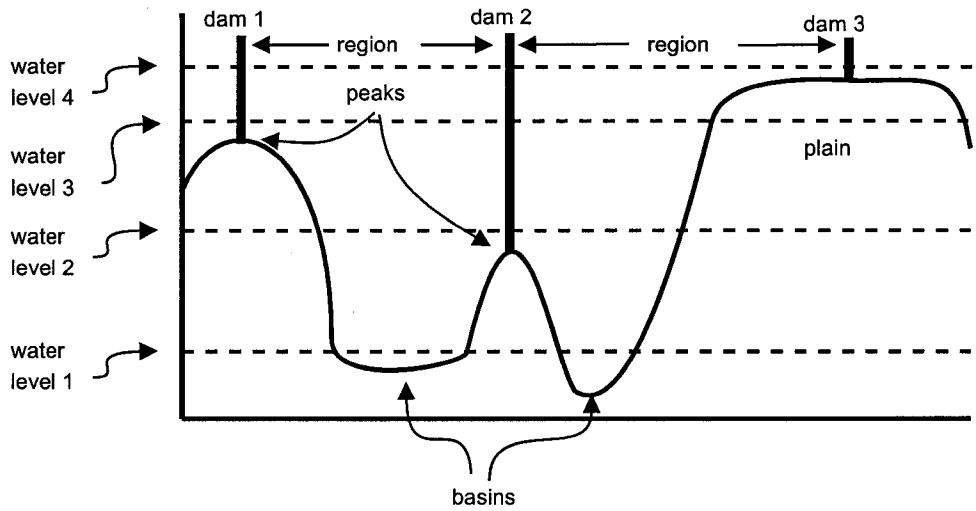


Figure D.1: Illustration of the watershed algorithm. The graphic is a cross-section through the local topography around two objects represented by basins. The vertical axis is gray level. The horizontal axis is location in the image.

The above description is very textual and general. For a formal mathematical treatment see [128]. Early descriptions of the watershed is provided by [141] and [142]. Efficient algorithms for watershed implementation for binary and gray scale images is presented by [143] and [144].

Appendix E

Pattern Classification

E.1 Bayesian Minimum Error-Rate Quadratic Discriminate

The idea of a Bayesian classifier is based on Bayes formula (for continuous, multivariate features)

$$P(\mathscr{C}_i|\mathbf{x}) = \frac{p(\mathbf{x}|\mathscr{C}_i)P(\mathscr{C}_i)}{p(\mathbf{x})} \tag{E.1}$$

where $P(\mathscr{C}_i)$ is the prior (*a priori*) probability, $P(\mathscr{C}_i|\mathbf{x})$ is the posterior (*a posteriori*) probability, and $p(\mathbf{x}|\mathscr{C}_i)$ is the likelihood of $\mathscr{C}_i$ with respect to $\mathbf{x}$. $p(\mathbf{x})$, defined as

$$p(\mathbf{x}) = \sum_{j=1}^{n} p(\mathbf{x}|\mathscr{C}_i)P(\mathscr{C}_i) \tag{E.2}$$

is the evidence. It acts as a scale factor that ensures the posterior probabilities sum to one. For classification, this term is usually dropped. The basic approach is to minimize the probability of error by choosing a class that maximizes the posterior probability $P(\mathscr{C}_i|\mathbf{x})$. This leads to Bayes decision rule

$$\text{if } P(\mathscr{C}_i|\mathbf{x}) > P(\mathscr{C}_j|\mathbf{x}) \text{ for all } j \neq i \text{ classify as } \mathscr{C}_i \tag{E.3}$$

which by eliminating the scale factor becomes

$$\text{if } p(\mathbf{x}|\mathscr{C}_i)P(\mathscr{C}_i) > p(\mathbf{x}|\mathscr{C}_j)P(\mathscr{C}_j) \text{ for all } j \neq i \text{ classify as } \mathscr{C}_i \tag{E.4}$$

A classifier for c classes can be represented as a set of discriminate functions $g_i(\mathbf{x})$, $i = 0, 1, \cdots, c$. The classifier operates by selecting the class with the largest discriminate, using the rule

$$\text{if } g_i(\mathbf{x}) > g_j(\mathbf{x}) \text{ for all } j \neq i \text{ classify as } \mathscr{C}_i \tag{E.5}$$

where $\mathbf{x}$ is the feature vector for the object to be classified. Surfaces in feature space where

$g_i(\mathbf{x}) = g_j(\mathbf{x})$ are called the decision surface between $\mathscr{C}_i$ and $\mathscr{C}_j$.

Armed with the idea of a discriminate, we can write a Bayesian minimum error rate discriminate as

$$g_i(\mathbf{x}) = P(\mathscr{C}_i|\mathbf{x}) = p(\mathbf{x}|\mathscr{C}_i)P(\mathscr{C}_i) \tag{E.6}$$

If f is a monotonically increasing function, $g_i(\mathbf{x})$ can be replaced with $f(g_i(\mathbf{x}))$ without changing the classification results. Hence, we can reformulate the last equation as

$$g_i(\mathbf{x}) = \ln p(\mathbf{x}|\mathscr{C}_i) + \ln P(\mathscr{C}_i) \tag{E.7}$$

$P(\mathscr{C}_i)$ is usually chosen manually based on prior knowledge of the problem, and $p(\mathbf{x}|\mathscr{C}_i)$ is often modeled as Gaussian distribution, namely

$$p(\mathbf{x}|\mathscr{C}_i) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] \tag{E.8}$$

where $\mathbf{x}$ is the d-dimension feature vector (hence d is the number of features), $\boldsymbol{\mu}$ is the d-dimension mean vector, $\boldsymbol{\Sigma}$ is the $d \times d$ covariance matrix, and $|\boldsymbol{\Sigma}|$ is the determinate of $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}^{-1}$ is its inverse. Because the normal distribution is completely specified by $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we say $p(\mathbf{x}) \approx N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The final Bayesian minimum error rate quadratic discriminate function for $\mathscr{C}_i$ can be derived by substituting Equation E.8 into Equation E.7, resulting in

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(\mathscr{C}_i) \tag{E.9}$$

This function provides hyperquadric (hence curves of various shapes including linear) decision boundaries.

For a Guassian distribution, the unknowns $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ must be estimated. One common method is maximum-likelihood. It can be proven [30] that the maximum-likelihood estimation of the population mean is the arithmetic average of the training samples, called the sample mean

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k \tag{E.10}$$

It can also be shown that the maximum-likelihood estimate of the covariance matrix is the arithmetic average of the n matrices $(\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^T$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^T \tag{E.11}$$

These are important results for pattern recognition because both are relatively straight forward to calculate from the training data. Note that Equation E.11 is biased [30]. An ele-

mentary unbiased estimate is called the sample covariance matrix

$$\mathbf{C} = \frac{1}{n-1}\sum_{k=1}^{n}(\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^T \tag{E.12}$$

The above formulations are described for continuous features, i.e. $\mathbf{x} \in \mathbb{R}^d$. However, the same formulations apply to discrete features except the density $p(\mathbf{x}|\mathscr{C}_i)$ is replaced by probabilities $P(\mathbf{x}|\mathscr{C}_i)$

E.2 Support Vector Machines

Following is a brief overview of theory behind SVMs from the point of view of pattern recognition. Unless otherwise noted, the theory is a combination and reformulation of material extracted from the more rigorous discussions in [49, 145] and more tutorial-like descriptions in [45, 54, 146]. For more information on statistical learning theory, SVMs, and kernel machines see [51, 59, 147]. For a detailed treatment of SVMs for pattern recognition see [64, 148]. For a more introductory yet extensive review see [54].

E.2.1 SVMs for Linearly Separable Patterns: Hard Margin SVMs

In this section, we focus on the simple problem of two-classes of linearly separable patterns. The SVM theory presented is the basis for dealing with more complex problems later.

Given a set of training data $\mathbb{T} = \{\mathbf{x}_i, y_i\}$, $i = 1 \ldots n$, where $y_i \in \{+1, -1\}$ is the class label, $\mathbf{x}_i \in \mathbb{R}^d$ is an input sample vector, d is the dimensionality of the sample space (the size of the vectors), and n is the number of samples in the data set. Assume that class $y_i = +1$ is linearly separable from $y_i = -1$, then the hyperplane separating the two classes is

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{E.13}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the adjustable weight vector normal to the hyperplane and $b \in \mathbb{R}$ is a bias. Both $\mathbf{w}$ and b are induced from the training set T such that the following conditions are satisfied

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_i + b \geq 0 \quad & \text{for } y_i = +1 \\ \mathbf{w} \cdot \mathbf{x}_i + b < 0 \quad & \text{for } y_i = -1 \end{aligned} \tag{E.14}$$

Define d_+ and d_- to be the distance from the hyperplane to the closest sample point in classes $+1$ and -1 respectively, then the margin of separation is $\rho = d_+ + d_-$. There is an infinite number of hyperplanes that separate the two classes, as illustrated in Figure E.1, but there is only one that maximizes ρ, it is referred to as the optimal separating hyperplane, defined as

$$\mathbf{w}_0 \cdot \mathbf{x} + b_0 = 0 \tag{E.15}$$

where $\mathbf{w}_0$ and b_0 are the optimal values of $\mathbf{w}$ and b in Equation E.13. A hyperplane defines a

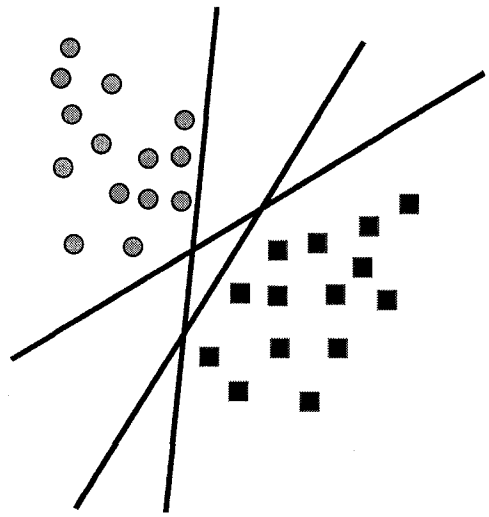


Figure E.1: Illustration of three out of the infinite number of hyperplanes separating two patterns.

multidimensional linear decision surface in input space. The optimal separating hyperplane is the hyperplane that has the maximal margin of separation between classes, has the lowest capacity, and minimizes the bounds on the actual risk [146, 48].

Using basic linear algebra it can be shown that the perpendicular distance from the optimal hyperplane to the origin is given by

$$\frac{|b_0|}{\|\mathbf{w}_0\|} \tag{E.16}$$

where $\|\mathbf{w}\|$ is the Euclidean norm of $\mathbf{w}$ (Figure E.2). If we rescale $\mathbf{w}_0$ and b_0 such that

$$\begin{aligned} \mathbf{w}_0 \cdot \mathbf{x}_i + b_0 &\geq 1 \quad \text{for } y_i = +1 \\ \mathbf{w}_0 \cdot \mathbf{x}_i + b_0 &\leq 1 \quad \text{for } y_i = -1 \end{aligned} \tag{E.17}$$

and combine into canonical form

$$y_i(\mathbf{w}_0 \cdot \mathbf{x}_i + b_0) \geq 1 \tag{E.18}$$

then all points $\mathbf{x}_s$ where $|\mathbf{w}_0 \cdot \mathbf{x}_i + b_0| = 1$ lie on the supporting hyperplanes H_+ and H_- and are called support vectors. The support planes are

$$\begin{aligned} H_+ &: \mathbf{w}_0 \cdot \mathbf{x}_i + b_0 = 1 \\ H_- &: \mathbf{w}_0 \cdot \mathbf{x}_i + b_0 = -1 \end{aligned} \tag{E.19}$$

The support hyperplanes are parallel to the optimal separating hyperplane (have the same normal) and no training samples fall in between them.

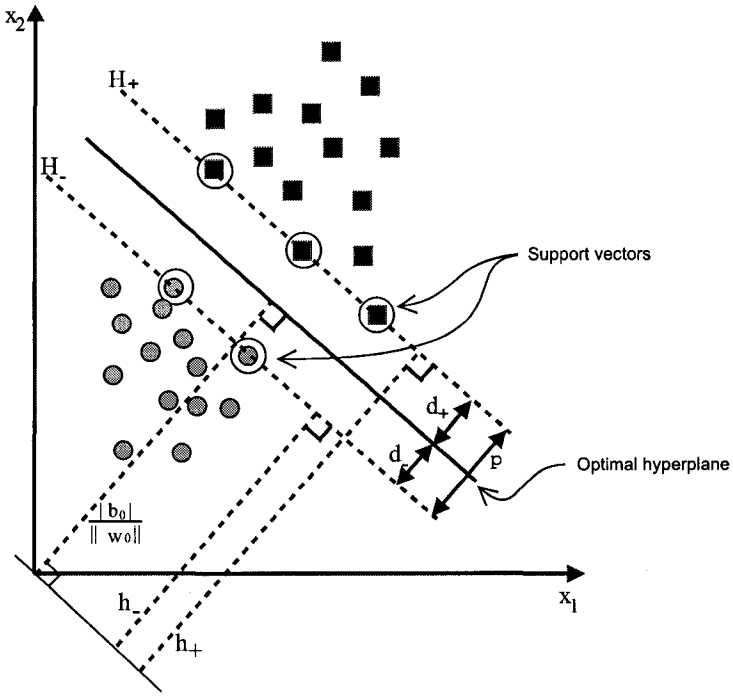


Figure E.2: An optimal separating hyperplane for two linearly separable, 2-dimensional, patterns, represented by circles and squares. The points circled along the lines H_- and H_+ are the support vectors.

With reference to Figure E.2 and some simple algebra, it is easy to see that

$$d_- = d+ = \frac{1}{\|\mathbf{w}_0\|} \tag{E.20}$$

hence

$$\rho = d_- + d_+ = \frac{1}{\|\mathbf{w}_0\|} + \frac{1}{\|\mathbf{w}_0\|} = \frac{2}{\|\mathbf{w}_0\|} \tag{E.21}$$

Then, to find the hyperplane which gives the maximum margin (optimal separating hyperplane), we minimize $\|\mathbf{w}\|$ under the constraints imposed by Equation E.18. Hence, to optimize ρ, it can be shown that we minimize the cost function

$$\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \tag{E.22}$$

in the quadratic program (QP optimization problem),

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \end{aligned} \tag{E.23}$$

The QP can be solved using the method of Lagrange multipliers. To do so we reformu-

late Equation E.23 to the equivalent Lagrangian function

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=0}^{n} \alpha_i (y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1) \tag{E.24}$$

where $\alpha_i \geq 0$ are Lagrange multipliers. The solution is found by minimizing $L(\mathbf{w}, b, \alpha)$ with respect to w and b, therefore we require

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}} &= \mathbf{0} \\ \frac{\partial L}{\partial b} &= 0 \end{aligned} \tag{E.25}$$

Application of Equation E.25 to Equation E.24 yields

$$\begin{aligned} \sum_{i=1}^{n} \alpha_i y_i &= 0 \\ \mathbf{w} &= \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \end{aligned} \tag{E.26}$$

Substituting Equation E.26 into an expanded and rearranged version of Equation E.24 and simplifying yields the dual form of the Lagrangian

$$D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{E.27}$$

The corresponding dual QP is

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_i y_{=} 0 \\ & \alpha_i \geq 0 \ , i = 1 \ldots n \end{aligned} \tag{E.28}$$

The duality of the expressed Lagrangian equations and QP problems above has the property that the maximum of D, subject to its constraints, occurs at the same w, b, and α as the minimum of L, subject to its constraints [49]. Thus we can find a solution by minimizing L or maximizing D. However, it is preferable to maximize D because it has the advantage of being expressed in terms of the training data only. Hence, we can compute the optimal separating hyperplane, which is the classifier decision surface, from this data alone. Furthermore, it leads to a discriminate function (Equation E.31) that depends only on the dot product of the patterns, $\{\mathbf{x}_i \cdot \mathbf{x}_j | i, j = 1 \ldots n\}$, and allows generalization of the discriminate

function to the nonlinear case

We can solve the dual QP problem relatively easily using QP optimization techniques and numerical methods from a well studied class of convex quadratic programming problems [54]. The solution yields the optimum Lagrangian multipliers $\boldsymbol{\alpha}^0 = [\alpha_1^0, \alpha_2^0, \ldots, \alpha_n^0]$, one α for each training sample $\mathbf{x}_i$ The samples $\mathbf{x}_i$ for which $\alpha_i > 0$ are the support vectors. The support vectors lie on the supporting hyperplanes H_- and H_+ and are the sample points closest to the optimal hyperplane (the decision boundary). These are the critical training samples because they carry all the information required for classification; all other points can be removed or moved (as long as they don't fall between H_- and H_+) without effecting the optimal hyperplane [49].

The optimal separating hyperplane is found by computing $\mathbf{w}_0$ and b_0. From Equation E.26, we see that the former is calculated as

$$\mathbf{w}_0 = \sum_{i=1}^{n} \alpha_i^0 y_i \mathbf{x}_i \tag{E.29}$$

Then the latter is calculated as

$$b_0 = 1 - \mathbf{w}_0 \cdot \mathbf{x}_s \tag{E.30}$$

where $\mathbf{x}_s$ is a support vector that lies on H_+, which is defined as $\mathbf{w}_0 \cdot \mathbf{x}_i + b_0 = 1$ (see Equation E.19).

A simple discriminate function for linearly separable data can be written as

$$g(\mathbf{x}) = \text{sign}(\mathbf{w}_0 \cdot \mathbf{x} + b_0) = \text{sign}\left(\sum_{i=1}^{n} y_i \alpha_i^0 (\mathbf{x}_i \cdot \mathbf{x}) + b_0\right) \tag{E.31}$$

A more detailed derivation of this function is shown in subsequent sections.

E.2.2 SVMs for Linearly Inseparable Patterns: Soft Margin SVMs

For inseparable patterns, the SVM described in the last section will not work because a hyperplane cannot be constructed without classification errors, which the above SVM does not permit. What is needed is a method to maximize the margin of separation, ρ, while minimizing the error [54]. To do this we need to relax the constraints on Equation E.17 to allow some points to lie on the correct side of the optimal hyperplane, but between it and the supporting planes (H_- or H_+), while minimizing the number that fall on the incorrect side of the optimal hyperplane, as illustrated in Figure E.3. This is done by introducing slack variables $\xi_i \geq 0, i = 1, \ldots, n$ in the constraints of Equation E.17 [49] as follows:

$$\begin{aligned} \mathbf{w}_0 \cdot \mathbf{x}_i + b_0 &\geq 1 - \xi_i \quad \text{for } y_i = +1 \\ \mathbf{w}_0 \cdot \mathbf{x}_i + b_0 &\leq -1 + \xi_i \quad \text{for } y_i = -1 \end{aligned} \tag{E.32}$$

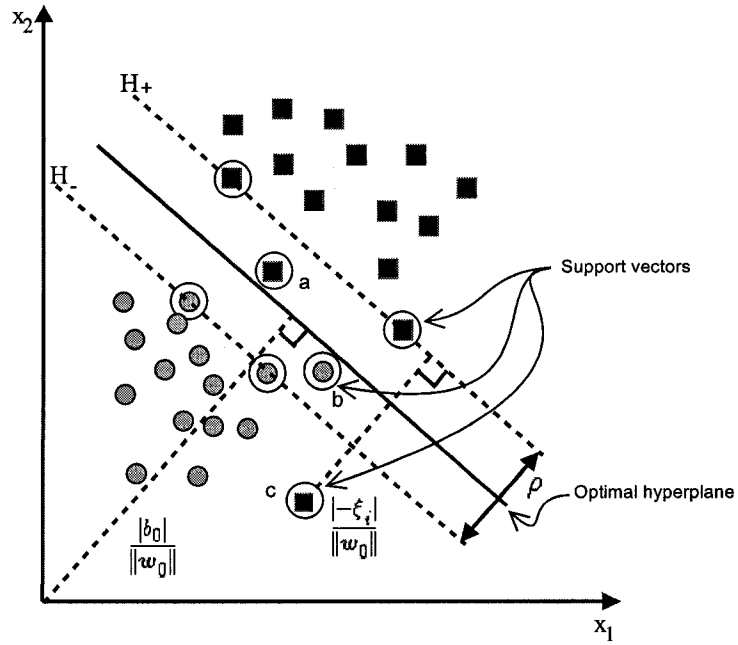


Figure E.3: An optimal separating hyperplane for two linearly inseparable, 2-dimensional, patterns, represented by circles and squares. The points circled are support vectors. Note that samples **a** and **b** are within the optimal hyperplane margin ρ, but on the correct side of the optimal plane, whereas **c** is an error.

Combining into canonical form we get

$$y_i(\mathbf{w}_0 \cdot \mathbf{x}_i + b_0) \geq 1 - \xi_i \tag{E.33}$$

which is the same as Equation E.18 except for the presence of the slack variable.

The presence of the slack variables, by necessity, effects the definition of support vectors; now the support vectors are all those that satisfy Equation E.33. When $0 \leq \xi_i \leq 1$, $\mathbf{x}_i$ lies on the correct side of the optimal hyperplane, but within ρ. When $\xi_i > 1$, $\mathbf{x}_i$ lies on the incorrect side of the optimal hyperplane, and so is in error.

To derive a SVM incorporating slack variables, we proceed as in the linear separable case detailed in Section E.2.1. First we formulate a new cost function which must be minimized with respect to $\mathbf{w}$.

$$\Phi(\mathbf{w}, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \tag{E.34}$$

Here $\sum_{i=1}^{n} \xi_i$ is the upper bound on the number of training errors. C, chosen a priori by the user, is called the regularization parameter. The higher C, the higher the cost associated with attaining training errors and the more complex the classifier.

Following Section E.2.1, we extend Equation E.23 to form the new primal QP

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1,\ldots,n \end{aligned} \tag{E.35}$$

The Lagrangian equivalent is

$$L(\mathbf{w},b,\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=0}^{n}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^{n}\gamma_i\xi_i \tag{E.36}$$

where γ_i are Lagrange multipliers introduced to enforce $\xi_i \geq 0$. Using methods similar to those used in the linearly separable case, the dual QP is (see [50, 59] for formal derivations)

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j \\ \text{subject to} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \; i = 1\ldots n \\ & C \geq 0 \end{aligned} \tag{E.37}$$

Note that the only difference between the inseparable QP and the separable QP is the upper bound, C, placed on the allowed values for the Lagrange multipliers α_i. As in the proceeding section, Equation E.37 is solved using convex quadratic programming problem solving techniques. The solution yields a vector of Lagrange multipliers $\boldsymbol{\alpha}^0 = [\alpha_1^0, \alpha_2^0, \ldots, \alpha_n^0]$. The optimal hyperplane is found by calculating $\mathbf{w}_0$ and b_0. As done in the separable case,

$$\mathbf{w}_0 = \sum_{i=1}^{n_s}\alpha_i^0 y_i \mathbf{x}_i \tag{E.38}$$

where n_s is the number of support vectors. To find b_0, Equation E.33 is rewritten as

$$\alpha_i^0(y_i(\mathbf{w}_0\cdot\mathbf{x}_i + b_0) - 1 + \xi_i) = 0 \tag{E.39}$$

and we solve for b_0. The only problem is we do not know ξ_i. However, if we chose a $(\mathbf{x}_i, y_i)$ such that $0 < \alpha_i^0 < C$ then $\xi_i = 0$. [49] suggests that it is better to calculate b_0 as the average of all b_i for all $(\mathbf{x}_i, y_i)$ when $0 < \alpha_i^0 < C$. The resultant discriminate function is the same as Equation E.31.

Table E.1: Common SVM kernels. In all cases, the kernel specific variables (σ, d, b, c) are chosen a priori by the user. For the neural network kernel, only certain values of b and c satisfy Mercer's theorem. Note, sometimes the RBF kernel is expressed using γ in place of $1/2\sigma^2$.

Name	Kernel $K(\mathbf{x}, \mathbf{x}_i)$
Linear	$\mathbf{x} \cdot \mathbf{x}_i$
Polynomial	$(\gamma \mathbf{x} \cdot \mathbf{x}_i + 1)^d$
Radial Basis Function (RBF)	$e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}}$
Neural Network	$\tanh(b(\mathbf{x} \cdot \mathbf{x}_i) - c)$

E.2.3 SVMs for Non-Linear, Inseparable Patterns: General SVMs

Using a methodology almost identical to that described in the previous sections, one can construct a SVM for inseparable, highly non-linear patterns. The method involves mapping the input vectors to a high dimensional feature space $\mathcal{F}$ (also called the dot-product space) via a nonlinear mapping

$$\Phi : \mathbb{R}^d \mapsto \mathcal{F} \tag{E.40}$$

where the patterns become linearly separable. The methods in the previous sections can then be applied, unchanged, in $\mathcal{F}$, i.e. the optimal separating hyperplane is constructed in $\mathcal{F}$. This is illustrated in Figure E.4.

Rewrite the optimization problem in its Lagrangian dual form as

$$D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{E.41}$$

and it can be solved as before, but finding Φ explicitly may be difficult. Additionally, it can be very expensive to compute it or the resulting dot-product in $\mathcal{F}$, which can have very high dimensions. However, for SVMs we can avoid both using the so called kernel trick [149]. Mercer's theorem [145] states that for certain mappings Φ and any two points $\mathbf{x}_i$ and $\mathbf{x}_j$ the inner product of the mapped points can be evaluated using a kernel function K without explicitly knowing the mapping [54]. Define K as

$$K(\mathbf{x}_i, \mathbf{x}_j) \triangleq \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{E.42}$$

then using Mercer's theorem it can be shown that a given K is suitable for a SVM (i.e. inner product in $\mathcal{F}$) but a suitable K or mapping Φ cannot be derived. Four common kernels that satisfy Mercer's theorem are summarized in Table E.1.

Using the notion of kernels Equation E.41 can be written as

$$D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{E.43}$$

This can be solved using the method of Lagrangian multipliers as described previously.

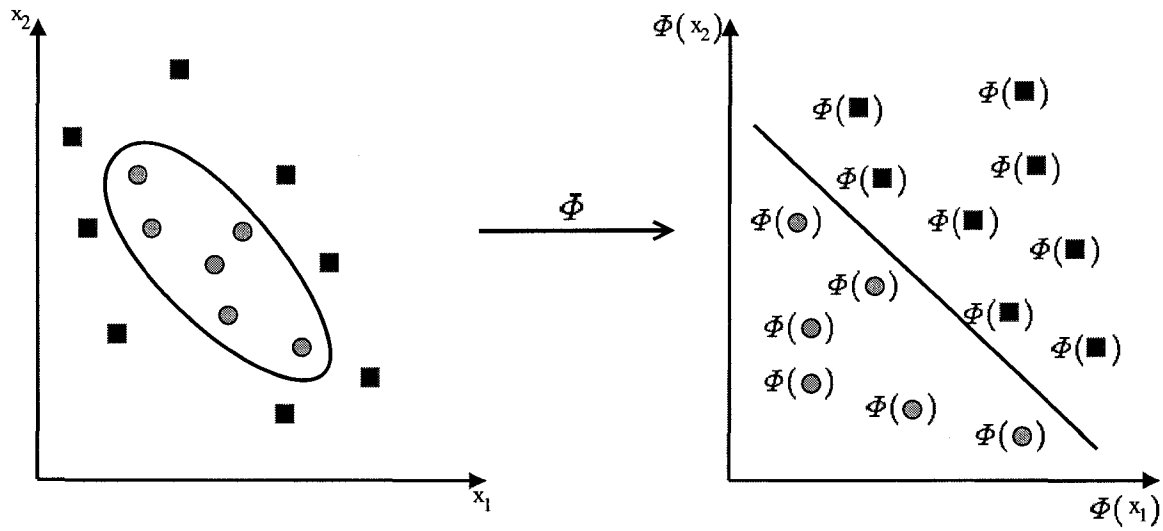


Figure E.4: Nonlinear map Φ from 2-dimensional input space to feature space. The nonlinear, non-separable patterns represented by circles and squares are transformed via Φ to linear space.

A discriminate function for classification implementing the optimal separating hyperplane can be written as

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} y_i \alpha_i^0 K(\mathbf{x}, \mathbf{x}_i) + b_0\right) \tag{E.44}$$

In essence a non-linear classifier is created from a linear classifier by replacing the dot-product with a kernel function. With the exception of this replacement, the SVM is the same as the linear SVM.

Appendix F

GA Concepts

F.1 The Basic GA

The basic operation of a GA is simple. The GA starts with an initial population of n randomly generated individuals, known as generation 0. The individuals, or chromosomes, represent a possible solution to the problem being solved and are specific to the problem domain. Each individual is evaluated using an evaluation mechanism designed for the problem. The results of evaluation determines the fitness of each individual. Following the principles of evolutionary theory, the fitness determines how well individuals "compete". In this sense, the greater the fitness of an individual the greater probability that genes of that individual will propagate to the next generation. In general, higher fit individuals stand a greater chance of mating with other individuals to form offspring that share characteristics of each parent. When two highly fit individuals mate, there is a good chance the resulting offspring will be better than the parents. As a consequence of this and the need to maintain a stable population size, some individuals will not mate, or mate less often. The result is that characteristics of less fit individuals will likely not make it to future generations. Generation i is created from generation $i-1$ individuals dominantly via mating. However, sometimes the genes of an individual will undergo a random mutation, making it slightly different from what normal mating would achieve. On average, with each subsequent generation the fitness of the chromosomes in the population improves. Eventually the child chromosomes are not significantly different from their parents, and the average fitness changes little. At this point the GA is said to have converged on a set of solutions. The standard genetic algorithm is shown in Algorithm 2.

Algorithm 2: Standard generational GA

```
Data: Evaluation data
Result: Final ℙ
1  begin
2      initialize ℙ_0
3      evaluate ℙ_0
4      i ← 1
5      while not halt condition do
6          select ℙ_i from ℙ_{i−1}
7          crossover ℙ_i
8          mutate ℙ_i
9          evaluate ℙ_i
10         i ← i + 1
11     end
12 end
```

F.2 Roulette Wheel Selection

A popular selection method that meets the desired selection properties of the best chromosomes mating more and the worst dying off is the roulette wheel selection. In this method, each individual in the current generation is assigned a slot on the wheel proportional to its fitness. The wheel is then spun once for each member of the population (n times), thus selecting the reproduction candidate based on the slot the wheel stops on. Copies of the n selected individuals enter a mating pool where subsequent genetic operators act on them.

The size (weight) of slot i for individual i is equal to the probability of selecting individual i:

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{F.1}$$

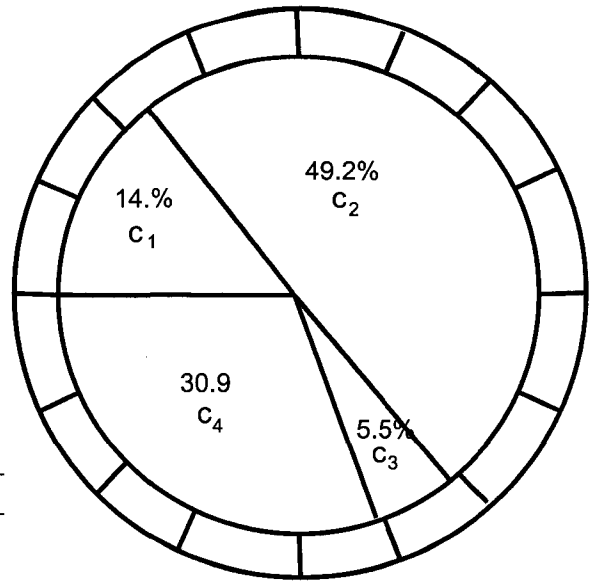
where f_i is the fitness of i and n is the population size. Thus with each wheel spin, i has the probability P_i of been selected and added to the mating pool. The expected number of copies of i in the mating pool is:

$$E_i = \frac{f_i}{\frac{\sum_{j=1}^{n} f_j}{n}} = n \times P_i \tag{F.2}$$

Figure F.1 shows an example of the roulette wheel for a trivial population of chromosomes.

ID	Chromosome	Fitness	$P_i \times 100\%$
c_1	01101	169	14.4
c_2	11000	576	49.2
c_3	01000	64	5.5
c_4	10011	361	30.9

(a)



(b)

Figure F.1: Illustration of the roulette wheel selection method [95, 98]. In this example, the chromosome encodes an integer x as binary and the fitness function if $f(x) = x^2$.

F.3 Single Point Crossover

A common technique for achieving crossover is the single-point crossover method. Two chromosomes, (c_i, c_j), are randomly chosen for mating. A position k is chosen randomly and uniformly from the range $[1, l_c - 1]$, where l_c is the length of the chromosome. Two offspring, (c'_i, c'_j), are created by swapping the tail portions of each. The tail portion is defined as all genes in the range $[k + 1, l_c]$. This is illustrated in Figure F.2.

F.4 Mutation

Mutation is a random change in one or more of the bits in a chromosome. For bit-string chromosomes, if mutation occurs, its value is flipped. Figure F.3 illustrates this.

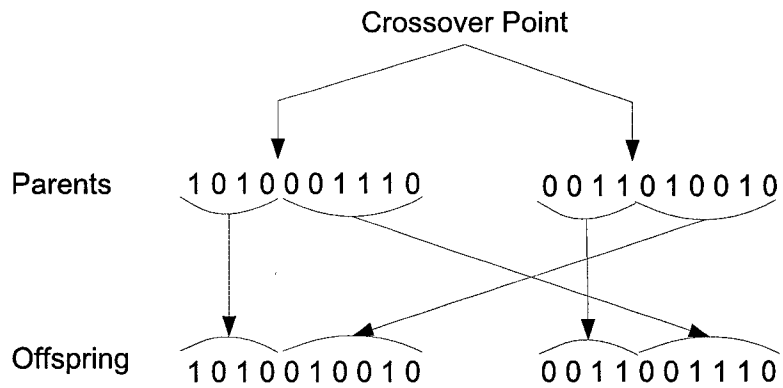


Figure F.2: Illustration of single point crossover in a GA. $l_c = 10$, $k = 4$, and the tail is bits 5 to 10 (modified from [101]).

$$\begin{array}{ccc} 01\mathbf{0}00 & \Longrightarrow & 01\mathbf{1}00 \\ f = 64 & & f = 144 \end{array}$$

Figure F.3: Illustration of GA chromosome mutation (continued from Figure F.1). In this example allele 2 (counting from 0, right to left) of the chromosome c_3 was mutated increasing of fitness from 64 to 144. This would increase the chance of c_3 being selected for mating.