SENSOR FUSION MODELS TO INTEGRATE ELECTRONIC NOSE AND SURFACE ACOUSTIC WAVE SENSOR FOR APPLE QUALITY EVALUATION

A Thesis in
Agricultural and Biological Engineering
by
Changying Li

© 2006 Changying Li

Submitted in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

December 2006
The thesis of Changying Li was reviewed and approved* by the following:

Paul H. Heinemann  
Professor of Agricultural and Biological Engineering  
Thesis Advisor  
Chair of Committee

Virendra M. Puri  
Professor of Agricultural and Biological Engineering

Paul N. Walker  
Professor of Agricultural and Biological Engineering

Devin G. Peterson  
Assistant Professor of Food Science

Roy E. Young  
Professor of Agricultural and Biological Engineering  
Head of the Department of Agricultural and Biological Engineering

*Signatures are on file in the Graduate School
ABSTRACT

Because of the importance of fresh produce in the U.S. to both the economy and people’s health, it is essential to maintain the quality of this valuable resource. In this project, sensor fusion technology was applied to two artificial noses: the Cyranose 320 electronic nose (Enose) and a surface acoustic wave sensor (zNose™), in order to develop a system for non-destructive, rapid detection the safety and quality of fresh produce.

Dominant volatile compounds associated with healthy apples and physically damaged apples were identified by gas chromatography and mass spectrometry (GC-MS). The results proved that the volatile compounds from healthy apples and damaged apples are different both qualitatively and quantitatively.

The Enose and zNose™ were firstly independently studied. Different statistical models, such as PLS-DA and PLS-CVA, were developed and performed on the data on individual days and one single month. It was found that statistical models were effective for separating healthy from damaged apples when individual days or single month data were analyzed. When data from different months were combined, statistical models could not give desirable results due to the non-linearity of this problem. In order to improve the system classification performance, artificial neural networks (ANN) were used to develop classification models. Three ANN models (back-propagation, probabilistic, and learning vector quantification networks) were developed and tested on data sets collected in three different months. Results showed that all three ANN models achieved better classification performance than statistical models when data from
different months were pooled together for both the Enose and zNose™ data. Among these three ANN models, the PNN was superior, considering the classification quality (85% and 77% classification accuracy for the Enose and zNose™ respectively) and efficiency (training was faster than BP and LVQ).

Another focus of this research was to reduce data dimensionality of the Enose and zNose™. Various methods were investigated towards this end. Although methods such as the PCA loadings method, F-value selection and sequential forward/backward search reduced data dimensionality to various degrees, evolutionary algorithms were proven to be a more powerful and robust search approach. Evolutionary algorithms reduced data dimensionality 75% and 50% for the Enose and zNose™ respectively, and the classification error rate for the two instruments was reduced by 10% for the Enose and 20% for the zNose™.

Multisensor data fusion models both at the feature and decision levels were developed to combine the Enose and zNose™ data to improve classification performance. At the feature level, ANN-based fusion models (dynamic selective fusion) reduced the classification error rate to 1.8% on average in 30 independent runs, and at the same time only about 50% of the sensors from the Enose and zNose™ were used for input. At the decision level fusion, a Bayesian network was developed to combine classification decisions made by the Enose and zNose™ classifiers independently. It was found that using soft evidence produced by the BP classifier either with or without prior performance knowledge gave the best improvement of classification performance.

Finally, trained models were tested on new data sets which were collected by measuring the presence of one bad apple placed amongst three good apples in a 4 L
concentration chamber. Sensor fusion models could achieve 81% and 82% classification accuracy at the feature level and decision level; when selected sensors were updated, the classification accuracy of sensor fusion models were improved to 97% at the feature level and 91% at the decision level.

This study introduced three artificial intelligent technologies into food quality and safety evaluation: artificial neural networks, evolutionary algorithms, and multisensor data fusion, utilizing information from two advanced volatile detection instruments. Sophisticated algorithms improved the performance of two artificial noses and showed promise of eventually achieving non-destructive detection of physically damaged and fungi-diseased apples.

**Key words:** electronic nose, surface acoustic wave sensor, zNose™, artificial nose, gas sensors, olfaction, odor, artificial neural networks, evolutionary algorithms, multisensor data fusion, genetic algorithms, covariance matrix adaptation evolutionary algorithms, differential evolution, probabilistic neural networks, back-propagation neural networks, learning vector quantification networks, principal component analysis, partial least square, discriminant analysis
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>xiii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xvi</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>xviii</td>
</tr>
<tr>
<td>Chapter 1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Significance of this study</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Objectives of this research</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Overview of methodology</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Structure of this thesis</td>
<td>6</td>
</tr>
<tr>
<td>Chapter 2. Background</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Apple production and postharvest issues</td>
<td>8</td>
</tr>
<tr>
<td>2.1.1 Losses of fruit production</td>
<td>10</td>
</tr>
<tr>
<td>2.1.2 Apple postharvest disorders</td>
<td>11</td>
</tr>
<tr>
<td>2.1.3 Fresh produce safety issues</td>
<td>13</td>
</tr>
<tr>
<td>2.2 Apple volatile compounds</td>
<td>15</td>
</tr>
<tr>
<td>2.2.1 Biogenesis of apple aroma</td>
<td>15</td>
</tr>
<tr>
<td>2.2.2 Apple volatile compounds</td>
<td>16</td>
</tr>
<tr>
<td>2.2.3 Diseased apple volatile profiles</td>
<td>20</td>
</tr>
<tr>
<td>2.3 Headspace analysis</td>
<td>20</td>
</tr>
<tr>
<td>2.3.1 SPME: solid phase micro extraction</td>
<td>23</td>
</tr>
<tr>
<td>2.3.2 GC-MS</td>
<td>25</td>
</tr>
<tr>
<td>2.4 Biological olfaction</td>
<td>28</td>
</tr>
<tr>
<td>2.4.1 Physiology of biological olfaction</td>
<td>29</td>
</tr>
<tr>
<td>2.4.2 Molecular understanding of olfaction</td>
<td>31</td>
</tr>
<tr>
<td>2.4.3 Computational olfaction</td>
<td>33</td>
</tr>
<tr>
<td>2.5 Machine olfaction</td>
<td>34</td>
</tr>
<tr>
<td>2.5.1 Electronic nose</td>
<td>34</td>
</tr>
</tbody>
</table>
2.5.2 Gas sensors ................................................................. 38
- Metal oxide semiconductor sensors ............................. 38
- Conducting polymer gas sensors ............................. 40
- Surface acoustic wave (SAW) sensors ...................... 41
2.5.3 Optical gas sensors and others ............................... 43
2.6 Pattern recognition algorithms ........................................ 44
2.6.1 Overview of pattern recognition .......................... 45
2.6.2 Artificial neural networks ....................................... 47
- BP network and gradient decent learning algorithm .......... 49
- PNN and radial basis networks .................................. 52
- Learning vector quantification and competitive learning ................................................................. 53
2.6.3 Statistical methods ................................................... 55
- Principal component analysis .................................. 55
- Bayesian discriminant analysis ................................. 56
2.7 Evolution algorithms and optimization ................................. 57
2.7.1 Optimization ............................................................ 57
2.7.2 Evolutionary algorithms .......................................... 60
- Genetic algorithms ...................................................... 61
- Evolutionary strategy (ES) ........................................... 64
2.7.3 Evolutionary multiobjective optimization (EMO) ........... 68
2.8 Multisensor data fusion ................................................... 70
2.8.1 Introduction ............................................................. 70
2.8.2 Architectures of data fusion ..................................... 74
2.8.3 Mathematical techniques in multisensor data fusion ....... 77
2.8.4 Applications ............................................................. 78

Chapter 3. Preliminary experiments ........................................... 82
3.1 The Enose sampling parameters determination .................. 82
3.1.1 Sampling method set up ............................................ 82
Chapter 4. Detection of apple defects using an electronic nose and zNose™

Abstract ...................................................................................................... 92
4.1 Introduction .......................................................................................... 93
4.2 Materials and Methods ......................................................................... 96
  4.2.1 Samples ............................................................................... 96
  4.2.2 Enose data acquisition .......................................................... 97
  4.2.3 zNose™ data acquisition ................................................... 99
  4.2.4 Data analysis ....................................................................... 100
  4.2.5 Gas chromatography-mass spectrometry ............................ 101
4.3 Results and Discussions........................................................................ 102
  4.3.1 Volatile compounds from GC-MS analysis................................. 102
  4.3.2 Enose sensor reduction ....................................................... 107
  4.3.3 Discrimination of volatiles by the Enose and zNose™ ...... 110
    4.3.3.1 Effect of different physical damage on classification ......................... 110
    4.3.3.2 Time effect ...................................................................... 112
    4.3.3.3 Orientation effect ................................................... 115
    4.3.3.4 Classification between undamaged and damaged apples ............... 116
  4.4 Conclusions .......................................................................................... 117

Chapter 5. ANN-integrated electronic nose and zNose™ system
for apple quality evaluation ...................................................................... 118
Abstract ...................................................................................................... 118
5.1 Introduction .......................................................................................... 119
5.2 Materials and Methods ......................................................................... 122
5.2.1 Sample preparation and data sets summary ........................................ 123
5.2.2 Enose experiment ............................................................................. 124
5.2.3 zNose™ experiment ........................................................................ 126
5.2.4 Feature extraction and PCA ............................................................. 127
5.2.5 ANN models (BP, PNN and LVQ) .................................................... 128

5.3 Results and Discussions ........................................................................ 129
5.3.1 The Enose and zNose™ sampling ................................................... 129
5.3.2 PCA clustering analysis .................................................................... 130
5.3.3 ANN model development and performance evaluation .................. 132
  5.3.3.1 BP network over-fitting prevention ........................................... 132
  5.3.3.2 Neural network architecture .................................................. 133
  5.3.3.3 Three ANN models classification performance comparison ........ 137
  5.3.3.4 Sensitivity and specificity ..................................................... 140
  5.3.3.5 Generalization ........................................................................ 142
  5.3.3.6 Incremental learning ............................................................ 145

5.4 Conclusions ............................................................................................ 146

Chapter 6. Genetic algorithms (GAs) and evolutionary strategy to optimize electronic nose sensor selection .......................................................... 149
Abstract .................................................................................................... 149
6.1 Introduction ............................................................................................ 150
6.2 Methodology .......................................................................................... 152
  6.2.1 Sampling process and data collection ............................................ 152
  6.2.2 Principal component analysis ...................................................... 153
  6.2.3 PNN classification model ............................................................. 154
  6.2.4 Sequential forward/backward search ......................................... 155
  6.2.5 Statistics method: F-value selection ............................................ 156
  6.2.6 Genetic algorithms (GAs) ............................................................. 157
  6.2.7 CMA evolutionary strategy .......................................................... 159
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Flowchart of methodology</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Formation of volatile aroma in fruits and vegetables</td>
<td>16</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Principle of dynamic headspace sampling: IG=inert gas; CG=carrier gas; SV=sample vessel; TR=trap; D=detector</td>
<td>22</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Static headspace analysis: CG=carrier gas; SV=sample vial; TH=thermostat; COL=column; D=detector.</td>
<td>23</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>SPME schematic diagram</td>
<td>24</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>A schematic diagram of gas chromatography and mass spectrometry (GC-MS)</td>
<td>26</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>Illustration of (a) chromatogram generated by a GC; (b) mass spectrum generated by an MS</td>
<td>28</td>
</tr>
<tr>
<td>Figure 2.7</td>
<td>Anatomy of human olfactory system</td>
<td>30</td>
</tr>
<tr>
<td>Figure 2.8</td>
<td>Olfaction decoding model</td>
<td>32</td>
</tr>
<tr>
<td>Figure 2.9</td>
<td>A generic electronic sensor architecture</td>
<td>37</td>
</tr>
<tr>
<td>Figure 2.10</td>
<td>Diagram of metal oxide semiconductor gas sensor</td>
<td>39</td>
</tr>
<tr>
<td>Figure 2.11</td>
<td>Working principle of conducting organic polymer gas sensor</td>
<td>41</td>
</tr>
<tr>
<td>Figure 2.12</td>
<td>(a), Cross-sectional view of Raleigh wave operation (b) Surface acoustic wave sensor schematic diagram</td>
<td>42</td>
</tr>
<tr>
<td>Figure 2.13</td>
<td>Optical gas sensor diagram</td>
<td>44</td>
</tr>
<tr>
<td>Figure 2.14</td>
<td>A typical pattern recognition system</td>
<td>46</td>
</tr>
<tr>
<td>Figure 2.15</td>
<td>Schematic diagram of a simple neuron model</td>
<td>48</td>
</tr>
<tr>
<td>Figure 2.16</td>
<td>A three layer feed forward error back-propagation model</td>
<td>50</td>
</tr>
<tr>
<td>Figure 2.17</td>
<td>Radial basis network model (a) and radial basis function (b)</td>
<td>52</td>
</tr>
<tr>
<td>Figure 2.18</td>
<td>Competitive network model</td>
<td>53</td>
</tr>
<tr>
<td>Figure 2.19</td>
<td>Global optimization approaches</td>
<td>58</td>
</tr>
<tr>
<td>Figure 2.20</td>
<td>Multimodal (a) and discontinuous (b) optimization problems</td>
<td>59</td>
</tr>
<tr>
<td>Figure 2.21</td>
<td>Comparison of different optimization schemes for different problems types</td>
<td>60</td>
</tr>
<tr>
<td>Figure 2.22</td>
<td>Four typical mutation operators</td>
<td>66</td>
</tr>
<tr>
<td>Figure 2.23</td>
<td>Evolution strategies (ES) principle flow chart</td>
<td>67</td>
</tr>
<tr>
<td>Figure 2.24</td>
<td>Pareto front for a two-objective function optimization problem</td>
<td>69</td>
</tr>
<tr>
<td>Figure 2.25</td>
<td>Rattlesnake receives visual and infrared information from environment</td>
<td>71</td>
</tr>
<tr>
<td>Figure 2.26</td>
<td>Data fusion process model</td>
<td>75</td>
</tr>
<tr>
<td>Figure 2.27</td>
<td>High level data fusion</td>
<td>75</td>
</tr>
<tr>
<td>Figure 2.28</td>
<td>Intermediate level fusion</td>
<td>76</td>
</tr>
<tr>
<td>Figure 2.29</td>
<td>Low level fusion</td>
<td>76</td>
</tr>
<tr>
<td>Figure 2.30</td>
<td>Mathematical algorithms for data fusion</td>
<td>77</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Cyranose 320 Enose (Smith Detection, Herts, UK) and zNose™ (Electronic Sensor Technology, Newbury Park, CA)</td>
<td>83</td>
</tr>
</tbody>
</table>
Figure 3.2  Enose 10 s (a) and 30 s (b) sample draw time response comparison
Figure 3.3  Enose sensor responses with four different sample concentration times
Figure 3.4  Enose sensor responses to empty container, water vapor and ambient air
Figure 3.5  The Enose responses to red Delicious apples with four different purge gases
Figure 3.6  zNose™ calibration using dynamic headspace (a); five replicate analyses performed on C6-C14 n-alkane vapors (b)
Figure 3.7  zNose™ responses to apple volatile compounds and room air
Figure 4.1  An experimental schematic for Enose data collection
Figure 4.2  The zNose™ 7100 schematic diagram
Figure 4.3  Number of compounds detected from undamaged (healthy) apples and damaged apples
Figure 4.4  Volatile compounds quantity comparison between undamaged (healthy) apples and damaged apples
Figure 4.5  GC-MS chromatograms of headspace volatiles from undamaged (healthy) and damaged Red Delicious apples 9 days after an artificially induced cut
Figure 4.6  Variance contribution from the first 10 PCs
Figure 4.7  Enose PCA plot 14 days after the cut
Figure 4.8  Enose PCA plot for undamaged (a) and damaged (b) apples on different days
Figure 5.1  Schematic of classification system based on ANN models
Figure 5.2  Three stages of the Enose sensor response: (A) baseline purge, (B) sample exposure, and (C) sensor refresh
Figure 5.3  Enose "smellprints"
Figure 5.4  zNose™ "smellprints"
Figure 5.5  Enose PCA plot for September data
Figure 5.6  zNose™ PCA plot for all three data sets
Figure 5.7  BP network early stopping training
Figure 5.8  (A) Log-sigmoid (logsig), (B) tan-sigmoid (tansig) and (C) radial basis transfer functions
Figure 5.9  BP network architecture
Figure 5.10  PNN network architecture
Figure 5.11  LVQ network architecture
Figure 5.12  Network classification illustration
Figure 5.13  BP, PNN an LVQ model classification results for Enose data
Figure 5.14  BP, PNN an LVQ model classification results for zNose™ data
Figure 6.1  PNN network architecture (IW is input weight; || dist || is a vector whose elements indicate how close the input is to the vectors of the training set)
Figure 6.2  CMAES algorithm flow chart
Figure 6.3  3D PCA plot comparison between individual-day data (a) and pooled data (b)
Figure 6.4  The SFS (above) and SBS (below) search process for the Enose sensor selection
Figure 6.5  F-values for 32 sensors
Figure 6.6  Binary coded GA (a) and integer coded GA (b) optimization convergence plots
Figure 6.7  Evolution history comparison between binary coded and real number coded
Figure 6.8  Evolution history of different population sizes
Figure 6.9  Sensor selection distribution for 30 random number seeds runs
Figure 6.10  Relationship between number of sensors and the classification error rate
Figure 7.1  The zNose™ spectrum for healthy apples and damaged apples
Figure 7.2  The zNose™ spectrum with 16-window discretization scheme
Figure 7.3  PNN network architecture
Figure 7.4  Flowchart of GA application
Figure 7.5  CMAES algorithm flow chart
Figure 7.6  Schematic diagram of Differential Evolution (DE)
Figure 7.7  GA evolutionary history
Figure 7.8  CMAES evolutionary history
Figure 7.9  DE evolutionary history
Figure 7.10  GA parametric study
Figure 7.11  CMAES parametric study
Figure 7.12  DE parametric study
Figure 7.13  zNose™ wavelength selection illustration for 16, 32 and 64 windows schemes (Note: the solid blue line represents selected wavelengths, the dashed red line shows removed wavelengths)
Figure 7.14  Frequency of selection of wavelengths in 64-window scheme
Figure 8.1  Enose and zNose™ "smellprints"
Figure 8.2  Feature level (top) and decision level (bottom) fusion schemes
Figure 8.3  PNN network architecture (IW is input weights; || dist || is a vector whose elements indicate how close the input is to the vectors of the training set; LW is layer weights)
Figure 8.4  BP network architecture (IW is input weights; LW is layer weights; b is bias)
Figure 8.5  CMAES algorithm flow chart
Figure 8.6  Bayesian fusion flowchart
Figure 8.7  Bayesian Network structure
Figure 8.8  Three ANN fusion schemes: (a) non-selective fusion; (b) static selective fusion; (c) selective fusion.
Figure 8.9  A typical 48-variable search history by the CMAES
Figure 8.10  Selected sensors from the zNose™ (top) and Enose (bottom)
LIST OF TABLES

Table 2.1 Estimated loses of some fresh fruits ................................. 10
Table 2.2 Apple external and internal physiological injuries .......... 11
Table 2.3 Important harvest diseases in apples ......................... 12
Table 2.4 Volatile compounds produced by apples ...................... 17
Table 2.5 Correlation between volatile levels and indices of maturity and ripeness in McIntosh Apples ................ 19
Table 2.6 Different SPME fibers and their recommended application fields 25
Table 2.7 Comparative properties of “cold” and “hot” sensors ........ 35
Table 2.8 Comparison of natural and GA terminology .................. 63
Table 2.9 Representative EMO algorithms ................................. 70
Table 3.1 The Enose sampling method set up .............................. 85
Table 3.2 Temperature settings for DB-5 column in the zNose™ ........ 89
Table 3.3 Sampling time set up for each process ....................... 89
Table 4.1 Cyranose 320 instrumental parameters for apple volatile measurement ............................................. 98
Table 4.2 zNose™ operational parameters .................................. 100
Table 4.3 Volatile compounds in the headspace of damaged red Delicious apples ....................................................... 106
Table 4.4 Volatile compounds in the headspace of undamaged Red Delicious apples ....................................................... 107
Table 4.5 Principal component loadings and sensor selection at day 5 after the cut .................................................. 109
Table 4.6 Sensor selection for different datasets ......................... 109
Table 4.7 Enose performance for the two different sensor selection schemes 110
Table 4.8 Classification accuracy (% correct) comparison for the Enose and zNose™ 14 days after deterioration 111
Table 4.9 The Enose and zNose™ correct classification to different orientation effect ............................................. 115
Table 4.10 Comparison of different data compression and classification models (% correct classification) .................. 116
Table 5.1 Sampling protocol for the Enose and zNose™ ................. 124
Table 5.2 Composition of testing and training sets ....................... 124
Table 5.3 BP network hidden neurons testing (Enose) .................. 135
Table 5.4 BP network hidden neurons testing (zNose™) ................ 136
Table 5.5 Classification confusion Table with BP, (PNN) and [LVQ] for Enose data ....................................................... 139
Table 5.6 Classification confusion table with BP, (PNN) and [LVQ] for zNose™ data ....................................................... 140
Table 5.7 Sensitivity and specificity for BP, PNN and LVQ networks with Enose data ....................................................... 142
Table 5.8 Sensitivity and specificity for BP, PNN and LVQ networks with zNose™ data ....................................................... 142
Table 5.9 Results of the generalization of BP, [PNN] and (LVQ)
networks in damaged apple classification, in terms of patterns correctly classified/number of patterns (Enose)
ACKNOWLEDGEMENTS

Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.

Sir Winston Churchill (1874 - 1965)

I am deeply grateful to my advisor, Dr. Paul Heinemann, for his steadfast support, constant encouragement, and invaluable guidance throughout my Ph.D. study. I have both benefited from his keen insights and enjoyed the freedom of thinking during my research. I sincerely thank him for his precious help and advice on my professional development and career choices.

I gratefully acknowledge professors serving in my committee: Dr. Virendra Puri, Dr. Paul Walker, and Dr. Devin Peterson for their time, suggestions, and support. I thank them for carefully reviewing my whole dissertation and giving important comments. Special thanks go to Dr. Peterson for kindly making his equipment available for use in this research. I also want to express my appreciation to Dr. Joseph Irudayaraj, who had served in my committee in the first two years, for his insights and careful revision of my manuscript. I thank Dr. David Hall, Dr. Rick Sherry, and Dr. Patrick Reed for their kind help on my research and providing some source code to me. Special appreciation must be extended to ABE Department Head Dr. Roy Young not only for providing financial support for me, which makes this work possible, but for his valuable encouragement, and spending time reading this thesis. For those faculty and staff members at ABE
Department, who gave me important guidance, help and encouragement during my study, Dr. Dennis Buffington, Dr. Manbeck, Nyman Wanda, Brenda, Bill, to name a few, I am truly in their debt.

Special thanks are due to my friend Yong Tang for introducing genetic algorithms and providing useful resources to me, and Huidong Gao for his stimulating discussions and reviewing part of my thesis. I feel privileged to have known so many wonderful friends during my Ph.D. study not only within the ABE Department, but in the Chinese Friendship Association, Student Activity Committee of AOC. Their gentle help and friendship makes this journey unforgettable. Deep appreciation also goes to my soul mates and constant intellectual companions: Yujian Xie, Feng Ding, and Yancheng Yang for giving me motivation, help and support.

I am greatly indebted to my dear parents for their consistent love, education, and understanding throughout my life, without which I could not even have had the chance to start this journey. I am obliged to my grand-grandma whom I had lived with for 13 years, and I feel guilty not being able to see her since I started this work. I also thank my younger sister for always being in my corner. This work is dedicated to them with all my love.
CHAPTER 1

INTRODUCTION

A tower of nine stories begins with a heap of earth; the journey of a thousand miles starts from where one stands.

_Lao-Tzu (604-531 BC)_

1.1 Significance of this study

Fruits and vegetables provide important nutrients and trace elements in the human diet such as vitamin C, vitamin A, vitamin B₆, flavonoids, magnesium, iron, zinc, and calcium (Knee, 2002). Clinical studies suggested that due to their antioxidant activity, frequent consumption of fruits and vegetables can reduce the risk of certain types of cancer, heart disease, stroke and other chronic diseases (Prior and Cao, 2000). Owing to their importance to human life, fruits and vegetables production is one of the most important parts in agricultural industry and their annual sales exceed 20 billion dollars in 2004 (USDA-NASS, 2006). However, fresh produce are also highly perishable commodities and it is estimated that more than 40% strawberries and 8% apples were wasted due to physical damages, spoilage, and fungi-induced diseases before they reach the market (Salunkhe and Desai, 1984). Furthermore, food safety has become the most important concern in fresh produce production due to several multi-state food-borne disease outbreaks and the impending threat of bio-terrorism (Centers for Disease Control
and Prevention, 1998). Therefore, early detection of spoiled and diseased fruits is paramount important not only to reduce economic losses but save human lives as well.

Although many attempts have been made to develop non-destructive methods for fruit quality and safety detection using X-ray (Yantarasri et al., 1998), chlorophyll fluorescence (Mir et al., 1998), nuclear magnetic resonance (NMR) (Clark et al., 1998), near infrared spectroscopy (NIRS) (Lu, 2004), and machine vision (Paulus et al., 1998), most of them are cumbersome, expensive, or incompetent when defects or pathogens are invisible or occluded by other healthy fruits.

Volatile compounds emitted by fresh produce provide important information on their quality and ripeness because the presence of diseases or development of maturity induces a compositional changes in volatiles (Simon et al., 1996). Likewise, pathogenic volatile compounds emitted from the surface of fresh produce can also be utilized for pathogen detection. The artificial nose, which mimics the human olfactory system to “smell” the fruits and makes decisions on fruit quality and safety based on volatile information, has shown promise for non-destructive measurement of fresh produce quality and safety in the past decade (Brezmes et al., 2001; Llobet et al., 1999; Oshita et al., 2000). In this research, two commercial artificial noses, the Cyranose 320 electronic nose and zNose™, were integrated by sensor fusion models to detect spoiled and fungi-diseased red ‘Delicious’ apples. Pattern recognition algorithms for these two artificial noses were developed using artificial neural networks; the high dimensionality problem of the electronic nose was addressed by using evolutionary algorithms, which are a robust search and optimization approach.
1.2 Objectives of this research

The following five objectives were pursued in this project:

1) Use gas chromatogram and mass spectrometry to identify what volatile compounds are present in the undamaged and damaged apple headspace gas, and use statistical tests to determine whether the Enose and zNose™'s responses to two classes of apples are statistically different.

2) Design pattern classification models that can effectively classify two classes of apples (undamaged and damaged) using the Enose and zNose™ multivariate data.

3) Develop feature selection algorithms using statistical and heuristic search methods to reduce data dimensionality of the Enose and zNose™.

4) Develop multisensor data fusion models to fuse the Enose and zNose™ data for better classification.

5) Validate trained multisensor data fusion models by using unseen data sets, and test these to determine if one damaged apple can be identified amongst a group of undamaged apples.

1.3 Overview of methodology

There are seven phases in this research, as illustrated in Figure 1.1.

1) In phase 1, sampling methods and a concentration chamber were designed. The Enose and zNose™ operating parameters were determined.

2) In the second phase, preliminary experiments were conducted to determine sample concentration time, purge gas, and data preprocessing techniques.
3) In the third phase, dominant volatile compounds emitted by healthy and damaged apples were identified by gas chromatography-mass spectrometry (GC-MS). Key volatiles that differentiate healthy and damaged apples were also identified in relevant literature.

4) Calibration models were designed in the fourth phase. Statistical models including partial least square discriminant analysis (PLS-DA) and canonical variance analysis (CVA) were developed and compared with artificial neural networks models such as backpropogation (BP), probabilistic neural network (PNN) and learning vector quantification (LVQ) networks.

5) In phase 5, different approaches were applied to select the most relevant features from the Enose and zNose™ data in order to reduce data dimensionality and improve classification results.

6) In phase 6, multi-sensor data fusion models were developed at feature level and decision level using artificial neural networks (PNN) and Bayesian network, respectively.

7) In the final phase, developed calibration models were validated using new data sets which were tested on containers holding one damaged apple amongst four total apples.
Figure 1.1. Flowchart of methodology
1.4 Structure of this thesis

In Chapter 2, a general knowledge and research background is presented in the following areas: food quality and safety issues, machine and biological olfaction, pattern classification, apple volatiles, volatile organic compounds measurement, and sensor fusion.

In Chapter 3, preliminary experiments were conducted to determine the two instruments’ operating parameters, sample handling techniques, and data preprocessing approaches.

In Chapter 4, the Enose and zNose™ were individually studied to classify healthy apples and damaged apples using statistical methods. The GC-MS experiments were conducted to determine differences in volatile compounds between healthy apples and damaged apples.

In Chapter 5, three artificial neural networks models: BP, PNN and LVQ were developed and their various performance indices were compared.

In Chapter 6, genetic algorithms (GA) and covariance matrix adaptation evolutionary algorithms (CMAES) were studied to select useful sensors from the Enose. Other methods such as SFS/SBS, and F-value selection were also compared.

In Chapter 7, three evolutionary algorithms (GA, CMAES and differential evolution) were studied to select relevant wavelength windows in zNose™ spectral data.

In Chapter 8, multi-sensor data fusion models were developed at the feature level and decision level using artificial neural networks and Bayesian network.
In Chapter 9, individual sensor’s models and sensor fusion models were validated by testing on new samples, in which one bad apple was placed among three healthy apples in a 4 L glass jar.

In Chapter 10, general conclusions are made and future research directions are given.

Chapter 4—Chapter 8 were five manuscripts written individually. They are stand-alone, aiming to solve different problems; they are also integral parts of this whole dissertation by answering five questions raised in this chapter. Hence, these five chapters remain in manuscript format.
CHAPTER 2

BACKGROUND

*If I have seen further it is by standing on the shoulders of giants.*

*Isaac Newton (1642-1727)*

In this section, theoretical background related to this research is presented on the following topics: food safety and quality issues, apple volatiles review, volatile organic compounds (VOC) measurement, biological olfaction system, machine olfaction system, pattern classification, feature selection, and multisensor data fusion.

Since literature reviews were conducted in chapters 4-8, which are in independent manuscript format, in order to avoid redundancy, only content that was not present in these five chapters is presented.

2.1. Apple production and postharvest issues

*There is no fruit in temperate climates so universally esteemed, and so extensively cultivated, nor is there any which is so closely identified with the social habits of the human species as the apple.*

- Dr. Robert Hogg, 1851

Apple (*Malus domestica*, Borkh) is one of the most often consumed fruits and can grow in temperate climatic zones in all continents except Antarctic. China became the largest apple producer by the beginning of the 21st century, followed by the United States,
Poland, Turkey, Italy, France, Germany, Argentina and Japan (USDA-FAS, 2005). As the second largest apple production country in the world, the United States produced nearly 5 million tons and generated 1.7 billion dollars revenue in 2004 (USDA-NASS, 2006). Washington State is the leading apple-producing state with 58% total production in the U.S., followed by New York (12%), Michigan (7%), California (4%), and Pennsylvania (4%) (USDA-NASS, 2006).

Sixty percent of U.S. apples were consumed as fresh fruit, and 39% were processed into apple products including apple juice, cider, dried and canned products. In 2002, the average U.S. consumer consumed an estimated 42 pounds of fresh apples and processed apple products (University of Illinois Extension, 2005).

The top five most popular apple cultivars in the U.S. are: ‘Delicious’, ‘Golden Delicious’, ‘Gala’, ‘Fuji’ and ‘Granny Smith’. Delicious apple is the most popular cultivar in the U.S. due to its distinct color and appearance. It is also known as ‘Hawkeye’, ‘Red Delicious’, and ‘Stark Delicious’ (Ferre and Warrington, 2003). Delicious apples can usually be stored for 3-4 months in ambient room air, and as long as 6-11 months in controlled atmosphere (CA) storage with 0.7 to 2.5% O₂, 0 to 4.5% CO₂, and at -0.5 to 1.1°C. They are not chilling-sensitive or sensitive to low O₂. But they are prone to scald and heat injury, particularly if they are picked too early. Delicious is also susceptible to watercore and moldy core, and can become mealy (soft, dry, and friable) at late harvest. Although they can get bitter pit, it is preventable by proper fertilization practices and crop-load management. They are susceptible to apple scab and mullein bug, but resistant to powdery mildew and highly resistant to fire blight (Ferre and Warrington, 2003).
2.1.1 Losses of fruit production

It is estimated that roughly 30-40% of crops in developing countries never get to the consumer due to food spoilage and waste during the marketing process (Salunkhe and Desai, 1984). This is especially true for fresh produce production because they are highly perishable. As seen in Table 2.1, 41.2% of strawberries were lost before they got to the consumer, and 8.2% of apples were wasted before reaching the market (Salunkhe and Desai, 1984). It is not surprising that some researchers even claimed the reduction of postharvest food losses the “hidden harvest” (Spurgeon, 1976). Minimizing or eliminating the food losses can not only increase world food supply and reduce hunger, but save energy used to produce the lost foods and reduce waste disposal costs as well.

Table 2.1 Estimated losses of some fresh fruits (Adapted from Salunkhe and Desai, 1984)

<table>
<thead>
<tr>
<th>Produce</th>
<th>Wholesale</th>
<th>Retail</th>
<th>Consumer</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strawberries</td>
<td>13.5</td>
<td>5.5</td>
<td>22.2</td>
<td>41.2</td>
</tr>
<tr>
<td>Apples</td>
<td>2.9</td>
<td>2.9</td>
<td>2.4</td>
<td>8.2</td>
</tr>
<tr>
<td>Peaches</td>
<td>12.3</td>
<td>5.8</td>
<td>10.8</td>
<td>28.9</td>
</tr>
<tr>
<td>Valencia oranges</td>
<td>1.4</td>
<td>0.8</td>
<td>3.7</td>
<td>5.9</td>
</tr>
</tbody>
</table>

The researchers have identified six primary sources of food losses: biological and microbiological, chemical and biochemical, mechanical, physical, physiological, and psychological factors (Salunkhe and Desai, 1984).

1) Biological and microbiological: refers to the food damage caused by insects, animals and microorganisms;

2) Chemical and biochemical: refers to undesirable reactions between chemical compounds in the food. For example, the Maillard reaction, fat autooxidation, undesirable enzyme catalyzed reactions, and pesticides or lubricating oil contamination;
3) Mechanical: means damages caused by improper mechanical handling of foods such as spillages, abrasion, bruising, and peeling, etc.;
4) Physical: refers to improper heat treatment, atmosphere and storage conditions.
5) Physiological: refers to changes caused by respiration and transpiration such as senescence in fruits and vegetables;
6) Psychological: human aversion of certain foods because of personal or religious reasons.

2.1.2 Apple postharvest disorders

Apple postharvest disorders affect its quality and consequently reduce its economic value. Producers experience economic losses because they have to dispose of infected apples; consumers are also at risk of eating undetectable infected apples. It is important to recognize common apple postharvest disorders in order to develop measures to prevent losses due to quality deterioration. Generally, apple postharvest disorders can be divided into two categories: disorders induced by physiological or abiotic injuries, and disorders induced by fungi diseases (Porritt, 1982). In the first class, it can again be classified into two classes: external injuries and internal injuries as listed in Table 2.2.

Table 2.2 Apple external and internal physiological injuries (Adapted from Porritt et al., 1982)

<table>
<thead>
<tr>
<th>External injuries</th>
<th>Internal injuries</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO₂ injury</td>
<td>Core browning</td>
</tr>
<tr>
<td>Chemical injury</td>
<td>Flesh browning</td>
</tr>
<tr>
<td>Friction injury</td>
<td>Internal CO₂ injury</td>
</tr>
<tr>
<td>Heat injury</td>
<td>Low-temperature breakdown</td>
</tr>
<tr>
<td>Jonathan spot</td>
<td>Senescent breakdown</td>
</tr>
<tr>
<td>Low-oxygen (alcohol) injury</td>
<td>Vascular breakdown</td>
</tr>
<tr>
<td>Russet</td>
<td>Water core</td>
</tr>
<tr>
<td>Soft scald</td>
<td></td>
</tr>
</tbody>
</table>
Pathogenic fungi, as well as non-pathogenic fungal species, yeast, and bacteria, cause the second apple postharvest disorder: fungi-induced diseases (Knee, 2002). They are naturally present on the surface of apples and can be transmitted during harvest, storage and processing operations. Pathogen spores are the culprits for disease transmission and can be redistributed from infected apples to healthy apples. Apple postharvest diseases caused mostly by fungal infections are listed in Table 2.3.

<table>
<thead>
<tr>
<th>Disease name</th>
<th>Fungi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue mold</td>
<td>Penicillium expansum</td>
</tr>
<tr>
<td>Gray mold</td>
<td>Botrytis cinerea</td>
</tr>
<tr>
<td>Mucor rot</td>
<td>Mucor piriformis</td>
</tr>
<tr>
<td>Bull’s-eye rot</td>
<td>Neofabraea malicorticis anamorph:</td>
</tr>
<tr>
<td></td>
<td>Cryptosporiopsis curvispora</td>
</tr>
<tr>
<td>Alternaria fruit rot</td>
<td>Alternaria spp.</td>
</tr>
<tr>
<td>Cladosporium fruit rot</td>
<td>Cladosporium herbarum</td>
</tr>
<tr>
<td>Side rot</td>
<td>Phialophora malorum</td>
</tr>
<tr>
<td>Black rot</td>
<td>Physalospora obtuse</td>
</tr>
<tr>
<td>Bitter rot</td>
<td>Glomerella cingulata anamorph:</td>
</tr>
<tr>
<td></td>
<td>Collectotrichum gloeosporioides</td>
</tr>
<tr>
<td>While rot</td>
<td>Botryosphaeria ribis</td>
</tr>
</tbody>
</table>

Factors affecting post-harvest apple disorders include ripeness and wounds (Knee, 2002). Apple ripening brings series of changes that may affect an apple’s resistance and susceptibility to diseases. For example, during the ripening process, enzymes break down pectic substances that maintain tissue structure, which also facilitates the fungal breakdown of fruit tissue; the increase in sugars during ripening also provide a food base for fungal infection; at the same time, some substances such as phenolics that can resist
diseases diminish with ripeness. Wound is another important cause for postharvest
disease development because it breaks the skin, which is the natural barrier to infection.

Controlled atmosphere (CA) and low temperature are usually the common
strategies for postharvest management (Wills, 1989). Apples are usually stored at low
temperature such as -1 to 4°C, and this measure poses a first defense line from
postharvest pathogens since low temperature can significantly inhibit the development of
postharvest decay. The high relative humidity of storage rooms is often kept in order to
prevent apple dehydration during long term storage; however, high humidity (99-100%)
usually facilitates development of fungal pathogens. In CA condition, apples are
typically kept in the atmosphere of 2-3% O₂ which is much less than the normal
concentration of 20% in the air and 1% CO₂. However, a tradeoff effect has to be
considered: elevated CO₂ atmospheres may inhibit pathogenic fungi development, but it
may also cause apple internal injury. In modified atmosphere packaging (MAP), the
packaging materials are specially made to have selective permeability. Apple respiration
can be modified according to the gas change in the package.

2.1.3 Fresh produce safety issues

Fruit and vegetable safety has always been the most important issue in the fruit
production industry. There were 428 foodborne disease outbreaks caused by fresh
produce in the United States from 1990-2003 and fresh produce has become the second
largest cause of foodborne diseases following seafood as the No. 1 vehicle of foodborne
diseases transmission (Center for Science in the Public Interest, 2004). For example,
several multistate foodborne disease outbreaks have been reported, including outbreaks
caused by Cyclospora parasites on fresh raspberries, hepatitis A virus on frozen strawberries, and Escherichia coli O157:H7 bacteria in apple cider, lettuce, and alfalfa sprouts (Centers for Disease Control and Prevention, 1998). It was found that most of produce-related foodborne diseases were caused by the following pathogens: Salmonella, Cyclospora, pathogenic Escherichia coli, calicivirus, and Shigella (Beuchat, 1996).

There are some characteristics of fruits and vegetables that make them the ideal vehicle for foodborne disease spread (Beier, 2004): first, normally no previous antimicrobial treatment is taken before they are consumed; second, fresh produce consumption has increased in recent years; third, there has not been any effective standard control methods applicable to all types of fresh produce due to their very different physical structures; fourth, large amount of fresh produce consumed in the US is imported from third world countries that do not have good on-farm food safety programs.

How did pathogens contaminate fresh produce? Previous studies suggested that most produce contamination was caused by human or animal fecal sources. For example, produce could be polluted by irrigation water used to irrigate field crops, by fertilization of soil with untreated sludge, and by infected wildlife. Other non-fecal sources may also cause produce contamination such as polluted environment, infected wounds from workers, and not sanitized equipment (Nguyen-the and Carlin, 1994). A better understanding of major sources of pathogens and the mechanisms of fresh produce contamination can improve the chance of developing measures to prevent fresh produce contamination.
2.2. Apple volatile compounds

2.2.1 Biogenesis of apple aroma

Aroma of fruits and vegetables is the key factor for assessing their quality as well as their identity. The biogenesis of aroma constituents in fruits and vegetables have been studied since the 1950s, and Salunkhe gave an excellent summary on aroma formation for 20 fruits and vegetables (Salunkhe and Do, 1976).

It was found that the aroma of fruits and vegetables originate from basic nutrients such as: carbohydrates (monosaccharide and disaccharides), proteins (free amino acids), fats (triglycerides or their derivatives), vitamins, and minerals (Knee, 2002). These nutrients are produced by photosynthetic and related metabolic activities occurring in the plant. The schematic diagram showing this process is illustrated in Figure 2.1.

There are three modes of volatile production which happen at three different stages of apple production: living plant, postharvest development, and processing phase (Salunkhe and Do, 1976). In the living plant, the aroma compounds are formed by biosynthesis such as photosynthesis. Factors such as climate, soil, macro- and micro-nutrients, etc., affect the photosynthesis and consequently influence aroma formation in the living apple plant. During post-harvest and storage development, apple volatile compounds are formed by its own enzymes, enzymes produced by coexisting microorganisms, and oxidation. Storage temperature, storage time and the ripeness of apples are also critical factors that affect volatile formation for apples. During processing stage, many organic sulfur volatile compounds are formed by heat, fermentation, oxidation, and precursor-enzyme incorporation. The main parameters that affect apple volatile formation include: processing temperature, blending time and speed, holding
time and temperature, pH, and oxygen availability. In processing, physical damage and mechanical stress break intact tissues of apples and release more volatile compounds.

Figure 2.1. Formation of volatile aroma in fruits and vegetables (from Salunkhe and Do, 1976)

2.2.2. Apple volatile compounds

Interests in apple volatile compound detection dates back to the early 20th century, and a great deal of information about apple aroma compounds has been obtained in the last century. More than 300 volatile compounds have been identified in apples and apple products, although only 20-40 compounds are responsible for their aroma (Lopez et al., 1997).
As early as 1920, a number of apple volatile compounds that were believed to be responsible for aroma were identified: acetaldehyde and esters of formic, acetic, and hexanoic acid (Power and Chestnut, 1920). By 1940s, White (1950) at the Eastern Regional Lab (USDA) enlarged the listing of compounds that contribute to apple aroma by finding eight alcohols, four carbonyls and numerous esters. However, the complexity of apple flavor was not realized until the 1950s when Meigh (1956) identified six alcohols, five aldehydes, three ketones, and six esterified acids (Table 2.4). During this time, apple volatile compounds were mainly collected from storage cabinets using cold traps and chemically identified using derivatives.

Table 2.4. Volatile compounds produced by apples (Meigh, 1956)

<table>
<thead>
<tr>
<th>Alcohols</th>
<th>Aldehydes</th>
<th>Ketones</th>
<th>Esterified acids</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethanol</td>
<td>Acetaldehyde</td>
<td>Acetone</td>
<td>C1-C6</td>
</tr>
<tr>
<td>D-2-Methylbutan-1-OL</td>
<td>N-Butanal</td>
<td>Ethyl methyl ketone</td>
<td></td>
</tr>
<tr>
<td>2-Methylpropan-1-OL</td>
<td>Propanal</td>
<td>Methyl propyl ketone</td>
<td></td>
</tr>
<tr>
<td>Methanol</td>
<td>Isobutanal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Isopropanol</td>
<td>Isovaleraldehyde</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C6-Alcohols</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The advent of the gas chromatography and mass spectrometry (GC-MS) technique greatly enhanced researchers’ capability to explore the components in the apple volatile profile. In 1967, 20 previously unreported volatiles were identified in Delicious apple essence (Flath et al., 1967), including hexanol, trans-2-hexenal, and ethyl 2-methyl butyrate. The finding of 4-methoxyallyl benzene, which is characterized by a spice-like aroma, is important due to the importance of its flavor (Williams et al., 1977b). Investigations also found that in order to have a sensory perception, esters must have a molecular weight between 100 and 130 (Dimick and Hoskin, 1981). In addition, C-6
alcohols and aldehydes formed through lipid oxidation make a significant contribution to the odor of apple juice and apple products made from crushed apples. Other factors that affect apple aroma include:

1) Contribution of peel to flavor

Studies found that apple peels produced significantly more volatiles than flesh after 24 hours storage at room temperature. One application of this finding is that apple peels can be used to produce essences. The temperature of ripening was found to be an important factor for volatile output: at 22°C, apple yielded the maximum amounts of esters while ester production was inhibited at 46°C (Guadagni et al., 1971).

2) Postharvest quality indices and flavor

One of the most important motivations for studying apple volatile compounds is that this information can be used for apple postharvest quality indices. Some apple volatile compounds such as ethylene can be used to define maturity and ripeness indices. Ethylene concentration can also be used as an indicator of varietal flavor in Delicious apples; fruit with lower ethylene concentrations also lack varietal flavor (Blanpied and Blak, 1976). Other fruit quality indices such as acidity, and “Magness-Taylor” firmness and stiffness coefficient also have the potential to be correlated with apple volatile compounds (Panasiuk et al., 1980; Sapers et al., 1977) (Table 2.5). These findings provide a theoretical basis the electronic nose to predict apple ripeness and harvesting time.
Table 2.5. Correlation between volatile levels and indices of maturity and ripeness in McIntosh Apples (Adapted from Panasiuk et al., 1980)

<table>
<thead>
<tr>
<th>Volatiles</th>
<th>Stiffness coefficient</th>
<th>Acidity</th>
<th>Magness-Taylor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aldehydes</td>
<td>-0.72</td>
<td>-0.73</td>
<td>-0.67</td>
</tr>
<tr>
<td>Esters</td>
<td>-0.69</td>
<td>-0.68</td>
<td>-0.63</td>
</tr>
<tr>
<td>Total peaks</td>
<td>-0.74</td>
<td>-0.74</td>
<td>-0.69</td>
</tr>
</tbody>
</table>

3) Storage and dehydration effects on apple flavor

Wills ((Wills, 1968) studied the effect of water loss on flavor volatile recovery for apples and found that reduced water concentration caused an increase in ester production. Wills also found that the dehydrated samples exhibited a marked decrease of carbonyls.

4) Sensory evaluation

Sensory evaluation is important to guarantee food products with consistent quality. Williams et al (1977a) studied correlation between sensory evaluation and apple volatile compounds, and found that the specific odor is not produced by a single component; instead, numerous compounds contribute to each descriptor (Williams et al., 1977a). For example, few of the individual compounds were sensed as “apple” when each component was isolated. This finding indicates that the electronic nose is suited for a sensory evaluation because it recognizes a smell pattern by volatile mixtures, rather than individual compounds.

5) Varietals and seasonal difference

Research found that apple volatile compounds are not only different between varieties (Golden Delicious and Granny Smith), but different from seasons within the same variety (Lopez et al., 1997). It is not surprising that the aroma component differences between classes and varieties can provide a characteristic sensorial perception for each type of fruit and variety; however, it is a little unexpected that ethyl propionate
and butyl acetate gave a fruity aroma in Golden Delicious apples in 1993, but ethyl acetate, ethyl propionate and propyl acetate were dominant compounds in 1994.

2.2.3 Diseased apple volatile profiles

Although a large amount of research has been done to identify apple volatile compounds, there is little on disease specific volatile detection in apples. Two Canadian researchers used GC-MS to identify volatile compounds that are specific to four fungi diseases in Cortland and Empire apples: *Botrytis cinerea* Pers., *Penicillium expansum* Link, *Mucor piriformis* Fischer, and *Monilinia* sp (Vikram et al., 2004). Only 34-36 compounds were selected for study among a total of 1081 different peaks in GC-MS. Their research found that dimethyl ether and propanal compounds are specific to *Penicillium* fungi disease in Cortland apples; while acetic acid methyl ester and styrene are specific to *B. cinerea* and *Monilinia* fungi diseases. Similarly, in Empire the compounds 3,4-dimethyl-1-hexene, butanoic acid-2methyl-pentyl ester, and 2-methyl propyl hexanoate are specific to *B. cinerea*, *M. piriformis* and *Monilinia* fungi diseases. This investigation provides solid evidence that these identified volatile compounds can be used for apple disease detection in storage.

2.3. Headspace analysis

Headspace analysis is a technique for the direct analysis of the volatiles in the gas phase above a liquid or solid sample (Rouseff and Cadwallader, 2001). Compared to traditional chemical extraction, adsorption, distillation, and precipitation methods, this method is essentially a non-destructive method with the advantages of simplicity,
rapidness, and without use of solvents. This is especially true in the case of food product volatile analysis since many foods are complex mixtures of oils, fats, and other solids, from which extracting volatiles using traditional methods would be difficult (Pillonel et al., 2002). Although this method has been widely used in many areas including environmental, forensic, pharmaceutical, biomedical, packaging, pesticide, food and flavor areas, the question of how volatiles are partitioned between the various components in food and many beverages is still not well understood (Rouseff and Cadwallader, 2001).

However, one obvious weakness of headspace analysis is that headspace extraction has a bias towards the high volatile compounds even though those compounds may be only minor components in the sample. Cautions should be made that this method sometimes cannot produce representative extracts of the sample, and other remediation methods such as liquid-liquid extraction should be used to correct this composition bias.

There are two types of headspace analysis: static and dynamic headspace. In dynamic headspace, the sample is purged with an inert gas until all volatile compounds are removed from the sample (Figure 2.2). The gas effluent purged from the sample vessel then passes through a trap which contains an adsorbent or is cooled to low temperature. When this extraction is complete, the extracted volatile compounds are then purged with the carrier gas and released by rapid heating for subsequent analysis.
Since dynamic headspace analysis can be used to analyze samples when gas concentrations are low, it has been extensively used for off-flavor analysis including peanut off flavor, cardboard storage off-flavor, and fruity off-flavors in milk (Rouseff and Cadwallader, 2001). However, this method should be used for analysis of highly volatile components because it is heavily skewed to the more volatile compounds.

Static headspace analysis is also called equilibrium headspace which requires volatiles to reach equilibrium in which state the volatiles evaporate into the gas phase at the same rate as the volatiles condense into the liquid phase. As shown in Figure 2.3, typically, a sample is pretreated with thermostat to reach equilibrium state, then the equilibrated headspace is conducted by inert carrier gas for further analysis.
Similarly, those volatile compounds with a lower boiling point will have the greater partial pressure and hence have a better chance to be extracted in the headspace. Thus, static headspace is also biased toward the more volatile components. Unlike dynamic headspace analysis, it is difficult to detect low concentration samples with this method due to the fixed volume.

2.3.1 SPME: solid phase micro extraction

SPME is a static headspace technique which is usually used with GC-MS. It uses a thin silica fiber coated with chromatographic material which can be either introduced into the headspace (headspace mode) or directly into the liquid sample (direct extraction mode) (Figure 2.4). Volatiles absorbed on the fiber are later thermally desorbed in the injection port of a gas chromatograph. The SPME was firstly developed in early 1990s as a preconcentration technique for water pollutants analysis (Zhang and Pawliszyn, 1995). During the past decade, its application has far exceeded its original purpose with 20%
food and botanical applications and 19% clinical and forensic applications (Pillonel et al., 2002).

Figure 2.4. SPME schematic diagram (Adapted from Pawliszyn, 2001)

The fiber coating is an important factor influencing the effect of SPME. The six most commonly used fiber coatings can be divided into two classes: 1) the pure liquid polymer coating such as polydimethylsiloxane (PDMS) or polyacrylate (PA) and 2) the mixed film, containing liquid polymer and solid particles such as carboxen-PDMS, Divinylbenzene (DVB)-PDMS, carbowax-DVB and DVB-Carboxen-PDMS (Pillonel et al., 2002). These fibers and their applications are listed in Table 2.6. Pure PDMS was designed to extract pollutants from aqueous samples so that it is strongly hydrophobic. The mixed films have the absorption properties of both the liquid polymer and porous particles. In this research, the combination of DVB, Carboxen and PDMS was used due to its absorption properties of volatile and semi-volatile compounds.
Table 2.6 Different SPME fibers and their recommended application fields (Adapted from Pillonel et al., 2002)

<table>
<thead>
<tr>
<th>Fiber</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDMS 100um</td>
<td>Volatiles</td>
</tr>
<tr>
<td>30um</td>
<td>Nonpolar semi-volatiles</td>
</tr>
<tr>
<td>7um</td>
<td>Nonpolar high molecule weight compounds</td>
</tr>
<tr>
<td>PA</td>
<td>Polar semi-volatiles</td>
</tr>
<tr>
<td>PDMS/DVB</td>
<td>Volatiles, amines, nitroaromatic compounds</td>
</tr>
<tr>
<td>Carbowax/DVB</td>
<td>Alcohols and polar compounds</td>
</tr>
<tr>
<td>Carboxen/PDMS</td>
<td>Gases and low molecular weight compounds</td>
</tr>
<tr>
<td>DVB/Carboxen/PDMS</td>
<td>Volatile and semi-volatile flavors and odors</td>
</tr>
</tbody>
</table>

Sampling conditions are also important in SPME applications. Generally, ambient headspace and immersion techniques work well for nonpolar analytes extraction, whereas heated headspace and immersion work best for polar analytes extraction. In addition, the headspace volume should be kept as small as possible since the extraction yield increases with decreasing headspace volume. The addition of salt and stirring liquid samples also significantly improve the extraction rate of analytes (Pawliszyn, 2001).

Although SPME headspace analysis selectively increases the apparent concentration of certain compounds and decreases the apparent concentration of others, it is still an attractive method for many analysts due to its speed, simplicity and ease use.

2.3.2 GC-MS

Gas chromatography-mass spectrometry (GC-MS) is an analytical instrument to identify individual volatile compounds in a mixture of chemicals (Oehme, 1998). As its name suggests, it is actually composed of two components: the first part, gas chromatography (GC), separates the components of a mixture and the second part, mass
spectroscopy (MS), characterizes each of the components individually (Figure 2.5). By combining the two techniques, volatile compounds can be evaluated both qualitatively and quantitatively (Ettre, 2001).

As shown in Figure 2.5, GC consists of a carrier gas tank, injection port, capillary column, detector, and GC-MS interface (Oehme, 1998). The separation is achieved when the sample mixture is injected into a mobile phase (carrier gas such as helium) and conducted through the stationary phase located within capillary column (usually 0.25 mm diameter). By changing pressure of the mobile phase and temperature of the stationary phase which is composed of chemicals that can selectively attract components in a mixture, different chemicals can be separated.

Apart from the stationary phase interaction, the temperature of the capillary column is another force for separation. The capillary column temperature can be
controlled and increased gradually by a computer program, which helps separation. As temperature increases, those compounds with lower boiling points come out of the column sooner than those with higher boiling points. The separated compounds then enter a detector which will create a signal when a compound is detected. The retention time is defined as the time between the injection and the elution, and it can be used as the characteristic for each volatile compound. Commonly used GC detectors include: the flame ionization detector (FID), thermal conductivity detector (TCD or hot wire detector), electron capture detector (ECD), atomic emission detector (AED), etc. The computer then generates a chromatogram from the signal as shown in Figure 2.6 (a).

However, compounds with similar properties usually have the same retention time (RT), so to make accurate decisions about the identity of a compound, mass spectrometry is needed. MS is composed of an ionization chamber, mass analyzer, and detector. The separated compounds are broken into charged molecular ions by a stream of electrons in the ionization chamber; then, they enter the mass analyzer filter which allows only certain M/Z (mass to charge ratio) ions to pass. The computer scans different M/Zs one at a time until a range of M/Zs are covered, and a mass spectrum is produced. As shown in Figure 2.6 (b), the x-axis represents the M/Z ratio and the y-axis represents the signal intensity for each of the molecular ions detected during the scan. The mass spectrum produced by a given chemical compound is essentially the same every time, so it can be used as the fingerprint for the molecule to identify the compound.
2.4 Biological olfaction

Humans can recognize approximately 10,000 scents and many animals, such as bloodhounds, have far superior olfactory system than humans (Axel, 1995). However, the olfactory system is the most complicated and least understood sense among human’s five senses (smell, taste, vision, hearing, and touch). Smell is regarded as human’s most evocative sense: the perception of certain odors induces our specific thoughts, memories, and behaviors. Although the sense of smell is often thought as a luxury or an aesthetic sense for humans, it is a primal sense for most animals. From an evolutionary standpoint, it is not only one of the most ancient senses, but one of the most important senses for most animals (Pearce et al., 2003). Olfaction enables most animals to identify food, predators and mates with both sensual pleasure and warning of danger, which is essential for their survival (Leffingwell, 2002). During the past fifteen years, scientists have made great efforts to understand the question of “how do we smell”, and provided a sound footing on the understanding of smell with the symbolic winning of the Nobel Prize by two American scientists by their work of “unraveling the enigma of smell” in 2004 (Altman, 2004). Since machine olfaction is inspired by the biological olfactory system, it
is necessary to have an overview and general understanding of biological olfactory even for an engineer working with the electronic nose. In this section, a general review of biological olfaction at the physiological, molecular and computational level was presented.

2.4.1 Physiology of biological olfaction

For humans, the detection of odors happens at the olfactory region, which is a small area of about 2.5 square centimeters but containing in total approximately 50 million primary sensory receptor cells (Leffingwell, 2002). This area is located in the roof of two nasal cavities of human nose (Figure 2.7).

The odorant must have some water solubility, a sufficiently high vapor pressure, low polarity, good lipophilicity, and surface activity in order to act on the nasal sensory tissues. In addition, no odorant was found to have a molecular weight greater than 300 (Leffingwell, 2002).

Olfactory epithelium plays an important role in the detection of odors. On one end of the olfactory epithelium, hairlike sensors called cilia extend outward into a layer of mucous and are in direct contact with the air. This mucous layer bathes the surface of the receptors at the epithelium surface and assists in transporting the odorant molecules that are soluble in the mucous to the upper olfactory system. Each olfactory receptor neuron has approximately 8-20 cilia that are 30-200 microns in length. The olfactory cilia functions to start molecular reception with the odorant and sensory transduction.
Within the olfactory epithelium, there are millions of olfactory receptor neurons. Unlike other neurons that die and never regenerate again, olfactory receptor neurons are continuously regenerated by its neural stem cells (Axel, 1995).

On the other end of the epithelium, fibers known as axons are bundled in groups of 10-100 to penetrate the ethmoidal cribriform plate of bone and reach the olfactory bulb of the brain. The axons converge to glomeruli which are connected in groups with a synaptic structure and converge into mitral cells. This convergence increases the sensitivity of the olfactory signal and explains why olfactory thresholds measured psychophysically in humans are often lower than those of single cell recordings (Firestein, 2001). From the mitral cells, the signal are sent directly to the high level of the central nervous system in the cerebral cortex, the area of the brain that controls thoughts and behaviors.

Figure 2.7. Anatomy of human olfactory system (Adapted from Leffingwell, 2002)


2.4.2 Molecular understanding of olfaction

After getting a basic anatomic understanding of the nose and olfactory system, it is important to understand the mechanism of the olfactory system at the molecular or genetic level. The 2004 Nobel Prize in Physiology or Medicine was awarded to Linda Buck and Richard Axel who made a seminal breakthrough and discovery of a large family of genes that encode odor receptors (Altman, 2004). Their findings revealed that there are as many as 1,000 genes for odor receptors in the mammalian genome, representing about 2% of all genes in the body and making it by far the largest gene family in the entire genome (Buck and Axel, 1991). This fact reflects the significance of the olfactory system for the survival and reproduction of most animals. Using gene cloning and molecular hybridization techniques, they found that each sensory neuron expresses only one receptor and is therefore functionally distinct.

How do humans and mammals detect 10,000 odors with only 1000 odor receptors? Many scientists have put forward different models for odor decoding (Firestein, 2001). It is generally believed that most odor molecules are recognized by more than one receptor and most receptors recognize several odors. Based on experimental evidence, one model is mostly accepted and depicted in Figure 2.8. As shown in Figure 2.8, the recognition of an odorant molecule is realized by different patterns of activated receptors, and the different extent to which they are activated, as shown by the shade of color (black means no activation). Some receptors may be best suited for one odorant molecule, but other receptors that are able to recognize some features of the molecule would participate in the discrimination of that compound (Firestein, 2001).
According to this model, theoretically, mammals should be able to detect billions of odorants by the vast possibilities of 1000 receptor combinations. This number far exceeds the number of odors that mammalians can actually detect. One possible explanation for this is that from an evolutionary standpoint, animals develop only those olfactory recognition capabilities for those odors that are biologically important to their survival and reproduction (Axel, 1995).

Scientists also found that there are two different olfactory systems in most mammals, including humans (Firestein, 2001). One is the main olfactory system which is the primary sense used by animals to find food, detect predators and prey, and mark territory; another accessory olfactory system which is physically separated from the main olfactory system is called the “sexual nose” or vomeronasal organ. The vomeronasal
organ is responsible for one specific task: to detect the pheromones that govern reproductive and social behaviors. The evidence that the sequence of amino acids in the receptors of two olfactory systems are different suggests that the two systems may have evolved independently of each other (Axel, 1995).

### 2.4.3 Computational olfaction

How the olfactory cortex decodes the signals provided by the olfactory bulb and makes sense of smell is one of the central and most elusive problems in neurobiology (Axel, 1995). Towards this end, many researchers put enormous efforts into the area of computational olfaction (Ermentrout et al., 2001; Gelperin, 1999; Gelperin, 2006; Hopfield, 1999).

The most fundamental work in this area was done by John Hopfield (1999). He constructed a model of the patterns of olfactory receptor responses to explain how animals perform the following four basic tasks (Hopfield, 1999):

1) Odor memory and recognition. A scent can be recognized later by previously stored memory.

2) Background elimination. A known odor can be identified when it is thoroughly mixed with an unknown background.

3) Component separation. The component odors and their intensities can be identified and separated when they are thoroughly mixed.

4) Odor separation. The odors of multiple unknown objects can be separated when they are mixed by air turbulence.
His model explains why the large number of receptors (~1,000) is important to implement computational strategies that would not be effective in a stimulus space of low dimension. He proposed that action potential timing and adaptation are two important mechanisms for effective neural computation.

2.5 Machine olfaction

2.5.1 Electronic nose

Because of the power of human olfactory systems, trained panelists were usually used for applications ranging from food quality inspection, perfumes and cosmetics aroma evaluation, to agricultural product odor detection. However, the main drawback of human sensory panels is that they can not detect volatile compounds without odor, while many such compounds are very important indicators for agricultural produce quality. Humans are also subjective and can only work for a short period of time. Analytical equipment such as gas chromatography- mass spectrometry (GC-MS) are effective in detecting constituents in volatile mixtures, but they are relative expensive and not portable, which make it not appropriate for certain online applications (Gardner and Bartlett, 1994). Consequently, the demand to develop fast, low-cost, sensitive, and non-destructive volatile-detecting sensors to classify different smell patterns instead of constituents from volatiles is enormous.

The development of the electronic nose meets this demand. In the early 1980s, Persaud and Dodd (Persaud and Dodd, 1982) at Warwick University developed a prototype of electronic nose as an intelligent chemical sensor array system to classify
odors. Since then, the development of the electronic nose has been growing rapidly and many conferences were held on this topic (Gardner and Bartlett, 1994).

The electronic nose is also called an artificial nose or mechanical nose. According to Gardner et al. (1994): “An electronic nose is an instrument which comprises an array of electronic chemical sensors with partial specificity and an appropriate pattern-recognition system, capable of recognizing simple or complex odors”.

The electronic nose system typically consists of two main components: gas sensors and pattern recognition software. A gas sensor is a device that responds to a wide range of volatile molecules in gases and is capable of converting a chemical quantity into an electrical signal. Typically, the gas sensors are based on different principles including electrical, optical and mass change. Generally, the commercial electronic noses can be divided into two categories based on their working temperature: hot sensors which operate at elevated temperature (100-500°C) and cold sensors which work at ambient temperature (Mielle, 1996). In Table 2.7, two categories of gas sensors are compared:

<table>
<thead>
<tr>
<th>Property</th>
<th>“Cold” sensors</th>
<th>“Hot” sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power consumption</td>
<td>Low (a few µW)</td>
<td>High (50-800 mW)</td>
</tr>
<tr>
<td>Response time</td>
<td>Slow (20-40s)</td>
<td>Fast (0.5s-a few s)</td>
</tr>
<tr>
<td>Selectivity</td>
<td>Good (SAW and BAW sensors can be enatioselective)</td>
<td>Poor (different for each type; can be adjusted by temperature modulation)</td>
</tr>
<tr>
<td>Carrier gas</td>
<td>May be inert (avoid oxidation of samples)</td>
<td>Synthetic air (must contain O₂)</td>
</tr>
<tr>
<td>Availability</td>
<td>Laboratory-made (poor reproducibility)</td>
<td>Commercially available (over 70 types)</td>
</tr>
<tr>
<td>Lifetime</td>
<td>Not guaranteed to be more than 12 months (typically 18 months)</td>
<td>More than 5 years in normal use</td>
</tr>
<tr>
<td>Price</td>
<td>Expensive for single sensors; very expensive for arrays</td>
<td>Low for single sensors; medium for arrays</td>
</tr>
</tbody>
</table>
An ideal gas sensor should possess the following characteristics (Mandelis and Christofides, 1993; Schaller et al., 1998):

a. High sensitivity to chemical compounds of interest and low sensitivity to humidity
b. Chemically selective: they must respond differently to different volatiles
c. Reversible
d. Noncontaminating and nonpoisoning
e. Short reaction and recovery time
f. Robust and durable
g. Easy calibration
h. Simple operation
i. Small dimension
j. Low noise and low cost

Data analysis (pattern recognition) is another very important component in the machine olfaction, and due to its importance and independence, it is introduced in section 2.6 separately.

A generic architecture of an electronic nose was proposed (Gardner and Bartlett, 1994) (Figure 2.9): an odor reacts with the sensor array which converts the chemical reaction into an electrical signal. Further data processing and classification is made by a pattern recognition engine.
It is worthy to mention that the term “electronic nose” has been admitted by many researchers because it possesses two basic characteristics of the human olfactory system: first, the electronic nose gas sensors and pattern recognition software are conceptually analogous to human olfactory receptors and the brain, respectively; second, the electronic nose recognizes the odor by an overall smell pattern instead of identifying each different constituent of an odor, which is similar to the human smell principle. Nevertheless, the electronic nose does not work in the same way as the human nose does, and its sensitivity is still far less powerful than the human nose, so many researchers argued that it is not appropriate to suggest such a strong link with human olfaction and it is better to use term “flavor sensors”, or “aroma sensors” (Mielle, 1996). In this dissertation, the term of electronic nose was used without quotation to follow the convention.
2.5.2 Gas sensors

Metal oxide semiconductor sensors

Metal oxide semiconductor gas sensors (MOS), which were invented by Taguchi in 1960s, are one of the earliest commercially available gas sensors with more than 70 types and many providers (Schaller et al., 1998). Basically, MOS gas sensors detect the conductivity change caused by the adsorption of gases and subsequent surface reactions. There are two types of MOS gas sensors based on the metal oxide coating film: n-type semiconductors which are composed of zinc, tin or iron oxide and respond to reducing compounds; p-type semiconductors which are composed of nickel oxide or cobalt oxide and respond to oxidizing compounds such as O₂, NO₂ and Cl₂ (Mielle, 1996).

The working mechanism of MOS gas sensors follows two steps: first, oxygen from air is adsorbed on the surface of metal oxide semiconducting film and oxygen traps free electrons from the semiconductor, which increases the resistance of the semiconductor; second, the electrons are freed by means of reaction of the oxygen and reducing gas, which reduces the resistance of the semiconductor. Hence, the presence of the reducing compounds at the surface of the semiconducting film increases the conductance in a nonlinear manner (Figure 2.10) (Pearce et al., 2003; Schaller et al., 1998).
The selectivity of a metal oxide film to different chemical compounds can be modified by doping with different catalytic metals, and changing the working temperature within a range of 50-400 °C (Sberveglieri, 1992). Due to its high working temperature, MOS gas sensors are insensitive to humidity and usually can be used for a longer time (5 years) than other gas sensors, such as conducting polymer sensors. However, the main weaknesses of MOS gas sensors include: they are extremely sensitive to ethanol which may blind the sensor to detect other compounds; they may be poisoned by irreversible binding by compounds such as sulfur compounds or weak acids such as vinegar and cheeses; their high working temperature prevents them from being used in an environment containing large amounts of flammable chemicals (Mielle, 1996).

MOS gas sensors can be manufactured in a large scale, which guarantees repeatability between different sensors and reduces the sensor costs. The main providers for this gas sensor include Figaro Engineering Inc. and New Cosmos Electric Co., Ltd (Pearce et al., 2003).
Conducting polymer gas sensors

The conducting polymer gas sensor is another widely used and commercially available gas sensor. Similar to the MOS gas sensor, the conducting polymer gas sensor also identifies odors by detecting sensor resistance change, although its operating mechanisms are more complex and have not been well understood so far (Mielle, 1996).

Basically, these types of sensors are made of three main components: a substrate, a pair of gold-plated electrodes, and a conducting organic polymer layer. When the sensor is exposed to an analyte, the conducting organic polymer film swells, which causes the increase in resistance because the conductive pathways through the material are disrupted (Figure 2.11). Typically, this type of sensor consists of a sensor array, and each sensor can respond to a variety of vapors with partial overlapped selectivity. An array of sensors which contain different polymers produce a distinct fingerprint for each odor, due to their different swelling properties. The pattern of the resistance change over the sensor array provides the evidence for qualitative classification of different smell patterns, and the amplitude of resistance change gives quantitative evidence for vapor concentration (Sberveglieri, 1992).

Polymers are relatively cheap and a large number of polymers with different functions are available, which can be used to fabricate different selective sensors. Commonly used polymers include polypyrroles, polyanilines, and polythiophene (Schaller et al., 1998). Another main advantage of using conducting polymer sensors is that they can be operated in ambient room temperature. However, their relatively slow response (20-40s) and drift over time are inherent drawbacks which prevent them from being used for rapid analysis and obtaining repeatable results over long periods of time.
One main provider of CP gas sensors is Cyrano Science (Smith Detection, Herts, UK) and their Cyranose electronic nose, which was used in this research.

Figure 2.11. Working principle of conducting organic polymer gas sensor (Adapted from http://nsl.caltech.edu/resnose.html, 2006)

**Surface acoustic wave (SAW) sensors**

A surface acoustic wave sensor detects volatile compounds by sensing the mass change based on a piezoelectric effect. Piezoelectric crystals have a very stable resonance radiofrequency which propagates on the surface of the crystal. The radiofrequency oscillation in SAW, known as “Rayleigh waves”, is adjusted by the mass change due to the presence of adsorbed volatile compounds (Sberveglieri, 1992) (Figure 2.12 (a)). Usually, the characteristic frequency is in the range of 100-1000 MHz. As shown in Figure 2.12 (b), the piezoelectric materials are made of ZnO or lithium niobate and they are coved by two pairs of interdigitated combs which are typically made of aluminum and used as wave emitters and reflectors. The sensing membrane can be chemically modified to adjust the sensor’s specificity. The frequency change $\Delta f_r$ of the SAW due to the adsorption of vapor and consequent mass change can be expressed as (Pearce et al., 2003):
\[ \Delta f_V = \frac{\Delta f_p c_V K_p}{\rho_p} \]  

(2.1)

where \( \Delta f_p \) is the change in frequency caused by the polymer membrane itself, \( c_V \) is the vapor concentration, \( K_p \) is the partition coefficient and \( \rho_p \) is the density of the polymer membrane.

SAW sensors are highly sensitive but noise caused by analog electronics easily interferes with them. Other drawbacks include the difficulty to replace sensors and a drift of response due to temperature fluctuations (Mielle, 1996).

---

Figure 2.12. (a), Cross-sectional view of Raleigh wave operation. (b), Surface acoustic wave sensor schematic diagram (Adapted from Sberveglieri, 1992 and Mandelis, 1993)
2.5.3 Optical gas sensors and others

Optical gas sensors detect odors by sensing the optical properties such as absorbance, reflectance, fluorescence or chemiluminescence when a light source excites the gas. A typical optical gas sensor consists of four components (Pearce et al., 2003): a light source, suitable optics, a detector and the sensor. Figure 2.13 shows a common configuration of an optical fiber chemosensor using fluorescence. At one end of the fiber, an analyte-sensing element is deposited on fibers with a diameter of 2 um and imaging bundles with a diameter of 500 um. Analyte gases can interact with the sensing element and generate optical property changes such as intensity change, spectrum change, and wavelength shift in fluorescence. These changes can be detected at the other end of the fiber and the responses reflect not only the nature of the vapor mixture but the concentration of gases (White et al., 1996).

Optical gas sensors have advantages such as being free from electromagnetic interference, extremely low light attenuation, and very high sensitivity (for fluorescence). However, they are also more expensive, more complex, and sometimes suffer from photodegradation on their fluorescent indicators (White et al., 1996).

Although they have not been commercialized, optical gas sensors are being actively researched and they are gaining much publicity. An example is Tufts University’s optical electronic nose, notable for the biological inspirations behind the sensors (Dickinson et al., 1996). Other research on surface plasmon resonance (SPR) and colorimeter coupled optical fibers is also underway (Ballantine et al., 1992; Nelson et al., 1996).
There are many other efforts to make artificial noses based on different principles, enhancing their capability to mimic a human nose. For instance, an electronic nose was made to differentiate fruit cultivars by utilizing only 1 s odor sniffing samples, and it achieved encouraging results (Gelperin et al., 1999; Ouellette, 1999). Researchers also tried to combine biotechnology (single stranded DNA) and nanotechnology (single wall nanotube) to develop new generation of artificial noses (Staïi et al., 2005), which may open a new door for gas sensing technology.

2.6 Pattern recognition algorithms

As pointed out in section 2.5 of this chapter, the machine olfactory system tries to mimic the human nose which senses smell and makes decisions based on the brain’s recognition. Essentially, the machine olfactory system is a combination of chemical sensors and a pattern recognition system that can discriminate different odor patterns. Thus, the pattern recognition part is indispensable in machine olfaction and it is important to have a general understanding of the algorithms used in pattern recognition. In this
section, a general theoretical background of popular statistical and non-parametric algorithms is presented.

2.6.1 Overview of pattern recognition

Over the past tens of millions of years, human beings have evolved highly sophisticated neural and cognitive systems for pattern recognition tasks, which have been crucial for our survival. In the past century, scientists and engineers have tried to design and build machines with certain capabilities of perception such as automated speech recognition, fingerprint identification, and DNA sequence identification (Duda et al., 2000). In general, a pattern recognition system consists of five components: sensing, segmentation, feature extraction, classification, and post-processing (Duda et al., 2000), as shown in Figure 2.14.
Figure 2.14. A typical pattern recognition system (Adapted from Duda, 2001)

1) Sensing. It is the first step for a pattern recognition system to sense the objects that are of interest. Typically, a transducer is used, such as a camera (for an imaging system), and chemical sensors (for an electronic nose). It is important since the downstream classification is very dependent upon the precision and accuracy of the source; characteristics including bandwidth, resolution, sensitivity, distortion, signal-to-noise ratio, latency, etc.

2) Segmentation and grouping. Segmentation is one of the deepest problems in pattern recognition, whose purpose is to isolate sensed objects from the background or from other objects.
3) Feature extraction. It measures object properties and extracts certain useful features which are used for classification. Good features should represent the object’s innate characteristics well, which make the later classification an easier job; on the other hand, a powerful classifier would not need sophisticated feature extractors. The purpose of feature selection is to select the most valuable and relevant features which are less sensitive to noise.

4) Classification. It is the central task for a pattern recognition system, which is to use the feature vectors provided by step 3 to assign the object to a category. Many classification algorithms are available, including statistical methods and non-parametric methods (e.g., artificial neural networks) which will be introduced in detail later in this section.

5) Post processing. Classification results are often used to recommend actions and each action has certain costs which can be either defined as classification error rate—the most used and simplest measure—or the total expected cost, called risk, that incorporates the knowledge about the classification problem itself. Another strategy in post processing is to use multiple classifiers, which are consistent with the idea of multisensor data fusion. This will be introduced later in this chapter.

### 2.6.2 Artificial neural networks

The human brain can easily perform tasks such as speech recognition, face identification, and optical character recognition, and this capability dwarfs even the fastest supercomputer (Hertz et al., 1991). The superior power of the biological nervous system has motivated neuron scientists, mathematicians, physicists, and computer
scientists to design a computational prototype called artificial neural networks (ANNs). These are different from the traditional von Neumann paradigm that is based on sequential instructions. The ANNs have the biological nervous system’s superior characteristics in pattern recognition including robustness, faulty tolerance, flexibility, high parallelism, and little power requirement (Hertz et al., 1991).

Although the earliest thought about ANNs arguably originated from Aristotle, the first serious and complete published model of ANNs was made by McCulloch and Pitts (1943). In the McCulloch-Pitts neuron model (Figure 2.15), the binary-threshold unit fires if the weighted sum $\sum_j w_{ij} n_j$ of the inputs reaches or exceeds the threshold $\mu_i$.

![Figure 2.15. Schematic diagram of a simple neuron model (Adapted from Hertz, 1991)](image)

This model calculates a weighted sum of its inputs from different units, and produces a one or a zero after comparing a certain threshold $\mu_i$:

$$n_i(t+1) = \Theta(\sum_j w_{ij} n_j(t) - u_i)$$

(2.2)

where $n_i$ is either 1 or 0, representing the state of neuron $i$ as firing or not, at time step $t$. The weight $w_{ij}$ represents the strength of the synapse connecting neuron $j$ to
neuron $i$. It can be positive or negative corresponding to an excitatory or inhibitory synapse respectively. It is zero if there is no synapse between $i$ and $j$. $\Theta(x)$ is the unit step function:

$$\Theta(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{otherwise}. \end{cases}$$  \hfill (2.3)

This model was generalized in the form shown below, where the unit step function $\Theta(x)$ is replaced by a nonlinear transfer function $g$, and $n_i$ is a continuous value.

$$n_i = g\left(\sum_j w_{ij} n_j - u_i \right)$$  \hfill (2.4)

**BP network and gradient decent learning algorithm**

Although the earliest neural model was developed in the 1940s (McCulloch and Pitts, 1943), its development is not smooth. Because the early Perceptrons could not solve the exclusive or (XOR) problem, it casts the doubt of the capability of a neural network and the development of artificial neural networks had been stagnant for almost 20 years (Hertz et al., 1991). In 1980s, the neural computational community was active again and many new algorithms including the Hopfield network and the Kohonen network were developed (Bishop, 1995). However, among those sophisticated models, one of the most important and widely used ANNs is the feed forward and error-back-propagation networks which are based on gradient decent learning algorithms (Bishop, 1995).

The gradient decent algorithm is introduced based on a three layer network in Figure 2.16. The sum-squared error is defined:
\[ E = \frac{1}{2} \sum_{p} (d^p - y^p)^2 \]  \hspace{1cm} (2.5)

where \( d^p \) is the expected true value; \( y^p \) is the network output;

We also define:

\[ net_j = \sum_i w_{ij} y_i \]  \hspace{1cm} (2.6)

\[ y_j = f(net_j) \] \( f \) is differentiable transfer function;  \hspace{1cm} (2.7)

![Figure 2.16. A three layer feed forward error back-propagation model](image)

For example:

\[ f = \tan \text{sig}(n) = \frac{2}{1 + e^{-2n}} - 1 \]  \hspace{1cm} (2.8)

From Equation (2.5-2.7), we can get:

\[ \delta_k = \frac{\partial E}{\partial net_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial net_k} = (y_k - d_k) f'(net_k) \]  \hspace{1cm} (2.9)
Since we do not have an expected true value $d_j$ as shown in Equation 2.5 to directly compute $\frac{\partial E}{\partial y_j}$, it can be calculated by the following equation:

$$\frac{\partial E}{\partial y_j} = \sum_k (\frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial y_j}) = \sum_k (\delta_k \cdot w_{jk}) \tag{2.11}$$

Hence, we get weight update equations:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{jk}} = \delta_k \cdot y_j \tag{2.12}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}} = \delta_j \cdot y_i \tag{2.13}$$

$$\Delta w_{jk} = -\eta \cdot \frac{\partial E}{\partial w_{jk}} \tag{2.14}$$

$$\Delta w_{ij} = -\eta \cdot \frac{\partial E}{\partial w_{ij}} \tag{2.15}$$

The gradient decent algorithm is named because the network weights are moved along the negative of the gradient of the performance function. The error back-propagation was named because the current layer weight update $\Delta w_{ij}$ depends on the previous layer weight state $\delta_k$ and $w_{jk}$.

A more elaborate form of gradient descent algorithm is shown in Equation 2.16 by adding a momentum term which is used to expedite the convergence process.

$$\Delta w_{ij}(t) = -\eta \cdot \frac{\partial E}{\partial w_{ij}(t)} + \alpha \cdot \Delta w_{ij}(t-1) \tag{2.16}$$
PNN and radial basis networks

The probabilistic neural network (PNN) is essentially a radial basis network. One of the strengths of this network is that it is easy to design and efficient to execute, which takes only a fraction of time it takes to train a BP network (Demuth and Beale, 2001). It is also suitable for classification problems since its output is either 1 or 0. Figure 2.17 is a radial basis network model and the transfer function it employs.

\[
\text{net input } n = \| \text{dist} \| \cdot w - p + b
\]

\[
\| \text{dist} \| = \langle \text{p}, w \rangle
\]

\[
a = \text{radbas}(\| w - p \| b)
\]

Figure 2.17. Radial basis network model (a) and radial basis function (b). (Adapted from Demuth and Beale, 2001)

The net input \( n \) to the radial basis transfer function is the vector distance between its weight vector \( w \) and the input vector \( p \), multiplied by the bias \( b \) (Equation 2.17). The \( \| \text{dist} \| \) produces a dot product of the input vector \( p \) and the single row input weight matrix \( w \).

\[
a = \text{radbas}(\| w - p \| b)
\]  

(2.17)

The radial basis transfer function is defined in Equation 2.18 and its function is shown in Figure 2.17 (b). It has a maximum value of 1 when its output is 0 and its value
decreases when the distance between $w$ and $p$ increases. By doing this, a radial basis neuron is virtually a detector that produces 1 whenever the input $p$ is identical to its weight vector $w$. In Equation 2.18, $n$ is the distance between $p$ and $w$. When it is greater than 0.833, the output of the radial basis function is less than 0.5 which is negligible.

$$f(n) = e^{-n^2}$$  \hspace{1cm} (2.18)

The bias term $b$ adjusts the sensitivity of the radial basis neuron. For instance, if $b = 0.1$, the radial basis function outputs 0.5 for any input vector $p$ at vector distance of 8.326 (0.8326/b) from its weight vector $w$.

**Learning vector quantification and competitive learning**

Learning vector quantification network is essentially a competitive network (Hagan et al., 1996). The architecture for a competitive network is shown in Figure 2.18 (Demuth and Beale, 2001).

![Figure 2.18. Competitive network model (Adapted from Demuth and Beale, 2001)](image)

The $\| ndist \|$ term is the negative distance between the input vector $p$ and the input weight matrix $IW$ (or $w$); the net input $n$ is computed by finding the negative
distance between input vector \( p \) and the weight vectors and adding the biases \( b \), as shown in Equation (2.19). The reason to add a bias term \( b \) is to give neurons that only win the competition rarely an advantage over neurons that win often.

\[
a = \text{compet}(-\|IW - p\| + b)
\]  

(2.19)

The competitive transfer function works by finding the index \( i^* \) of the neuron with the largest net input, and setting its output to 1. All other outputs are set to 0.

\[
a_i = \begin{cases} 1, & i = i^* \\ 0, & i \neq i^* \\ \end{cases}, \text{ where } n_{i^*} \geq n_i, \forall \ i, \text{ and } i^* \leq i, \forall n_i = n_{i^*}
\]  

(2.20)

The weights of the winning neuron are updated with the Kohonen learning rule. If the \( i^{th} \) neuron wins, the elements of the \( i^{th} \) row of the input weight matrix are adjusted as shown below.

\[
w(q)_i = w(q - 1) + \alpha(p(q) - w(q - 1))
\]  

(2.21)

Where \( \alpha \) is the learning rate and \( p \) is input vector.

According this learning rule, the neuron whose weight vector was closest to the input vector is updated to be even closer, which is also called winner-take-all strategy. This induces an effect that the winning neuron is more likely to win when a similar vector is presented the next time, and less likely to win when a very different vector is presented. As more and more training vectors are presented, eventually, each neuron adjusts its weights to a certain input vector and the competitive network can be used to classify new pattern vectors.
2.6.3 Statistical methods (reviewed to here)

Principal component analysis

The principal component analysis (PCA) is one of the most useful techniques for multivariate data analysis, which is based on Karhunen-Loeve expansion and used to reduce the dimension of the problem by projecting the data from a p-dimensional space to a k-dimensional space (k<p). The following derivations are reiterated from three sources (Anderson, 2003; Jolliffe, 2002; Pearce et al., 2003).

Suppose we have a multivariate data matrix with p vectors (variables):

\[
X = \begin{pmatrix}
X_1 \\
X_2 \\
\vdots \\
X_p
\end{pmatrix}
\]  

(2.22)

with variance-covariance matrix

\[
\text{var}(X) = \Sigma = \begin{pmatrix}
\sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1p} \\
\sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{1p} & \sigma_{2p} & \cdots & \sigma_p^2
\end{pmatrix}
\]  

(2.23)

We can use eigenvectors \( e_i \) to decompose and recombine each variables to form the principal component \( Y_i \).

\[
Y_i = e_{i1}X_1 + e_{i2}X_2 + \ldots + e_{ip}X_p
\]  

(2.24)

The eigenvectors \( e_{1i}, e_{2i}, \ldots, e_{pi} \) should maximizes:

\[
\text{var}(Y_i) = \sum_{k=1}^{p} \sum_{l=1}^{p} e_{ik} e_{il} \sigma_{kl} = e_i' \Sigma e_i
\]  

(2.25)
and meet two constrains: the sum of the coefficients of the eigenvectors \( e_i \) is set to one (Equation 2.26); and the recombined vectors are uncorrelated (Equation 2.27).

\[
e'_i e_j = \sum_{j=1}^{p} e^2_{ij} = 1 \tag{2.26}
\]

\[
\text{cov}(Y_{i-1}, Y_i) = \sum_{k=1}^{p} \sum_{l=1}^{p} e_{i-1,k} e_{il} \sigma_{kl} = e'_i \Sigma e_i = 0 \tag{2.27}
\]

The proportion of variation explained by the first \( k \) principal component is

\[
\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_p} \tag{2.28}
\]

we try to make the above value as close to one and at the same time make \( k \) as small as possible, which are obviously two conflicting goals and need to be balanced in practice.

This method is especially useful for the electronic nose (Enose) data analysis since the majority of Enose sensors are highly correlated with each other and could be reduced to a few principal components by using PCA. However, caution should be made that PCA may not give satisfactory results when the sensor responses are not linear because PCA is a technique for linear problems (Pearce et al., 2003).

**Bayesian discriminant analysis**

Both the linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) are based on Bayesian discriminant theory (Anderson, 2003).

In Bayesian discriminant analysis, the *prior* probability is defined as the probability of a random sample belonging to population \( \pi_i \):

\[
p_i = \text{Pr}(\pi_i) \tag{2.29}
\]
We assume that the observations $X$ from population $\pi_i$ is sampled from a multivariate normal distribution with mean vector $\mu_i$ and variance-covariance matrix $\Sigma_i$.

The *posterior* probability that a random sample belongs to population $\pi_i$ is

$$p(\pi_i) = \frac{f(x/\pi_i) p_i}{\sum_{j=1}^{g} f(x/\pi_j) p_j}$$

(2.30)

The decision rule is to classify the sample unit into the population $\pi_i$ that maximizes the posterior probability $p(\pi_i)$.

Specifically, for a linear discriminant analysis which has homogeneous variance-covariance matrices:

$$\Sigma_1 = \Sigma_2 = \cdots = \Sigma_g = \Sigma$$

The linear discriminant function can be calculated:

$$d^L_i(x) = -\frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i + \mu_i^T \Sigma^{-1} x = d_{i0} + \sum_{j=1}^{p} d_j x_j$$

(2.31)

Where mean vector $\mu_i$ and variance-covariance matrix $\Sigma_i$ can be obtained from training data set. The decision rule is to classify the sample into the class that has the largest linear function score.

### 2.7. Evolution algorithms and optimization

#### 2.7.1 Optimization

Optimization is to minimize or maximize the objective function by adjusting the input variables of a device, mathematical process, or experiment (Cuthbert, 1987). More strictly in a mathematical term, global optimization can be defined as the process of
finding the global minimum (or maximum) within a search space $S$ (Coello Coello et al., 2002):

Given a function $f: \Omega \subseteq S = \mathbb{R}^n \rightarrow \mathbb{R}$, $\Omega \neq \emptyset$ for $\vec{x} \in \Omega$ the value

$$f^* = f(\vec{x}^*) > -\infty$$

is called a global minimum if and only if

$$\forall \vec{x} \in \Omega: f(\vec{x}^*) \leq f(\vec{x})$$

(2.32)

where $\vec{x}^*$ is the global minimum solution, $f$ is the objective function, and the set $\Omega$ is the feasible region ($\Omega \subseteq S$).

Some representative optimization methods include calculus-based analytical optimization method (Thompson, 1992), Nelder-mead downhill simplex method (Nelder and Mead, 1965), enumerative method, and random search/walk (Goldberg, 1989). A more complete list of optimization methods are listed in Figure 2.19.

Figure 2.19. Global optimization approaches (Adapted from Coello Coello, 2002)
Although calculus-based method sometimes can elegantly find the minimum value if the function is differentiable, this method has two severe shortcomings (Goldberg, 1989): 1). this method seeks the best in a neighborhood of the current point and is incapable of solving the problem with multiple peaks as shown in Figure 2.20; 2). many practical problems are not differentiable (Figure 2.20. (b)). Enumerative approaches are straightforward by searching the objective function values at every point in space, one at a time. However, the weakness of this method is obvious due to its inefficiency: many practical problems are too large to search one at a time. Random search algorithms have achieved increasing popularity due to its simplicity of searching and saving the best. However, this method is also not efficient and it can not perform better than enumerative method (Goldberg, 1989).

Figure 2.20. Multimodal (a) and discontinuous (b) optimization problems (Adapted from Goldberg, 1989)

Because of these limitations, these traditional optimization algorithms are only suitable for a very limited problem domain (Figure 2.21). Although genetic algorithms
use random choice as a tool to guide their search, they surpass their counterparts in terms of robustness and they search and optimize a problem in the following four ways that are different from traditional methods (Goldberg, 1989):

1) GAs work with a coding of the parameter set, not the parameters themselves;

2) GAs search from a population of points, not a single point;

3) GAs use objective function information, not derivatives or other auxiliary knowledge;

4) GAs use probabilistic transition rules, not deterministic rules;

Figure 2.21. Comparison of different optimization schemes for different problems types (Adapted from Goldberg, 1989)

2.7.2 Evolutionary algorithms

Evolutionary algorithm (EA) is inspired by natural evolution and genetics for optimization, search, and machine learning. It is a broader term which includes not only genetic algorithms, but the other two algorithms: evolutionary programming and evolutionary strategy. These three algorithms were developed roughly at the same time
in the course of 1960s to 1970s: evolutionary programming (EP) by Fogel in San Diego (Fogel, 1962), genetic algorithms by John Holland in University of Michigan (Holland, 1975), and evolutionary strategies by students at the Technical University of Berlin (Rechenberg, 1965). Together with other natural evolution-inspired algorithms such as differential evolution (DE) (Storn and Price, 1995), they are all termed evolutionary algorithms (EA) or evolutionary computation (EC). Although three evolutionary algorithms: GAs, ES and DE are introduced in Chapter six and seven, in this section, the general background of two main evolutionary algorithms (genetic algorithms and evolutionary strategies), a brief introduction of other optimization algorithms, and multiobjective optimization using evolutionary algorithms are discussed.

Other natural optimization methods such as simulated annealing (Kirkpatrick et al., 1983), particle swarm optimization (Parsopoulos and Verahatis, 2002), and colony optimization (Dorigo and Maria, 1997) are beyond the scope of this discussion.

**Genetic algorithms**

Genetic algorithm is an optimization and search technique based on the mechanics of genetics and natural selection (Goldberg, 1989). A GA tries to find the global minima or maximize the fitness by evolving a population composed of many individuals under certain selection rules (Haupt and Haupt, 2004). Genetic algorithms was first proposed and developed by John Holland at University of Michigan (Holland, 1975), and further popularized by one of his students David Goldberg who is now a professor at University of Illinois at Urbana-Champaign with a classic book: *Genetic algorithms for optimization, search, and machine learning* (Goldberg, 1989). John
Holland’s another student De Jong illustrated the usefulness of the GA as an optimization method by testing on a set of deliberately selected functions and made the first intensive study to find the optimized parameters for GA in his dissertation (De Jong, 1975). These functions had the following characteristics (Goldberg, 1989): continuous and discontinuous, convex and nonconvex, unimodal and multimodal, quadratic and nonquadratic, low and high dimensionality, and deterministic and stochastic, and interestingly, they were adopted as testing bed in many other researches in following years (Michalewicz, 1994). In recent years, the multiobjective genetic algorithms have become the central theme of their development (Coello Coello et al., 2002; Deb, 2001).

A simple genetic algorithm is composed of three important operators: 1. selection, 2. mating, and 3 mutation. Selection preferentially samples higher fitness solutions and biases the population to converge to the best solutions. Mating is the process of creating one or more offsprings from the selected parents. The simplest single point crossover follows two steps: first, a crossover point k is randomly selected between \([1 \leq l]\) \((l\) is the length of the chromosome) along the parents chromosomes; second, two new offspring are created by swapping all binary code of two parents between positions \(k+1\) and \(l\) inclusively. There are also other complicated crossover algorithms including two-point crossover and uniform crossover (Haupt and Haupt, 2004). The mating and selection operators allow GAs to globally search promising regions of a problem space. Lastly, mutation flips the binary code from 1 to 0 or vice versa. It prevents the development of a uniform population that is incapable of further evolution. In combination with selection, mutation allows GAs to locally search the problem space near a given solution. More detailed introduction of these three operators is presented in Chapter 6 and 7.
Genetic algorithms were inspired by genetics and natural selection and a comparison between biological genetics and artificial GAs was given in Table 2.8 (Goldberg, 1989). The strings of GAs are corresponding to chromosomes in biological systems. In natural system, the chromosomes are composed of genes with the value of genes called alleles and the position of a gene called locus, whereas in GAs strings are composed of features or variables with different values and positions along the string. In natural systems the total genetic package is called the genotype, while in GAs the total package of strings is called a structure; the phenotype in natural system is comparable to a particular parameter set or solution alternative in GAs.

<table>
<thead>
<tr>
<th>Natural</th>
<th>Genetic algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome</td>
<td>String</td>
</tr>
<tr>
<td>Gene</td>
<td>Feature, character, or detector</td>
</tr>
<tr>
<td>Allele</td>
<td>Feature value</td>
</tr>
<tr>
<td>Locus</td>
<td>String position</td>
</tr>
<tr>
<td>Genotype</td>
<td>Structure</td>
</tr>
<tr>
<td>Phenotype</td>
<td>Parameter set, alternative solution</td>
</tr>
<tr>
<td>Epistasis</td>
<td>Nonlinearity</td>
</tr>
</tbody>
</table>

Compared to other algorithms, the main strengths of a GA include (Haupt and Haupt, 2004):

1) Optimizes discontinuous, noisy, and non-convex problem with extremely complex cost surfaces;

2) Deals with a large number of variables;

3) Parallelism;

4) Works with numerically generated data, experimental data, or analytical functions.
Evolutionary strategy (ES)

Evolution strategy (ES) is another powerful optimization technique that is based on natural evolution and adaptation. Interestingly, it was not devised to find the minima of functions, but developed as a set of rules for the automatic design in a wind tunnel (Beyer and Schwefel, 2002). Although both ES and GAs are natural inspired search and optimization methods, and they both use search operators include selection, recombination and mutation, there are some differences between these two schemes. The most fundamental difference between them is that evolution strategies use the mutation as its basic variation operator (Beyer and Schwefel, 2002). Mutation in ES is achieved by generating Gaussian distributed random numbers and adding them in each vector value. In addition, unlike GAs, Evolution strategies primarily use real-vector coding instead of binary coding. ES has the self-adaptation mechanism that can be realized by adjusting the mutation strength (the standard deviation of Gaussian distribution) during the optimization. Due to these interesting differences, it is noteworthy to introduce three important operators of this algorithm here and it is reiterated from Beyer et al. (2002).

Selection determines that only the \( u \) best individuals with high fitness values (objective function values) can get a chance of reproduction. It can be expressed as follows:

\[
\beta^{(g+1)}_p = \{\alpha_{1,\gamma}, \ldots, \alpha_{u,\gamma}\}
\]

(2.34)

where \( \alpha_{i,\gamma} \) means taking the \( i \)th best individual out of \( \gamma \) individuals.

There are two types of selection in ES, which are termed plus selection (\( \mu + \lambda \)) and comma selection (\( \mu, \lambda \)). In the case of (\( \mu, \lambda \)) selection, parents in the previous
generation are discarded even they are better than all offspring, and only the \( \lambda \) newly generated offspring individuals are considered as the selection pool and the \( \mu \) best individuals are selected from \( \lambda \) offspring pool. Obviously, this selection does not provide search-relevant information and it may take long time to find the optimal solution in the search space. The plus selection \((\mu + \lambda)\) includes the old parents in the selection pool which is of size \( \gamma = \mu + \lambda \). This selection technique introduces elitism into the ES and preserves the best individual.

Mutation is the main source of genetic variation in the ES. There are three criteria for mutation operator designing:

1). Reachability: given a parental state \((y_p, s_p)\), any other finite state \((y, s)\) can be reached within a finite number of mutation steps.

2). Unbiasedness: this ensures that there is no preference of any of the selected individuals in ES and variation explores the whole search space.

3). Scalability: the mutation strength \((\sigma \text{ of Gaussian distribution})\) or the average length of a mutation step should be adjustable in order to adapt to the properties of the fitness landscape.

There are four typical elementary mutation operators: inversion, insertion, 2-exchange (reciprocal exchange), and shifting (displacement) as illustrated in Figure 2.22.
Unlike standard crossover in GA, where two parents produce two offspring, the application of ES recombination operator to a parent family of size $ρ$ produces only one offspring. Dominant recombination and intermediate recombination are two standard classes of recombination used in ES. In dominant recombination, the $k^{th}$ component of recombinant is determined exclusively by the $k^{th}$ component of the randomly selected parent individual $a_{mk}$:

$$(r)_k := (a_{m_k})_k, \quad \text{with } m_k := \text{random}\{1, \ldots, ρ\}. \quad (2.35)$$

In intermediate recombination, the kth component of recombinant is determined by the centroid of the $ρ$ parents vectors $a_m$.

$$(r)_k := \frac{1}{ρ} \sum_{m=1}^{ρ} (a_m)_k \quad (2.36)$$
Figure 2.23. Evolution strategies (ES) principle flow chart

- **Initialize parent population:**
  \[ \beta^{(0)}_p := \{(y^{(0)}_m, s^{(0)}_m, F(y^{(0)}_m)), m = 1, ..., \mu \} \]

- **Marriage:** \( (\beta^{(g)}_p, \rho) \)

- **Recombination:** \( s_l, y_l \)

- **Mutation:** \( s_l, y_l \)

- **Evaluate objective function:** \( \hat{F}_i = F(y_i) \)

- **Offspring population:**
  \[ \beta^{(e)}_o := \{((\hat{y}_i, s_i, \hat{F}_i), l = 1, ..., \lambda \} \]

- **Selection:** \( (\mu, \lambda) : \beta^{(g+1)}_p = (\beta^{(e)}_o, \mu) ; \)
  \( (\mu + \lambda) : \beta^{(g+1)}_p = (\beta^{(e)}_o, \beta^{(e)}_p, \mu) ; \)

- **Termination condition meet?**
  - Yes
  - End
2.7.3 Evolutionary multiobjective optimization (EMO)

Since the evolutionary multiobjective optimization is becoming more prevalent (Coello Coello et al., 2002), it is necessary to introduce this scheme.

Multiobjective optimization is defined as a problem to find the vector \( \vec{x}^* = [x_1^*, x_2^*, ..., x_n^*]^T \) which satisfies the \( m \) inequality constraints (Coello Coello et al., 2002):

\[
g_i(\vec{x}) \geq 0 \quad i = 1, 2, ..., m
\]  \hspace{1cm} (2.37)

and \( p \) equality constraints:

\[
h_i(\vec{x}) = 0 \quad i = 1, 2, ..., p
\]  \hspace{1cm} (2.38)

and optimizes the vector function

\[
\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})]^T
\]  \hspace{1cm} (2.39)

In other words, multiobjective optimization intends to determine the particular set \( x_1^*, x_2^*, ..., x_n^* \) which satisfies Equations (2.37) and (2.38) and yields the optimum values of all the objective functions.

For most multiobjective optimization problems, the objectives are usually in conflict with each other and goal is to find a solution which gives the values of all the objective functions acceptable to the decision maker (Osyczka, 1985). A two-objective optimization problem is shown in Figure 2.24, and it is found that the decrease of one objective function is with the cost of the increase of another objective function. The tradeoff line is called Pareto front which is obtained by computing the points \( \Omega \) and their corresponding \( f(\Omega) \) with two objective functions.
Although many efforts have been made to develop effective multiobjective optimization methods, evolutionary algorithms were found to be especially suitable to solve multiobjective optimization problems in the following two reasons (Coello Coello et al., 2002): 1). they can deal simultaneously a set of population within a single run for problems with huge decision spaces. 2). these methods have been demonstrated to be effective in solving high-order Pareto optimization problems.

Essentially, the EMO algorithms are identical to single objective genetic algorithms because both these two approaches search complex problem spaces using a process that is analogous to Darwinian natural selection. The main difference between EMO algorithms and single objective GAs is that how fitness is assigned. EMO algorithms evaluate sampling designs in terms of a vector of objectives instead of any single objective because it may perform poorly with respect to the other objectives.
The first evolutionary multiobjective optimization (EMO) algorithms was proposed in 1980s (Schaffer, 1985) and it was termed the vector evaluated genetic algorithms (VEGA). It was designed to search decision spaces for the optimal tradeoffs among a vector of objectives. Since then, many varieties of EMO algorithms have been developed and successfully applied in various fields (Coello Coello et al., 2002). Table 2.9 lists some representative EMO algorithms developed during the past two decades.

Table 2.9. Representative EMO algorithms (Adapted from Coello Coello et al., 2002)

<table>
<thead>
<tr>
<th>EMO name</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>VEGA</td>
<td>(Schaffer, 1985)</td>
</tr>
<tr>
<td>HLEA</td>
<td>(Hajela and Lin, 1992)</td>
</tr>
<tr>
<td>MOGA</td>
<td>(Fonseca and Fleming, 1993)</td>
</tr>
<tr>
<td>MOMGA</td>
<td>(Van Veldhuizen and Lamont, 2000)</td>
</tr>
<tr>
<td>MOMGA-II</td>
<td>(Zydallis et al., 2001)</td>
</tr>
<tr>
<td>NPGA/NPGA II</td>
<td>(Erickson et al., 2001; Horn et al., 1994)</td>
</tr>
<tr>
<td>NSGA/NSGA-II</td>
<td>(Deb, 2000; Srinivas and Deb, 1994)</td>
</tr>
<tr>
<td>PAES and PAES-II</td>
<td>(Knowles and Corne, 2000; Knowles and Corne, 2001)</td>
</tr>
<tr>
<td>SPEA and SPEA2</td>
<td>(Zitzler et al., 2001; Zitzler and Thiele, 1999)</td>
</tr>
</tbody>
</table>

2.8 Multisensor data fusion

2.8.1 Introduction

Humans and animals have developed the capability to utilize information from different senses to enhance the accuracy of detection and assessment of enemies, and prey in surrounding environments, which consequently improved their chances of survival during the long process of evolution (Hall, 1997). For instance, humans evaluate the edibility of a food not only by sense of vision, but by sense of taste, smell, and touch. Another good biological example of the synergistic integration of multisensor information can be found in rattlesnakes: researchers have identified neurons in
rattlesnakes’s optic tectums (a midbrain structure found in vertebrates) that are responsive to both visual and infrared information (Newman and Hartline, 1982) as shown in Figure 2.25. Both the left eye (visual) and pit organ (infrared) of a rattlesnake can receive information from the same region in the environment. Certain neurons that respond to either visual or infrared information could be used to detect prey in dark conditions; other neurons that only respond to information from both sources of information could be used to differentiate warm-blooded and cold-blooded preys.

Figure 2.25 Rattlesnake receives visual and infrared information from environment (Adapted from Newman and Hartline, 1982)

This feature of multisensory data fusion from biological systems was borrowed by modern engineers and the concept of multisensor data fusion was first developed by the Department of Defense (DoD) (Klein., 2004). The resulting definition is:

\[
\text{A multilevel, multifaceted process dealing with the automatic detection, association, correlation, estimation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates,}
\]
and complete and timely assessments of situations and threats and their significance.

Initially, data fusion was primarily applied and studied in military areas such as battlefield intelligence, surveillance and target acquisition. Recently, it has also been widely applied to non-military uses including monitoring of complex machinery, medical diagnosis and remote sensing (Hall and Llinas, 2001). Due to the significant expenditures by the Department of Defense (DoD) and recent rapid development of new sensors, advanced processing techniques, and improved processing hardware, data fusion has gained more attention and developed as a relatively independent engineering discipline (Hall, 1997).

The potential advantages of data fusion include robust operational performance, extended spatial coverage, extended temporal coverage, increased confidence of target location and identity, reduced ambiguity, improved system reliability, increased dimensionality, etc. (Hall, 1997). Multiple sensors could provide information with different fidelity and thus reduce the overall uncertainty and increase reliability in the case of sensor error or failure. When each sensor could only provide a subset of the whole feature space, complementary information is provided and an improved system performance from sensor fusion is expected (Waltz and Llinas, 1990).

Despite all of the advantages of data fusion mentioned above, there are also risks in applying sensor fusion in practice: data fusion may give worse results than could be obtained by using an appropriate single sensor. The error could arise in the following three situations (Luo and Kay, 1995):
1) Error in the integration and fusion process. For instance, in stereo vision and multi-target tracking research, registration is a main problem. It is to determine that the information from each sensor is referring to the same features in the environment (note – this sentence isn’t clear).

2) Error in sensory information. It is caused by white noise, Gaussian noise and independent noise during the sampling process and can be modeled by certain probabilistic distributions.

3) Error in system operation. It is usually caused by a coupling effect between different components of a system during operation.

Future research on data fusion was projected in the following five areas (Luo and Kay, 1995):

1) Develop fusion techniques that will allow multisensory systems to operate in unknown and dynamic environments.

2) Adapt fusion techniques to highly parallel computer architectures to take full advantage of the parallelism inherent in the techniques.

3) Develop a standard for sensor modeling and interfaces.

4) Continued research of artificial intelligence (AI) and artificial neural networks (ANN) will provide both theoretical and practical insights for data fusion.

5) Develop hardware that contains integrated solid-state chips that can perform multiple sensor data fusion.
2.8.2 Architectures of data fusion

A standard data fusion process model was created by the Joint Directors of Laboratories (JDL) Data Fusion Working Group, which was founded in 1986 (Hall, 1997; Kokar and Kim, 1994). The top level of JDL data fusion process model is shown in Figure 2.26.

- Sources of information: includes local sensors, distributed sensors linked to a fusion system or reference information, geographical information.
- Human Computer Interface (HCI): allows human input commands and communicates results via alerts and displays.
- Level 0 processing: preprocesses the source information by normalizing, formatting, ordering, and compressing data in order to address process estimation and processor computational and scheduling requirements.
- Level 1 processing (Object Refinement): achieves refined representation of individual objects by fusing locational, parametric, and identity information from individual sensors.
- Level 2 processing (Situation Refinement): develops a refined assessment of relationships among objects and events in the context of the environment.
- Level 3 processing (Threat Refinement): predicts the enemy threats and opportunities for operations using inferences drawn from level 2 associations.
- Level 4 processing (Process Refinement): improves the performance of the data fusion by continuously refining estimates and assessments through planning, control, and modifying the fusion process itself.
For identity fusion of Level 1, there are three types of architectures at different levels (Alexander et al., 1998; Hall and Llinas, 2001):

1. High level fusion: as shown in Figure 2.27, it is performed by using the identity declaration provided by each sensor; the fusion of the identity declaration is then made by using Bayesian references or Dempster-Shafer method.

2. Intermediate level fusion: as shown in Figure 2.28, it is performed by using the extracted features from each sensor and the identity declaration process includes techniques such as knowledge-based approaches (expert system, fuzzy logic), or training-
based approaches (discriminant analysis, neural networks, Bayesian technique, k nearest neighbors, center mobile algorithms).

(3). Low level fusion: as shown in Figure 2.29, it combines the signals provided by different sensors before any processing. It implies that the sensors must be similar, and the signals consequently must be commensurate. Methods for this level of fusion include: neural networks, template methods, and cluster algorithms.

The selection of these architectures depends on characteristics of the sensors, computational resources, and other issues.
2.8.3 Mathematical techniques in multisensor data fusion

Compared to other well developed engineering disciplines, multisensor data fusion is relatively new and multidisciplinary. It includes a diverse set of traditional disciplines including digital signal processing, statistical estimation, control theory, artificial intelligence, and classic numerical methods (Crowley and Demazeau, 1993; Hall and McMullen, 2004). In Figure 2.30, mathematical algorithms that can be used in data fusion are listed.

Figure 2.30. Mathematical algorithms for data fusion (Adapted from Klein, 2004)

Artificial neural networks: they can be used in both data level and feature level fusion. Multiple sources of data or features are fed into the artificial neural networks’ input layer and the ANNs are trained to map input data into output categories. Since ANNs do not have explicit mathematical equations to depict the relationship between
input and output, they are also called blackbox fusion methods. More detailed discussions of ANNs are presented in Section 4 of this Chapter.

Cluster algorithms can be used for data level fusion. It groups data into natural sets or clusters based on their internal similarities without any training. A similarity metric is defined to measure the closeness between any two feature vectors. There are five steps for a cluster algorithms operation: (1) selection of data; (2) definition of variables and features; (3) computation of similarities of data; (4) use of cluster analysis to create groups based on similarities; (5) validation.

Bayesian inference is used for decision level fusion. It is a probability-based reasoning method based on Bayes’ rule. It uses \textit{a priori} knowledge about events or objects in an observation space to calculate the conditional \textit{a posteriori} probability of a hypothesis being true given supporting evidence. In Chapter 8, Bayesian inference was used for decision level data fusion model development.

The voting method is used in decision level fusion. It simply treats each sensor’s declaration as a vote in which majority, plurality, or decision-tree rules are used. It is simpler than Bayesian inference and easy to implement. Additional discrimination can be introduced by giving different weights to each sensor’s declaration.

2.8.4 Applications

Historically, data fusion methods were developed primarily for military applications. Examples of DoD related applications include ocean surveillance, air to air defense, battlefield intelligence, object identities verification using IFF (Identify Friend or Foe) systems, surveillance and target acquisition, strategic warning and defense, and
land mine detection (Hall, 1997; Jagerbro et al., 1998). Each of these military applications involves a particular focus, sensor suite, desired set of inferences, and a particular set of challenges.

In recent years, data fusion has been increasingly applied to civilian applications including: remote sensing, automated control of industrial manufacturing systems, medical diagnosis, and food quality and safety inspection (Brooks and Iyengar, 1998; Hague et al., 2000).

In the remote sensing area, satellite data have been obtained to identify and locate entities and objects. Examples include systems to monitor agricultural resources such as the crop yield prediction and disease monitoring, to locate natural resources, and to monitor weather and natural disasters. Multispectral data are processed on a pixel-by-pixel basis and input to a neural network to automatically classify the contents of the image (Waltz, 1995).

Multisensor data fusion has also been applied to monitor complex mechanical equipment and industrial manufacturing equipment. The accuracy of equipment failure detection can be improved by using multiple sensors including accelerometers, temperature gauges, acoustic sensors, and infrared measurements. Data fusion in this application could reduce costs for maintenance and improve safety (Hansen et al., 1995).

In the medical area, more and more sophisticated sensors have been used in medical diagnosis. Sensors such as nuclear magnetic resonance (NMR) devices, acoustic imaging devices and fiber-optic probes, provide various ways to examine patients. If these devices and their data could be integrated, the accuracy and reliability of diagnosis would be greatly improved (Brooks and Iyengar, 1998; Di Natale et al., 2000).
In food quality and safety inspection, which is of interest to this dissertation, research was conducted to use multiple sensors to improve detection accuracy (Di Natale et al., 2001; Perrot et al., 1996). A general methodology was proposed for sensor fusion design for fruit quality assessment applications (Steinmetz et al., 1999b). This methodology included three different fusion methods, and eight steps to implement the fusion model. His method was also tested on two fruits, melon and peach, and desirable results were obtained (Steinmetz et al., 1996; Steinmetz et al., 1999a). Research efforts were made to combine all features of artificial olfactory and artificial taste to classify three different beverages, beer, brandy, and vodka (Li et al., 2000). The gas sensor has five metal-oxide-semiconductor gas sensors and the taste sensor has three electrodes with lipid membranes. A fuzzy c-means algorithm (FCM) was used for data preprocessing and a fuzzy neural network was used for data fusion. It was found that using redundant and complementary information from olfactory space and taste space, the system classification performance was improved by 10%. Roussel (2003) used three sensors including the electronic nose, Fourier transform infrared (FT-IR), and ultraviolet spectrometers to discriminate grape varieties based on its musts (grape juices before fermentation). Multisensor data fusion was conducted at the data and decision levels (Roussel et al., 2003a; Roussel et al., 2003b). It was found that data fusion did not give better results than those obtained by the best individual sensor (FT-IR) with 9.6% error rate at the data level. The error rate was reduced to 4.7% when decision level fusion models were applied. This research indicated the risks of worsening the classification results by using inappropriate data fusion architectures and algorithms: the addition of raw noisy and correlated data worsened the classification results instead of improving
Some researchers also tried to integrate three sensor modalities: gas sensors (e-nose), electrochemical liquid sensors (e-tongue) and an optical system (e-eye) to classify six red Spanish wines based on their different geographic origins and aging stages (Rodriguez-Mendez et al., 2004). The authors proposed to use correlation coefficient to determine whether these sensors were complementary and selected those less correlated for data fusion model development. Ozer (et al., 1995) used different sources of information and parameters (color, firmness, size, shape and weight) to evaluate melon’s quality (Ozer et al., 1995). Those quality indicators were used as input to a Recurrent Auto-Associative Memory that classifies the fruit into four maturity stages. This fusion achieved 85.1% correct classification and 10.1% neighbor class classification. Boilot et al. (Boilot et al., 2003) used four different electronic noses and evaluated them by testing on six standard fruit samples, pure liquids, and mixtures. Feature extraction techniques were investigated using PCA and genetic algorithms (GA). Radial basis function and a probabilistic neural network (PNN) were used to develop a low level data fusion model. An 86.7% correct classification was achieved by the fusion model, which outperformed the results obtained with individual ENs.
Chapter 3

Preliminary Experiments

Experiments that yield light are more worthwhile than experiments that yield fruit.

Francis Bacon (1561-1626)

In this chapter, preliminary experiments were conducted on both the Enose and zNose™ to determine their sampling parameters, and test their responses to apple volatile compounds. Other issues such as data preprocessing techniques are addressed in the following chapters.

3.1 The Enose sampling parameters determination

3.1.1 Sampling method set up

The Cyranose 320 electronic nose (Smith Detection, Herts, UK) used in this research consists of a 32 thin-film carbon-black conducting polymer sensor array inside the instrument (Figure 3.1 (a)). The commercially available software PC Nose that accompanied the Cyranose 320 can collect and classify data using embedded statistical methods. However, in this research, PC Nose was only used to collect raw data and the consequent data processing was implemented by calibration models developed by the author.
Before the experiments were conducted, the Enose sampling method, which controls sampling processes such as sample draw time, pump speed and substrate temperature, needs to be set up. The typical sample draw time is from 10 seconds to 60 seconds for the Cyranose 320. The criterion for choosing appropriate sample draw time is to determine the minimum time when sensors reach constant value at the end of the sample draw time. The Enose’s 32 sensor real-time responses to three red Delicious apple volatile sample,s using 10 s and 30 s draw times, are shown in Figure 3.2. It is evident that the sensor responses did not reach constant values when the 10 s sample draw time was used. By extending the sample draw time to 30 s, relatively constant responses were achieved.
Figure 3.2. Enose 10 s (a) and 30 s (b) sample draw time response comparison

The pump speed is set to ‘medium’ for both ‘baseline purge’ and ‘sample draw’, so that there were no pressure or other flow effects on the calculation of sensor responses. It is set to ‘high’ for ‘sample purge’ and ‘air purge’ so that the analyte can be quickly removed from the sensors.

The set point for substrate temperature is typically at least 7 °C higher than the highest expected ambient temperature during normal operation. In summer, the highest expected ambient temperature ranges from 30 °C to 35 °C, so a substrate temperature of 42 °C was chosen (Table 3.1). Sample temperature affects the concentration and composition of the headspace as well as the vaporization rate when the headspace gas is measured. However, samples were measured in ambient room temperature without heat treatment in order to simulate real world situations. Since the polymer-composite sensors in the Cyranose 320 are sensitive to water vapor, the water content differences between the sample and the purge streams should be as small as possible in order to avoid humidity interference. However, relative humidity in the lab room air usually reaches
50-60%, which counteracts high humidity (90%) in the glass jar to some extent. Thus, the relative humidity was not specially controlled in this study.

Table 3.1. The Enose sampling method set up

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (sec)</th>
<th>Pump speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline purge</td>
<td>10</td>
<td>Medium</td>
</tr>
<tr>
<td>Sample draw1</td>
<td>30</td>
<td>Medium</td>
</tr>
<tr>
<td>Sample draw2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Snout removal</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1st sample gas purge</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1st air intake purge</td>
<td>10</td>
<td>High</td>
</tr>
<tr>
<td>2nd sample gas purge</td>
<td>60</td>
<td>High</td>
</tr>
<tr>
<td>2nd air intake purge</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Substrate heater</td>
<td>42°C</td>
<td></td>
</tr>
<tr>
<td>Active sensors</td>
<td>All</td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>Canonical</td>
<td></td>
</tr>
</tbody>
</table>

3.1.2 Enose preliminary experiments

A group of preliminary experiments were conducted to compare the Enose’s responses to different air samples, to determine the purge gas and the apple concentration time.

Figure 3.3 compares Enose sensor responses with four different sample concentration times: 4 hr, 6 hr, 9 hr and 25 hr. Sensors 5, 6, 23, 31 were deselected because of their significantly larger responses than other sensors, which makes observation of other sensors’ responses difficult. It is observed that the Enose responses generally increase as the concentration time increases. The concentration time of 25 hr
gave the highest magnitude of response for the 28 sensor array. The concentration time of 6 hrs was chosen after considering the balance between sample preparation time and sensor response.

Figure 3.3. Enose sensor responses with four different sample concentration times

In order to verify that the Enose was actually responding to apple headspace volatiles, the Enose was used to measure different air samples: air in an empty container, water vapor, and ambient room air. Their responses are shown in Figure 3.4.
From Figure 3.4, it is observed that the Enose sensor responses to the empty glass jar and ambient air are similar, and they are close to zero compared to the Enose sensor responses to water vapor. The water vapor response magnitudes are comparable to apple headspace gas. This result proved that the Enose responds differently to ambient air and apple headspace and the previous Enose response to apple headspace gas is reliable.

In order to investigate the effect of purge gas on the Enose sampling, tests were conducted on four different purge gases: water vapor, nitrogen (N$_2$), air in empty container, and ambient room air. The Enose response to apple headspace gas using the water vapor purge is significantly lower than the response using the other three purge approaches because water content in the purge gas counteracted the Enose sensor response.
responses (Figure 3.5). Although nitrogen is recommended as a purge gas because most of gas sensors are not sensitive to it, experiment results indicated that three purge gases (N₂, empty container gas and ambient air) generated similar Enose sensor responses. In order to simplify the sampling process, the ambient room air was used as the purge gas.

![Enose response to 3 Red apples with different purge gas](image)

**Figure 3.5.** The Enose responses to red Delicious apples with four different purge gases

### 3.2 The zNose™ sampling parameters determination

The surface acoustic wave sensor, zNose™ (Electronic Sensor Technology, Newbury Park, CA), which is also called faster GC analyzer, is another instrument used to detect apple headspace volatiles in this research (Figure 3.1 (b)).

The recommended sensor temperature is between 20-60 °C according to the users manual. The maximum allowable temperature is 60 °C because sensor overload is more likely to happen if a lower sensor temperature is chosen. A DB-5 column is used in the
zNose™ and its temperature should not exceed the maximum value set by the manufacturer and thus was set to 40 °C for this project. The suggested valve temperature for volatile organics is between 50 and 70 °C, and up to 170 °C for semi-volatiles. It was set to 70 °C since volatile compounds are of interest in this research. The suggested inlet temperature is between 50 and 100 °C for volatiles, and it was set at 100 °C in this study. Trap temperature is the temperature the trap achieves during injection. The Tenax trap was set to 250 °C as suggested by the zNose™ users’ manual. All of these temperature settings are shown in Table 3.2.

Table 3.2. Temperature settings for DB-5 column in the zNose™

<table>
<thead>
<tr>
<th>Components</th>
<th>Sensor</th>
<th>Column</th>
<th>Valve</th>
<th>Inlet</th>
<th>Trap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>60 °C</td>
<td>40 °C</td>
<td>70 °C</td>
<td>100 °C</td>
<td>250 °C</td>
</tr>
</tbody>
</table>

Similarly, the zNose™ method also controls the sampling process, which consists of six steps (Table 3.3). The pump event starts at 0 s and lasts for 0.5 s; the inject instructs the zNose™ instrument to move the valve to the inject position, which lasts 5 s; the trap gives an order to the zNose™ to heat the trap loop for 0.5 s and at the same time the ramp event directs the zNose™ to increase the column temperature gradually to reach 40 °C; the data event instructs the zNose™ to take sample for 10 s; the bake event cleans the SAW sensor at the end of each run for 15 s.

Table 3.3. Sampling time set up for each process

<table>
<thead>
<tr>
<th>Sample</th>
<th>Pump</th>
<th>Inject</th>
<th>Trap</th>
<th>Ramp</th>
<th>Data</th>
<th>Bake</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>0 s</td>
<td>0.5 s</td>
<td>5.5 s</td>
<td>6.0 s</td>
<td>6.0 s</td>
<td>16.0 s</td>
</tr>
</tbody>
</table>

Unlike the Enose, which does not have a standard calibration method, the zNose™ can be calibrated using a standard solution containing C6-C14 n-alkanes which keeps the
retention time scale uniform. It is a dynamic headspace method as shown in Figure 3.6 (a). Calibration was conducted by inserting a 6-in (convert to SI) long sparging needle through the septum of the 40 ml vial and into the n-alkane mixture. The bent, side hole sampling needle of the zNose™ was inserted through the septum of the 40 ml vial and placed in the headspace air instead of the mixture. Five replicate analyses performed on n-alkane vapors and the waterfall spectrum are shown in Figure 3.6 (b). Each peak in Figure 3.6 (b) represents hydrocarbons C6 to C14.

The helium (He) is not only used as a carrier gas during the zNose™ sampling process, but also as the purge gas between two samplings in order to remove volatile residues on the SAW sensor. The zNose™ also has good response to apple samples after 6 hours sample concentration time. The zNose™ responses to room air and apple
volatiles generated sharply different spectra, which indicated that the SAW sensor within the zNose™ is sensitive to apple volatile compounds (Figure 3.7).

Figure 3.7. zNose™ responses to apple volatile compounds and room air
CHAPTER 4
DETECTION OF APPLE DETERIORATION USING AN ELECTRONIC NOSE AND ZNOSETM

I have often admired the mystical way of Pythagoras, and the secret magic of numbers.

Sir Thomas Browne (1605-1682)

ABSTRACT

Damage in apples can cause fruit spoilage, reduce commodity economic value, and give rise to food quality and safety concerns. This research investigated use of electronic nose (Enose, Cyranose 320) and zNoseTM-based nondestructive protocol for rapid detection of deterioration in apples. Important compounds associated with apple aroma were identified using gas chromatography and mass spectrometry and the differences were observed after 6 days exposure to artificially induced damage in the form of a cut. High dimensional data were compressed by principal component analysis (PCA) and partial least squares (PLS). Linear discriminant analysis (LDA) and canonical variate analysis (CVA) models were developed based on the compressed data. Experiments showed that both the Enose and zNoseTM were able to effectively detect the volatile differences between undamaged apples and damaged apples four or more days after the cut. Differences in number of cuts had some effect on volatile compound emissions. Apples subjected to two cuts and three cuts generated volatile profiles that
were significantly different from uncut apples. Varying the orientation of cut apples did not generate significant differences in the volatile profile. The PLS-LDA model produced the best classification accuracy with 96% correct classification using the zNose™ and 85% using the Enose.

4.1 Introduction

Apples are one of the most important fruit commodities in the United States, with widespread applicability in cider and sauce production. In 2001, total apple production in the United States was valued at $1.5 billion, with an average national consumption of 45.2 pounds of fresh apples and processed apple products every year (University of Illinois Extension, 2005).

However, during the process of sorting, storing, transporting, processing, and packaging, apples are easily damaged and prone to deterioration. It is estimated that fresh produce departments in grocery stores experience 10% loss because the apples spoil, contain undetected defects, or deteriorate in quality before they can be sold (NE-179 Project report, 2001). Furthermore, dropped apples can contaminate non-pasteurized cider and induce possible illness outbreaks (Centers for Disease Control and Prevention, 1998). Increasing instances of food-borne diseases in the US provide incentive to develop novel techniques to test the safety and quality of fruits and vegetables and effectively reduce the risk of food contamination.

Among various indices of fruit quality, fruit aromatic volatiles provide very important information on fruit quality and ripeness. More than 200 volatile compounds related to various cultivars of apples have been identified by GC-MS (Dimick and Hoskin,
However, around 60% of the total volatiles emitted by ripe apples are “butyl acetate” and “hexyl acetate” (Marrazzo, 1999). Compositional changes in volatiles may occur during fruit ripening and vary depending on the presence of diseases and physical damage (Simon et al., 1996), hence evaluating apple quality by measuring total aromatic volatiles emitted by apples is a promising strategy.

Traditionally, there are two ways to measure volatiles. The first is a trained human sensory panel (olfactometry). The main setback of this method is that it cannot detect volatile compounds without odor, and some non-odorous compounds are very important indicators of fruit quality. Furthermore, this method is subjective and varies from time to time and person to person. Another method is to use an analytical instrument such as gas chromatography-mass spectrometer (GC-MS). This method is effective but relies on expensive instruments and requires highly skilled operators. In the past two decades, a new volatile measurement instrument, referred to as the Enose, has gained popularity and is becoming an attractive alternative for non-destructive evaluation of fruit quality and ripeness.

The electronic nose is also referred to as an artificial or mechanical nose, and consists of an array of electronic chemical sensors which are partially specific to certain chemicals and capable of recognizing volatile compounds with a pattern-recognition system (Gardner and Bartlett, 1994). Since the concept of the electronic nose was first developed in 1982 (Persaud and Dodd, 1982), the electronic nose has gained wide application in the food industry for quality control and process monitoring (Schaller et al., 1998). Several research groups have applied the electronic nose to predict fruit (apple, pear, and banana) ripeness (Brezmes et al., 2001; Llobet et al., 1999; Oshita et al., 2000).
Electronic odor sensing techniques have been used for quality sorting of blueberries (Simon et al., 1996), spoilage identification of beef (Balasubramanian et al., 2004), peanut off-flavor detection (Osborn et al., 2001), sausage fermentation monitoring (Eklov et al., 1998), grain quality inspection (Jonsson et al., 1997) and other food products (Benady et al., 1995; Gardner et al., 1992; Roussel et al., 2003a). A preliminary study on the application of Enose to detection of food-borne microorganisms was also explored (Dutta et al., 2005; Powell et al., 2002).

The zNose™, a more recently developed tool, is based on surface acoustic wave (SAW) propagation, which produces a change in oscillation of the fundamental frequency corresponding to the compounds absorbed. The system controller then interprets the detector response to identify and quantify each species based on a predefined profile within its resolution limit. The retention time and area of peaks can be used to identify volatile compounds qualitatively and quantitatively (Electronic Sensor Technology, 2001). Investigations on the use of the zNose™ to classify honey and wine, based on their constituents, have demonstrated its potential in food quality evaluation (Korach, 2004; Lammertyn et al., 2004; Tewari and Irudayaraj, 2005; Veraverbeke et al., 2004).

The objectives of this research were to:

1) Identify differences in dominant volatiles emitted by undamaged and deteriorated apples using gas chromatography and mass spectrometry;
2) Differentiate undamaged from deteriorated apples using multivariate analysis;
3) Investigate Enose sensor reduction, effects of different physical damage, placement of cut, and time effect on differentiating undamaged from deteriorated apples.
4.2 Materials and Methods

4.2.1 Samples

Red ‘Delicious’ apples were selected for this study because they are the most widely grown cultivar in the United States (University of Illinois Extension, 2005). Undamaged apples (without defects) were purchased at a local grocery store and stored in a refrigerator at 4-5°C to inhibit respiration. Prior to testing, all samples were kept in room air for 6 hours to reach the ambient air temperature (23 °C) and then were stored in 2L glass jars for 6 hours for headspace gas equilibrium. After each measurement, apples were maintained at ambient temperature for 48 hours to allow deterioration. Controlled damage was induced at the surface through a 10 mm deep and 70 mm long single slice cut at the apple surface. Glass jars were used as headspace gas concentration chambers to minimize the effect of the environmental gas matrix. The jars were sealed by a plastic cap with a Teflon septum. A 5 mm diameter hole was drilled at the center of the lid, through which the Enose and zNose™ intake needles could be inserted and samples could be taken from the headspace. A silicon stopper was used to seal the hole.

Three sets of apples were tested respectively in 2004 and 2005. In the first experiment, to investigate the effect of apple physical damage on volatile emissions, four groups of apples (control, 1 cut, 2 cuts, and 3 cuts) with 6 apples each were measured by the Enose and zNose™ over a period of 14 days after which the apples were severely rotten. Data from set 2 were obtained by measuring two groups of six apples (control and damaged) for a period of 11 days until the apples were severely rotten. Four groups of six apples with different cut orientations (control, damaged up; damaged side and damaged
down) were tested during the third experiment. Damaged apples were placed in different positions, e.g. the cut was placed upward, sideward and downward. The test lasted for 10 days after which the apples were very rotten, and measurements were taken every other day.

4.2.2 Enose data acquisition

The Cyranose 320 electronic nose (Smith Detection, Herts, UK) used in this research consists of an array of 32 individual thin-film carbon-black polymer composite chemiresistors (Cyrano Science Inc., 2000). These sensor materials are thin films deposited across two electrical leads on an alumina substrate, to form conducting chemiresistors. When a composite film is exposed to a vapor-phase analyte, the polymer matrix acts like a sponge and swells while absorbing the analyte. The increase in volume causes an increase in resistance because the conductive carbon-black pathways through the material are disrupted. The polymer will shrink to its original size and restore conductivity after the analyte desorbs.

The normalized change in resistance is presented by the following equation:

$$\frac{\Delta R}{R} = \frac{R_{odor} - R_{air}}{R_{air}}$$  \hspace{1cm} (4.1)

Where $R_{air}$ is the sensor resistance to purge gas air and $R_{odor}$ is the sensor resistance due to volatile absorption. Sensors 5, 6, 23 and 31 were deselected because they are very sensitive to polar compounds such as water vapor, which could adversely affect the sensor data when present in high amounts. The other 28 sensors were kept active. Measurements were performed under ambient conditions of temperature and
humidity (temperature 23.4±1.4 °C, humidity 19±0.8%). Temperature and humidity were measured by hygro-thermometer Model 444712 (Extech Instruments, Waltham, MA).

The Enose samples volatile compounds emitted by apples by inserting a snout needle into the 5mm hole on the lid of the glass jar. These data can be stored in the Enose and downloaded to the computer by an RS 232 cable. The Enose instrument sampling parameters are shown in Table 4.1 and the experimental set-up is shown in Figure 4.1.

Table 4.1. Cyranose 320 instrumental parameters for apple volatile measurement

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Time (sec)</th>
<th>Pump Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline purge</td>
<td>10</td>
<td>Medium</td>
</tr>
<tr>
<td>Sample draw</td>
<td>30</td>
<td>Medium</td>
</tr>
<tr>
<td>1st air intake purge</td>
<td>10</td>
<td>High</td>
</tr>
<tr>
<td>2nd sample gas purge</td>
<td>60</td>
<td>High</td>
</tr>
<tr>
<td>Digital filtering</td>
<td>On</td>
<td></td>
</tr>
<tr>
<td>Substrate heater</td>
<td>On (42 °C)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.1. An experimental schematic for Enose data collection
4.2.3 **zNose™ data acquisition**

The zNose™ (Electronic Sensor Technology, Newbury Park, CA) used in this work consists of a surface acoustic wave sensor (SAW), pneumatic controls and support electronics. The SAW detector, which is based on piezo-electric technology, is used to detect volatile organic compounds.

As shown in Figure 4.2, there are two steps for the zNose™ sampling process: sample collection and vapor analysis. During the sample collection process, apple headspace volatiles are drawn into the inlet via a pump. The volatile compounds pass through the valve and are adsorbed onto the trap. The second step starts when the valve is rotated to put the trap in line with the column for the inject sequence. The absorbed compounds are vaporized by heating the trap and transported down to the capillary column where the compounds are separated based on their different solubility. When different volatiles sequentially exit the column and stick on the SAW detector, the frequency of the surface acoustic wave is altered, which results in a shift in frequency with respect to elution time based on the molecular weight of the compounds eluted.

The zNose™ was equipped with a 5 cm long sampling needle at the inlet, which is inserted into the concentration chamber for sampling. The operational parameters are shown in Table 4.2. The sampling time was 10 seconds, during which the gas sample was released from the trap inside the system and carried over the column (DB-5) in a helium flow of 3 cm$^3$/min. The different chemical components in the gas sample were separated and sequentially detected by the SAW detector. The system was baked for 5 s after each data sampling period to clean the SAW detector. One blank run was conducted
between each sample measurement to ensure proper purging of the system to attain a stable baseline.

Figure 4.2. The zNose\textsuperscript{TM} 7100 schematic diagram

<table>
<thead>
<tr>
<th>Table 4.2. zNose\textsuperscript{TM} operational parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Category</strong></td>
</tr>
<tr>
<td>Sensor</td>
</tr>
<tr>
<td>Column</td>
</tr>
<tr>
<td>Valve</td>
</tr>
<tr>
<td>Inlet</td>
</tr>
<tr>
<td>Trap</td>
</tr>
<tr>
<td>Maximum column temperature</td>
</tr>
</tbody>
</table>

4.2.4 Data analysis

The Enose and zNose\textsuperscript{TM} multi-dimensional data were compressed using principal component analysis (PCA) and partial least square (PLS) prior to processing by discriminant analysis. PCA is a multivariate technique used for reducing the dimensionality of the data while preserving the structure. PCA uses eigenvectors and eigenvalues to define the reduced subspace which is a representation of the original N-dimension space. The principal components are linear combinations of dependent
variables. The coefficients of the linear combinations are the eigenvectors of the covariance or correlation matrix. A correlation matrix was used in this analysis to enhance the influence of small spectral features. The PCA score plot can provide information on the clustering of data while the PCA loading plot can be used to investigate the contribution from each sensor. PLS is a multivariate regression capable of compressing the data while retaining the information content.

Linear discriminant analysis (LDA) and canonical variate analysis (CVA) were used to separate data into classes based on similarity. LDA is applied when different groups of population have homogeneous variance-covariance matrices. LDA will classify data from samples into the population that has the largest linear score function. Mahalanobis distance was used in LDA. CVA derives canonical variables (CV) which are linear combinations of the original variables. Canonical variables maximize the variance between groups and minimize the variance within groups. Discriminant functions can be derived from these canonical variables and be used for classification. WIN-DAS software (John Wiley & Sons, Ltd, UK) was used for statistical analyses.

Multivariate analysis of variance (MANOVA) was used to investigate whether apple volatiles depend on apple quality. SAS software (SAS Institute, Cary, NC) was applied for this purpose.

4.2.5 Gas chromatography-mass spectrometry

Apple headspace gas was measured by Agilent 6890 GC (Agilent Technologies, Palo Alto, CA) with a mass spectroscopy coated with DBwax. Helium was used as the carrier gas at 1.1 ml/min. The solid phase microextraction (SPME) technique was applied
for sample extraction. The StableFlex fiber was used since it is stable with both polar and non-polar organic solvents. It consists of a 10mm long, 30 μm diameter silica fiber coated with a 30 μm thickness of poly (composed of Divinylbenzene /Carboxen /Polydimethylsiloxane). Before sampling, the fiber was conditioned for 3 mins in the gas chromatograph injection port at 250 °C. Volatile compounds were sampled by inserting the conditioned fiber through a Teflon-coated silicone septum into a glass headspace jar and then extending the fiber from its sheath. Headspace sampling lasted for 20 minutes, and desorption in the GC injection port lasted for 3 minutes. A sampling time of 24 min was needed for GC-MS analysis. Volatiles from the headspace of damaged and undamaged apples were measured by GC-MS after 2, 4, 6 and 9 days cut treatment respectively, and each sample had two replications. Samples were kept in the glass jar for 6 hours to allow volatiles to concentrate before each measurement. The same sampling approach was use for both the Enose and zNose™.

4.3 Results and Discussions

4.3.1 Volatile compounds from GC-MS analysis

Volatile compounds emitted by tested apples were measured by GC-MS from day 2 to day 9. In Figure 4.3, it is observed that more volatile compounds were detected from undamaged apple headspace than from damaged apples at the early stage of treatment (based on the number of peaks detected). As time advanced, the number of compounds detected from undamaged apple headspace varied from 47 to 53, whereas the number from damaged apples increased 73% from 41 to 71.
In Figure 4.4, peak areas from GC-MS chromatograms were compared between undamaged apples and damaged apples. It was clear that the total peak area for undamaged apples was two times higher than that from damaged apples on day 2. Volatile abundance from undamaged apples did not vary much from day 2 to day 9, while it increased more than 120% from damaged apples during this period of time and almost matched that from undamaged apples nine days after cutting treatment. Volatile profiles from damaged apples became more complicated (more peaks were detected) than healthy apples as time advanced. Figure 4.5 shows qualitative differences between undamaged and damaged apples 9 days after the cut. More peaks can be observed from damaged apples than from undamaged apples. In Tables 4.3 and 4.4, the 23 most abundant volatile components detected from undamaged and damaged apples were listed and compared. The most abundant volatile components for undamaged apples were: 2-methylbutan-1-ol, acetic acid butyl ester, hexanoic acid ethyl ester, butanoic acid 2-methyl-2-methylbutyl ester, propyl hexoate, butyl 2-methylbutanoate, acetic acid butyl ester, and hexyl 2-methyl butyrate. The most abundant volatile components for damaged apples were: 1-butanol 2-methyl-acetate, benzene ethyl, dodecane 2,6,10-trimethyl, styrene, 3,7,11-tridecaetrienenitrile 4,8,12-trimethyl, cyclopentane 1,2,3-trimethyl, butyl hexanoate and hexanoic acid ethyl ester. The predominant volatile compounds in both classes are esters and alcohols: e.g., ethyl acetate, hexyl acetate, propyl acetate, farnesol and 1-butanol,2-methyl-acetate. However, the volatile gas headspace in damaged apple showed traces of alcoholic compounds (ethanol, 1-hexanol, 1-pentanol and more 1-butanol) which were not present in undamaged apples.
Figure 4.3. Number of compounds detected from undamaged (healthy) apples and damaged apples

Figure 4.4. Volatile compounds quantity comparison between undamaged (healthy) apples and damaged apples
Figure 4.5. GC-MS chromatograms of headspace volatiles from undamaged (healthy) and damaged Red Delicious apples 9 days after an artificially induced cut.
Table 4.3. Volatile compounds in the headspace of damaged Red Delicious apples

<table>
<thead>
<tr>
<th>No.</th>
<th>Retention time (second)</th>
<th>Peak Area (count)</th>
<th>Compounds identified ( ^{a} )</th>
<th>Molecular weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.503</td>
<td>30064972</td>
<td>Ethanol</td>
<td>46</td>
</tr>
<tr>
<td>2</td>
<td>3.521</td>
<td>30676402</td>
<td>2-Methylbutan-1-ol</td>
<td>88</td>
</tr>
<tr>
<td>3</td>
<td>5.898</td>
<td>106319670</td>
<td>Benzene, ethyl-</td>
<td>106</td>
</tr>
<tr>
<td>4</td>
<td>6.306</td>
<td>121081818</td>
<td>1-Butanol, 2-methyl-, acetate</td>
<td>130</td>
</tr>
<tr>
<td>5</td>
<td>6.534</td>
<td>59324134</td>
<td>Styrene</td>
<td>104</td>
</tr>
<tr>
<td>6</td>
<td>8.99</td>
<td>32498478</td>
<td>Hexanoic acid, ethyl ester</td>
<td>144</td>
</tr>
<tr>
<td>7</td>
<td>9.298</td>
<td>21818860</td>
<td>Acetic acid, hexyl ester</td>
<td>144</td>
</tr>
<tr>
<td>8</td>
<td>9.437</td>
<td>22056111</td>
<td>Benzene, 1-methoxy-3-methyl-</td>
<td>122</td>
</tr>
<tr>
<td>9</td>
<td>10.287</td>
<td>22993984</td>
<td>Butanoic acid, pentyl ester</td>
<td>158</td>
</tr>
<tr>
<td>10</td>
<td>11.072</td>
<td>29100636</td>
<td>Propyl hexanoate</td>
<td>158</td>
</tr>
<tr>
<td>11</td>
<td>11.271</td>
<td>33705942</td>
<td>Butanoic acid, 2-methyl-, 2-</td>
<td>172</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>methylbutyl ester</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>13.069</td>
<td>49110701</td>
<td>Butyl hexanoate</td>
<td>172</td>
</tr>
<tr>
<td>13</td>
<td>13.981</td>
<td>25258242</td>
<td>Hexyl-2-methylbutyrate</td>
<td>186</td>
</tr>
<tr>
<td>14</td>
<td>16.093</td>
<td>26788026</td>
<td>Cyclohexane, 1,2,3-trimethyl-</td>
<td>126</td>
</tr>
<tr>
<td>15</td>
<td>16.534</td>
<td>24516235</td>
<td>isomer of 2-propyl-1,1,3-</td>
<td>168</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>trimethylcyclohexane</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>16.613</td>
<td>77788909</td>
<td>Dodecane, 2,6,10-trimethyl-</td>
<td>212</td>
</tr>
<tr>
<td>17</td>
<td>16.823</td>
<td>25485297</td>
<td>1-Heptanol, 6-methyl-</td>
<td>130</td>
</tr>
<tr>
<td>18</td>
<td>16.964</td>
<td>53543634</td>
<td>Cyclopentane, 1-pentyl-2-</td>
<td>182</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>propyl-</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>17.006</td>
<td>31061470</td>
<td>1-Hexene, 3,3,5-trimethyl-</td>
<td>126</td>
</tr>
<tr>
<td>20</td>
<td>17.152</td>
<td>21164105</td>
<td>Farnesol</td>
<td>222</td>
</tr>
<tr>
<td>21</td>
<td>17.991</td>
<td>25299705</td>
<td>1,6-Octadiene, 3,5-dimethyl-,</td>
<td>138</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cis-</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>18.187</td>
<td>55924269</td>
<td>3,7,11-Tridecatrienenitrile,</td>
<td>231</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4,8,12-trimethyl-</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>18.942</td>
<td>39144539</td>
<td>Isopulegol 1</td>
<td>154</td>
</tr>
</tbody>
</table>

\( ^{a} \) As detected by GC-MS
### Table 4.4. Volatile compounds in the headspace of undamaged Red Delicious apples

<table>
<thead>
<tr>
<th>No.</th>
<th>Retention time (second)</th>
<th>Peak area (count)</th>
<th>Compounds identified (^{a})</th>
<th>MW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.526</td>
<td>103697058</td>
<td>1-Butanol, 2-methyl-</td>
<td>88</td>
</tr>
<tr>
<td>2</td>
<td>4.922</td>
<td>80146634</td>
<td>Acetic acid, butyl ester</td>
<td>116</td>
</tr>
<tr>
<td>3</td>
<td>6.368</td>
<td>1.102E+09</td>
<td>1-Butanol, 2-methyl-, acetate</td>
<td>130</td>
</tr>
<tr>
<td>4</td>
<td>7.081</td>
<td>38044822</td>
<td>Amylacetate</td>
<td>130</td>
</tr>
<tr>
<td>5</td>
<td>7.793</td>
<td>40144460</td>
<td>Propyl 2-methylbutyrate</td>
<td>144</td>
</tr>
<tr>
<td>6</td>
<td>8.372</td>
<td>62318699</td>
<td>1-Butanol, 3-methyl-, propanoate</td>
<td>144</td>
</tr>
<tr>
<td>7</td>
<td>8.907</td>
<td>63668915</td>
<td>Butyl butyrate</td>
<td>144</td>
</tr>
<tr>
<td>8</td>
<td>8.987</td>
<td>41829790</td>
<td>Hexanoic acid, ethyl ester</td>
<td>144</td>
</tr>
<tr>
<td>9</td>
<td>9.309</td>
<td>209195633</td>
<td>Acetic acid, hexyl ester</td>
<td>144</td>
</tr>
<tr>
<td>10</td>
<td>9.926</td>
<td>80602435</td>
<td>butyl 2-methylbutanoate</td>
<td>158</td>
</tr>
<tr>
<td>11</td>
<td>10.286</td>
<td>59866306</td>
<td>2-methyl-butyl butyrate</td>
<td>158</td>
</tr>
<tr>
<td>12</td>
<td>11.072</td>
<td>87867723</td>
<td>propyl hexoate</td>
<td>158</td>
</tr>
<tr>
<td>13</td>
<td>11.272</td>
<td>113098658</td>
<td>Butanoic acid, 2-methyl-, 2-methylbutyl ester</td>
<td>172</td>
</tr>
<tr>
<td>14</td>
<td>13.078</td>
<td>161934435</td>
<td>Hexanoic acid, butyl ester</td>
<td>172</td>
</tr>
<tr>
<td>15</td>
<td>13.98</td>
<td>66153901</td>
<td>Hexyl 2-methyl butyrate</td>
<td>186</td>
</tr>
<tr>
<td>16</td>
<td>14.286</td>
<td>37225524</td>
<td>3-Methylbutyl hexanoate</td>
<td>186</td>
</tr>
<tr>
<td>17</td>
<td>16.603</td>
<td>46261602</td>
<td>Dodecane, 2,7,10-trimethyl-</td>
<td>212</td>
</tr>
<tr>
<td>18</td>
<td>16.784</td>
<td>54015771</td>
<td>n-Hexyl n-hexanoate</td>
<td>200</td>
</tr>
<tr>
<td>19</td>
<td>16.957</td>
<td>34349766</td>
<td>cis-3-Decene</td>
<td>140</td>
</tr>
<tr>
<td>20</td>
<td>17.226</td>
<td>31582165</td>
<td>1-Octene, 3,7-dimethyl-</td>
<td>140</td>
</tr>
<tr>
<td>21</td>
<td>17.984</td>
<td>21254327</td>
<td>cis-Farnesol</td>
<td>222</td>
</tr>
<tr>
<td>22</td>
<td>18.179</td>
<td>47395576</td>
<td>3,7,11-Tridecatrienitrite, 4,8,12-trimethyl-</td>
<td>231</td>
</tr>
<tr>
<td>23</td>
<td>18.935</td>
<td>37069356</td>
<td>(S)-4,4-Dimethyl-2-(4-methyl-3-cyclohexen-1-yl)-1,5-hexadiene</td>
<td>204</td>
</tr>
</tbody>
</table>

\(^{a}\) As detected by GC-MS

#### 4.3.2 Enose sensor reduction

Not all sensors in the Cyranose 320 Enose were sensitive to volatile compounds emitted by apples. Therefore, the sensors which make the most contribution were determined and selected. In general, a PCA loading plot can provide information about the contribution from each sensor. In order to select a subset of sensors, features with minimum loadings (zero is the minimum value) are removed and only sensors with maximum loading values will be considered and selected.
Based on this knowledge, sensor selection procedures were undertaken for the second experimental sample set. Four data sets ranging from day 5 to day 11 were analyzed. Figure 4.6 shows the PC loading analysis for day 5. The first 4 PCs accounted for 97.7% variance and were chosen for analysis. Table 4.5 shows the PC loading analysis for the data from day 5. Maximum and minimum PC loadings were selected during the first four PCs and corresponding sensors were selected based on their PC loadings. On day 5, five sensors were selected: sensor 8, 10, 20, 26, 27. Sensor 26 was selected three times, which means this sensor was very important in differentiating samples based on volatiles generated.

Figure 4.6. Variance contribution from the first 10 PCs
Table 4.5. Principal component loadings and sensor selection at day 5 after the cut

<table>
<thead>
<tr>
<th>PC number</th>
<th>Loadings</th>
<th>Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC 1</td>
<td>Max 0.1972</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Min 0.1307</td>
<td>26</td>
</tr>
<tr>
<td>PC 2</td>
<td>Max 0.4071</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Min -0.3667</td>
<td>27</td>
</tr>
<tr>
<td>PC 3</td>
<td>Max 0.8109</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Min -0.2696</td>
<td>10</td>
</tr>
<tr>
<td>PC 4</td>
<td>Max 0.5438</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Min -0.2921</td>
<td>10</td>
</tr>
</tbody>
</table>

Four typical days (from day 5 to day 11) were chosen for sensor selection investigation because cut apples developed deterioration during this period. In Table 4.6, different sensors were chosen for different days, because volatile compounds from both undamaged apples and damaged apples would change as undamaged apples became more mature and damaged apples developed deterioration. Hence, different sensors which were sensitive to different volatile profiles developed by undamaged and damaged apples in various periods would be selected. However, sensor 26 was selected unanimously from four datasets, sensors 10 and 24 were chosen three times, sensors 30, 27, 28, 22 were selected twice, and sensors 4, 8, 12, 20 and 32 were chosen only once.

Table 4.6. Sensor selection for different datasets

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Sensors selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 days</td>
<td>20, 26, 27, 10, 8</td>
</tr>
<tr>
<td>7 days</td>
<td>28, 26, 24, 10, 12, 30</td>
</tr>
<tr>
<td>9 days</td>
<td>18, 26, 27, 22, 24, 30</td>
</tr>
<tr>
<td>11 days</td>
<td>28, 26, 24, 32, 22, 10, 4</td>
</tr>
</tbody>
</table>

Performances from different sensor selection schemes were compared in Table 4.7. For the data set from day 5, an 82% reduction in the number of sensors yielded a correct classification of 87.5%, compared to 83.3% from 28 sensors. The other three data
sets, from day 7 to day 11, had similar results: the reduced sensor sets perform better than or as well as 28 sensors.

Table 4.7. Enose performance for the two different sensor selection schemes

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Correct classification rate (%)</th>
<th>Number of sensors used</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>28 sensors</td>
<td>Reduced sensors</td>
</tr>
<tr>
<td>5 days</td>
<td>83.3</td>
<td>87.5</td>
</tr>
<tr>
<td>7 days</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>9 days</td>
<td>87.5</td>
<td>91.7</td>
</tr>
<tr>
<td>11 days</td>
<td>91.7</td>
<td>100</td>
</tr>
</tbody>
</table>

4.3.3 Discrimination of volatiles by the Enose and zNose™

4.3.3.1 Effect of different physical damage on classification

To investigate the effect of apple physical damage on volatile emissions, six control (undamaged) apples and two replications of six apples each with one, two or three cuts (36 total damaged apples) were measured by the Enose and zNose™ after 14 days. The classification results are shown in Table 4.8. Results show that both the Enose and zNose™ are capable of detecting differences between different levels of physical damage.

The highest classification accuracy was obtained for the control samples: a 100% classification accuracy was obtained for the Enose and 83.3% for the zNose™. The misclassifications are mostly between the two cut and three cut, one cut and control treatments. When all the three cut treatments were combined into one group, the classification accuracy was 98% overall for Enose, and 92% overall for the zNose™.

110
Table 4.8. Classification accuracy (% correct) comparison for the Enose and zNose™ 14 days after deterioration

<table>
<thead>
<tr>
<th>Class (original)</th>
<th>Class (result)</th>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>one cut</td>
<td>two cut</td>
</tr>
<tr>
<td>Enose</td>
<td></td>
<td></td>
</tr>
<tr>
<td>one cut (12)</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>two cut (12)</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>three cut (12)</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Control (12)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>zNose™</td>
<td></td>
<td></td>
</tr>
<tr>
<td>one cut (12)</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>two cut (12)</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>three cut (12)</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Control (12)</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

In the PCA plot shown in Figure 4.7, it is clearly observed that apples used as control were well separated from the other three cut treatments after 14 days. Although the induced cut treatments have a pattern of clustering, the distances between these three treatments are much shorter than the distance between the treatment and the control group.

This result indicates that both the Enose and zNose™ are sensitive to the volatile differences between undamaged apples and cut apples. However, volatile differences within the damaged apples, especially between 2 cuts and 3 cuts, are less evident and neither the Enose nor the zNose™ could effectively determine the difference. From the PCA plot, it can be seen that generally, the undamaged apples produce different volatile profiles (both qualitatively and quantitatively) from cut apples after 14 days. These findings were consistent with the GC-MS data, which indicated that more ethanol, 1-hexanol, 1-pentanol and 1-butanol compounds were detected from damaged apples than from undamaged apples 9 days after the cut.
Figure 4.7. Enose PCA plot 14 days after the cut

4.3.3.2 Time effect

Apples are a climacteric fruit and continue to live and ripen after picking. Consequently, they emit higher levels of volatiles when placed in room air temperature compared to a cool CA (Controlled Atmosphere) environment. It was speculated that volatile compounds emitted by undamaged and damaged apples would both change as days advance. Experiments were conducted to test this hypothesis. Data from June were used to investigate the effect of time on apple volatile evolution. In this investigation, two groups of apples (the control and damaged apples) were individually investigated. Each group was sampled 6 times from day 0 to day 10. Samples were placed in the lab where air temperature and humidity were not closely controlled.

The MANOVA test was conducted to see whether the Enose signals are dependent on the time that apples were placed in the room air temperature within each
group (control and damaged). The null hypothesis $H_0$ states that there was no significant difference between samples taken at different time in each group. The hypothesis $H_0$ was rejected at the 5% significant level for both groups (P-value<0.0001 for both undamaged apples and damaged apples), which means that volatile profiles from both undamaged apples and damaged apples evolved with time.

It is clear from the PCA score plot in Figure 8 that both undamaged (Figure 4.8 (a)) and damaged (Figure 4.8 (b)) apples showed some common patterns: data points from day 4, day 6 and day 8 were grouped relatively closely whereas data points from day 10 were clustered away from other groups. Data from day 0 and day 2 showed a relatively large variation, suggesting that artificially damaged apples developed deterioration four days after the cut and emitted volatiles that were much different from previous days, after ten days of exposure to room air. Volatile profiles from undamaged apples that were stored in room temperature for 10 days would be different from those of apples when they were first placed in room air. This result is consistent with GC-MS measurements where volatile compounds (peaks detected in gas chromatogram) were 20% more in the 9-day-old samples compared to 6-day-old samples. Volatile compounds measured from GC-MS experiments point to possible increases over time in amounts of butyl 2-methylbutanoate, acetic acid and hexyl 2-methyl butyrate. The zNose™ results showed similar patterns.
Figure 4.8. Enose PCA plot for undamaged (a) and damaged (b) apples on different days.
### 4.3.3.3 Orientation effect

Experiments were conducted in June to investigate the effect of apple orientation on volatile detection. Three groups of cut apples were placed in the glass jar in different ways: cut spots were placed upward, sideward and downward. Four classes (control, cut up, cut down and cut side) and two classes (control and cut) were evaluated by PLS-DA. In Table 4.9, a much higher correct classification rate exists in two classes than in four classes, which implies that misclassifications were mainly caused by attempting to differentiate between the three cut treatments. Variance between the three differently positioned cut apples was lower than the variance between cut and control apples.

Further investigation by SAS MANOVA indicated that the three cut treatment groups were not significantly different from each other at the 5% significant level (P-value=0.059), but they were significantly different from the control apples (P-value<0.0001). These results indicated that the orientation of the sample in the glass jar did not make a significant difference in apple volatile emission and subsequent Enose/zNose™ measurement. This is because the apple headspace reached equilibrium after six hours concentration and hence different orientation did not have much effect on apple volatile headspace formation.

<table>
<thead>
<tr>
<th>Date</th>
<th>Enose</th>
<th>zNose™</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4 classes (%)</td>
<td>2 classes (%)</td>
</tr>
<tr>
<td>Day 0</td>
<td>68.8</td>
<td>97.9</td>
</tr>
<tr>
<td>Day 2</td>
<td>75</td>
<td>93.8</td>
</tr>
<tr>
<td>Day 4</td>
<td>66.7</td>
<td>97.9</td>
</tr>
<tr>
<td>Day 6</td>
<td>87.5</td>
<td>97.9</td>
</tr>
<tr>
<td>Day 8</td>
<td>79.2</td>
<td>97.9</td>
</tr>
<tr>
<td>Day 10</td>
<td>91.7</td>
<td>93.8</td>
</tr>
</tbody>
</table>
4.3.3.4 Classification between undamaged and damaged apples

Data collected in March 2005 were analyzed to assess the potential of Enose and zNose™ to differentiate undamaged apples from damaged apples. Based on the above analysis, samples from day 5 to day 11 were combined to obtain 48 samples for each control and damaged groups. Two data compression techniques (PCA and PLS) and two discriminant analysis methods (LDA and CVA) were applied and compared.

Table 4.10 compares four different classification methods and the performance of the two instruments. The PLS compression technique generated better classification results than the PCA method both for the Enose and zNose™. PLS-CVA gave an 85% accurate classification using data from Enose and PLS-LDA with the zNose™ data gave 96% correct classification.

Based on the findings from the GC-MS data, damaged apples developed a more complex volatile profile with time. In other words, more volatile components would be released from the damaged apples on day 11 than day 5 and different volatile profiles may develop. Combining the data from day 5 with data from day 11 did not provide consistent results.

Table 4.10. Comparison of different data compression and classification models (% correct classification)

<table>
<thead>
<tr>
<th>Classification models</th>
<th>Enose (%)</th>
<th>zNose™ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>LDA</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>CVA</td>
<td>75</td>
</tr>
<tr>
<td>PLS</td>
<td>LDA</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>CVA</td>
<td>85</td>
</tr>
</tbody>
</table>
4.4 Conclusions

The GC-MS investigation shows that, during the initial stages of the cut, damaged and undamaged apples had only quantitative differences in volatile headspace; undamaged apples produced 100-130% higher mass abundance of volatile components. However, damaged apples develop a more complicated volatile profile and more alcoholic compounds (1-butanol, ethanol and 1-hexanol) than undamaged apples as time advances. Both the Enose and zNose™ are sensitive to these volatile profile changes. Four different classification models were built and compared, and the PLS data compression technique produced the best result. The extent of physical damage had some effect on apple volatile emissions, however, the volatile differences between different number of cuts were less evident. The effective period for both the Enose and zNose™ to classify undamaged apples from damaged apples may start four days after damage was induced. Different orientation of damaged apples did not produce significant difference in apple volatile emission. The Enose sensor selection was investigated and the number of sensors was successfully reduced by 82% while obtaining a better classification accuracy. The reduced sensor numbers could potentially shorten the data processing time, and could be used to build an application-specific electronic nose and hence reduce the cost. In addition, reduced sensor numbers may also be helpful to improving classification accuracy by reducing number of variables. Further research will be conducted to enhance the system performance by using non-linear classification models and sensor fusion technology.
CHAPTER 5

ANN-INTEGRATED ELECTRONIC NOSE AND ZNOSE™ SYSTEM

FOR APPLE QUALITY EVALUATION

What a piece of work is man! How noble in reason! How infinite in faculty! In form
and moving how express and admirable! In action, how like an angel! In
apprehension how like a god! The beauty of the world! The paragon of animals!

William Shakespeare (1564-1616)

ABSTRACT

The fresh produce industry generates more than one billion dollars each year in
the U.S. market. However, fresh produce departments in grocery stores experience as
much as 10% loss because the apples contain undetected defects and deteriorate in
quality before they can be sold. Apple defects can create sites for pathogen development
which can cause food-borne illness. It is important to develop a non-destructive system
for rapid detection and classification of defective fresh produce. In this study, an artificial
neural network (ANN)-based electronic nose and zNose™ system was developed to
detect physically damaged apples. Principal component analysis was used for clustering
plot and feature extraction. The first five principal components were selected for the
electronic nose data input and the first ten principal components were selected for the
zNose™ spectrum data. Different ANN models (back-propagation networks (BP),
probabilistic neural networks (PNN) and learning vector quantification networks (LVQ))
were built and compared based on their classification accuracy, sensitivity and specificity, generalization and incremental learning performance. For the Enose data, the BP and PNN classification rate of 85.3% and 85.1%, respectively, was better than the LVQ classification rate of 73.7%; for the zNose™ data, the three ANN models had similar performance which were less favorable than the Enose, with classification rates of 77%, 76.8% and 74.3%. The three ANN models’ performances were also measured by their sensitivity, specificity, generalization and incremental learning.

5.1 Introduction

Apple (Malus domestica, Borkh) production constitutes a large part of the fresh produce industry in the U.S. According to the USDA National Agricultural Statistical Service (NASS), total apple production in the U.S. in 2003 was valued at 1.7 billion dollars (USDA-NASS, 2006). Apples are consumed not only as a fresh product, but by widespread application in apple cider and sauce as well. However, apples are prone to internal defects and external bruises during harvest and transportation. These defects reduce the value of apples and hurt the economic interest of both fresh produce retailers and consumers. Fresh produce departments in grocery stores experienced about 10% economic loss because of the apple spoilage and deterioration (NE-179 Project report, 2001). Even worse, apple defects can create sites for pathogen development which can cause food-borne illness, and contaminated non-pasteurized cider can induce illness outbreaks (Centers for Disease Control and Prevention, 1998). Increasing cases of food-borne diseases caused by contaminated apple products in the U.S. provide incentive to develop non-destructive and fast techniques to detect apple defects.
Many attempts have been made to develop non-destructive techniques for food quality and safety inspection, such as x-ray detection (Tao and Ibarra, 2000), nuclear magnetic resonance detection (Cho et al., 1990), mid-infrared spectroscopy (Tewari and Irudayaraj, 2005), hyperspectral imaging (Lu, 2004) and machine vision technology (Ward and Nussinovitch, 1996). Although these technologies achieved success in some applications, the inherent disadvantages of these instruments are that they are expensive and time consuming, which inhibit wide application in the food industry.

Various studies have proven that compositional changes in volatiles occur during fruit ripening, and vary depending on the presence of diseases and physical damage (Simon et al., 1996). By detecting these changes, deterioration in fruit can be detected and physically damaged apples can be differentiated from healthy apples. The electronic nose (Enose) technology, which mimics the human olfactory system by a chemical sensor array, is a less expensive instrument and can get results in a few minutes, and hence has been increasingly applied in the food industry in the last decade. Fruit (apple, pear, and banana) ripeness prediction was studied by several research groups using the electronic nose (Brezmes et al., 2001; Llobet et al., 1999; Oshita et al., 2000). The Enose was also used for quality sorting of blueberries (Simon et al., 1996), peanut off-flavor detection (Osborn et al., 2001), sausage fermentation monitoring (Eklov et al., 1998), grain quality inspection (Jonsson et al., 1997) and other food products (Benady et al., 1995; Gardner et al., 1992; Roussel et al., 2003a). Some preliminary studies of food-borne microorganisms detection using the Enose was also explored (Dutta et al., 2005; Powell et al., 2002).

The zNose™ is a more recently developed gas sensor which is based on surface acoustic wave (SAW) propagation. A change in oscillation of the fundamental frequency
occurs when the compounds are absorbed on the detector surface (Electronic Sensor Technology, 2001). Limited research on the use of the zNose™ to classify honey and wine, based on their constituents, has demonstrated its potential in food quality evaluation (Lammertyn et al., 2004; Tewari and Irudayaraj, 2005; Veraverbeke et al., 2004).

The success of the Enose and zNose™ application largely relies on the use of chemometric techniques. Normally, statistical methods (multivariate analysis) were used to build classification models. However, to solve a non-linear problem, an artificial neural network (ANN) provides a more powerful tool to accomplish this goal (Haugen and Kvaal, 1998).

Many studies have been conducted to build ANN classification models for Enose applications. Research on apple ripeness determination was conducted by employing a tin oxide resistive gas sensor and neural networks (Hines et al., 1999). A similar study was tried for banana ripeness prediction using a neural network-based electronic nose (Llobet et al., 1999). Identification of spoiled beef using an Enose and a radial basis function neural networks model achieved satisfactory results with maximum classification accuracy of 100% (Panigrahi et al., 2006). Other investigations using the Enose with ANN models include cigarette brand identification (Luo et al., 2004), swine house odor quantification (Sohn et al., 2003) and clinical bacteria detection (Pavlou et al., 2004).

The objectives of this research were to:

1) Make aroma fingerprints for the healthy apples and physically damaged apples using the Enose and zNose™;

2) Reduce the input data dimensionality for both Enose data and zNose™ data;
3) Develop classification models using three artificial neural networks for the Enose and zNose™, respectively, to distinguish between healthy and damaged apples, and compare different ANN model performances.

5.2 Materials and Methods

The general methodology flow chart for this research is illustrated in Figure 5.1. In the first step, headspace gas from apples in the concentration chamber was sampled by the Enose and zNose™. Then, a baseline correction was conducted for the zNose™. In the next step, 3-D PCA plots were drawn for the pooled data to observe whether two classes of apples were separated. Feature extraction using PCA was used afterwards. After this step, three ANN models were built and evaluated using various indices.

5.2.1 Sample preparation and data sets summary

Red ‘Delicious' apples were purchased from a local grocery store and stored in the refrigerator at 4-5°C to inhibit respiration. Apple samples were kept in room air for 6 hours to reach the ambient air temperature before each test. The equilibrium time for headspace concentration was also 6 hours. Apples were maintained at room air temperature (20±1 °C) for 48 hours between each measurement.

A 2 L glass jar was used as headspace gas concentration chamber, sealed by a plastic cap with a Teflon septum. A 5 mm diameter hole was drilled at the center of the plastic cap. The Enose and zNose™ needles could be inserted from this hole which was sealed by an inert silicon stopper after sampling.
Apples were intentionally damaged by inducing a 10 mm deep cross-slice cut at the apple surface. The damaged apples were exposed to room air for deterioration development. The measurements were conducted every other day from day 4 to day 14 after the cut treatment. Other apples without the cut treatment were considered as “healthy” apples.

Three sets of experiments were conducted in March, June, and September 2005, respectively. Sampling time and number of samples for each group are shown in Table 5.1.
Table 5.1. Sampling protocol for the Enose and zNose™

<table>
<thead>
<tr>
<th>Period of sampling</th>
<th>Days of sampling</th>
<th>No. of replications</th>
<th>No. of samples</th>
<th>Data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>March, 2005</td>
<td>5,7,9,11,14</td>
<td>24</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>June, 2005</td>
<td>4,6,8,10</td>
<td>24</td>
<td>96</td>
<td>2</td>
</tr>
<tr>
<td>September, 2005</td>
<td>5,7, 9, 11,13</td>
<td>48</td>
<td>240</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>456</td>
<td>Pooled</td>
</tr>
</tbody>
</table>

Three data sets with 456 data points in total were pooled for future classification model training and testing. Pooled data were divided into four folds, and for each fold, 25% of the data (114 values) were extracted to be the test set and the rest (342 values) were treated as the training set. After four extractions, each quarter of data were extracted as the testing set once, and four different folds were made to train and test the ANN models. The same data extraction method was performed for both the Enose and zNose™ data (Table 5.2).

Table 5.2. Composition of testing and training sets

<table>
<thead>
<tr>
<th>Category</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthy apples</td>
<td>57</td>
<td>57</td>
<td>57</td>
<td>57</td>
</tr>
<tr>
<td>Damaged apples</td>
<td>57</td>
<td>57</td>
<td>57</td>
<td>57</td>
</tr>
<tr>
<td>Total tested</td>
<td>114</td>
<td>114</td>
<td>114</td>
<td>114</td>
</tr>
<tr>
<td>Total for training</td>
<td>342</td>
<td>342</td>
<td>342</td>
<td>342</td>
</tr>
</tbody>
</table>

5.2.2 Enose experiment

The Cyranose 320 electronic nose (Smith Detection, Herts, UK) was used in this research. The electronic nose (Enose) consists of 32 individual thin-film carbon-black
polymer composite chemiresistors arranged into an array. The sensor materials are thin films deposited across two electrical leads on an alumina substrate, creating the conducting chemiresistors. When the composite film is exposed to a vapor-phase analyte, the polymer matrix acts like a sponge and swells while absorbing the analyte. The increase in volume causes an increase in resistance because the conductive carbon-black pathways through the material are disrupted. The polymer will shrink to its original size and restore the conductivity after the analyte is removed (Cyrano Sciences Inc., 2000).

Signals from the Enose were preprocessed by normalization as shown in Figure 5.2. The final signal of each sensor was calculated by the following equation:

\[
S_n = \frac{R_{n,max} - R_{n,0}}{R_{n,0}} \quad (n=1,2,\ldots,32) \tag{5.1}
\]

where \(S_n\) is the No. \(n\) sensor’s final signal which is used for further classification, \(R_{n,0}\) is the No. \(n\) sensor’s average resistance during stage A: baseline purge; and \(R_{n,max}\) is the No. \(n\) sensor’s maximum resistance during stage B: sample exposure.

Figure 5.2. Three stages of the Enose sensor response: (A) baseline purge, (B) sample exposure, and (C) sensor refresh.
5.2.3 zNose™ experiment

The zNose™ (Electronic Sensor Technology, Newbury Park, CA) used in this research consists of a surface acoustic wave (SAW) detector, pneumatic controls and support electronics. When different compounds are adsorbed on the surface of a SAW detector, the added mass of the material lowers the oscillating frequency of the SAW crystal. The system controller interprets the detector response and attempts to identify and quantify each material based on a predefined program. The time of arrival at the detector identifies the compound, and the frequency shift caused by each analyte is characteristic of the amount of material deposited on the detector. These two factors make quantification possible (Electronic Sensor Technology, 2001). The zNose™ was equipped with a 5 cm long sampling needle at the inlet, which is inserted into the concentration chamber for sampling. The different chemical components in the gas sample were separated and sequentially detected by the SAW detector. The system was baked for 5 s after each data sampling period to clean the SAW detector. A helium gas purge was conducted between each sample measurement to ensure that previous volatiles absorbed on the detector were removed and a stable baseline was obtained.

The retention time scale of the zNose™ was calibrated using standard headspace vapors containing C₆-C₁₄ n-alkanes in order to accurately measure sample volatiles. The standard procedures can be found in the user's manual (Electronic Sensor Technology, 2001).

During the experiments, environmental fluctuations may cause the different components to be released and detected at slightly different retention times. This shift may be accumulated and lead to misinterpretation if different spectra are compared. To
counteract this problem, algorithms developed by Lammertyn et al. (Lammertyn et al., 2004) were applied for spectra horizontal shift and baseline correction.

5.2.4 Feature extraction and PCA

Principal component analysis (PCA) was used for two purposes in this research. The first was for feature extraction. Both the electronic nose and zNose™ data have high dimensionality. Not all 32 sensors from the Enose are sensitive to apple volatiles, or these 32 sensors may be highly correlated with each other. The zNose™ generates spectral data with 516 data points for each sample, which is too large to feed into neural networks.

PCA is based on a linear projection of multidimensional data onto different coordinates based on maximum variance and minimum correlation. As a result, less significant components can be eliminated, reducing the data representation to only those responsible for the most significant contribution. For the Enose, the PCA was used to compress the original data and the first five PCs, which represent 99.5% of the variances, were extracted for classification. For the zNose™, the PCA was applied to compress the data and the first 10 PCs representing 68.4% of the variances were used for further classification.

Another use of PCA is for data clustering analysis. The 3-D PCA plot uses the first three principal components to draw the plot, which enables visual observation of the data to determine whether the data show a clustering pattern. The 3-D PCA plot also can help explain the performance of classification models.
5.2.5 ANN models (BP, PNN and LVQ)

In this research, three types of ANN paradigms were developed and compared. The Backpropagation (BP) network is known for its ability to generalize well on a wide variety of problems. This network is generally robust, although one drawback is that the training is slow. Typically, three layers (input, hidden, and output layer) are sufficient for the vast majority of problems and each layer is connected to the immediately previous layer.

The Probabilistic Neural Network (PNN) is known for its ability to train quickly on sparse data sets. PNN separates data into a specified number of output categories. PNN networks are three layer networks wherein the training patterns are presented to the input layer and the output layer has one neuron for each possible category. The number of neurons in the hidden layer should be the same as the number of training patterns. The network produces activations in the output layer corresponding to the probability density function estimate for that category. The highest output represents the most probable category.

The Learning Vector Quantification Network (LVQ) is a supervised competitive ANN which transforms high dimensional data to a two-dimensional grid, without regarding data topology. LVQ is intended for statistical classification. It uses pre-assigned cluster labels to data items to facilitate the two-dimensional transformation so as to minimize the average expected misclassification probability. An LVQ network has a first competitive layer and a second linear layer. The competitive layer learns to classify input vectors. The linear layer transforms the competitive layer’s classes into target
classifications defined by the user. Both the competitive and linear layers have one neuron per class. The disadvantage of LVQ is the slow speed of training.

5.3 Results and Discussions

5.3.1 The Enose and zNose™ sampling

Different patterns between healthy and damaged apples were observed from the "smellprint" made by the electronic nose and zNose™ (Figures 5.3 and 5.4). The results support the hypothesis that the compounds emitted from red 'Delicious' apples will change according to the physiological state and the Enose and zNose™ are capable of detecting this difference.

Figure 5.3. Enose "smellprints"
5.3.2 PCA clustering analysis

Before developing ANN classification models, PCA plots were made to observe whether healthy apples and damaged apples were clustered in groups. In Figure 5.5, 3D PCA plots were drawn based on Enose data from individual days, i.e. from day 5 after treatment to day 13 in the September set. It was observed that the healthy apples and damaged apple data points showed clustering, and they can be easily differentiated. However, if all data from day 5 to day 13 were pooled together, the clustering pattern was not obvious and the data points from healthy apples and damaged apples overlapped with each other, which means the nature of the pooled data from healthy apples and damaged apples are very close and it is hard to classify them. The possible reason for this is that...
both the healthy apple and damaged apple experience physiological changes including respiration and apple degradation by the growth of microorganisms during the experiment, and these physiological changes may induce changes in the headspace volatile profile. In other words, the headspace volatiles emitted by healthy apples of day 13 may be similar to the headspace volatiles from damaged apples of day 5. The pooled data for a period of 8 days would induce a great difficulty for classification models to correctly classify the two classes of apples.

A similar pattern was shown for the zNose™ data. Pooled data from all three data sets also had considerable overlapping and caused a very difficult problem for conventional statistical classification models (Figure 5.6).

Figure 5.5. Enose PCA plot for September data
Figure 5.6. zNose™ PCA plot for all three data sets

5.3.3 ANN model development and performance evaluation

5.3.3.1 BP network over-fitting prevention

During network training, the training set error is driven to a very small value, but when new data are presented to the network, the error is large. This situation is called over-fitting and it is a common problem in network training. One reason for over-fitting is the limited supply of data. If the number of parameters in the network is close to or even larger than the total number of points in the training set, over-fitting might happen. In this study, an early stopping technique provided by the MATLAB NN Toolbox (Demuth and Beale, 2001) was applied to the BP network to prevent over-fitting.
In this technique, the available input data are divided into two subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set which is used for monitoring error during training. When the network begins to overfit the data, the error on the validation set will typically begin to rise and then the training will be stopped as shown in Figure 5.7. In this research, 75% of the data (342) was divided as training subset and the rest 25% data (114) was used for validation.

![Figure 5.7. BP network early stopping training](image)

5.3.3.2 Neural network architecture

The three-layer network structure was applied to all ANNs. The input layer had five neurons for the Enose and ten neurons for the zNose™ since five and ten principal components were used, respectively. The output layer had two neurons for two classes.
(healthy and damaged apples) with 1 representing “true” and 0 representing “false”. The number of neurons in the hidden layers is important for ANN model performance. Too few hidden neurons results in high training and generalization error due to underfitting and high statistical bias, whereas too many hidden neurons results in low training error, but high generalization error due to overfitting. The number of hidden neurons is typically determined by trial-and-error.

For the BP networks, the MATLAB imbedded resilient method was used for training, \( \text{tansig} \) (tan-sigmoid) was used for the hidden layer transfer function, and \( \text{logsig} \) (log-sigmoid) was used for the output layer transfer function since the range of output is from 0 or 1 (Figure 5.8 A and B). In PNN networks, the \( \text{radbas} \) (radial basis) function was used in the radial basis layer (Figure 5.8 C).

The tan-sigmoid, log-sigmoid and radial basis functions can be expressed as follows:

\[ \text{logsig}(n) = \frac{1}{1 + e^{-n}} \]  \hspace{1cm} (5.2)
\[ \text{tansig}(n) = \frac{2}{1 + e^{-2n}} - 1 \]  \hspace{1cm} (5.3)
\[ \text{radbas}(n) = e^{-n^2} \]  \hspace{1cm} (5.4)

![Graphs](image)

Figure 5.8. (A) Log-sigmoid (logsig), (B) tan-sigmoid (tansig) and (C) radial basis transfer functions (Demuth and Beale, 2001)
For the Enose data, the optimal number of hidden neurons for the BP networks was determined by testing different numbers of neurons. Hidden layers with 5, 10, 20, 25 and 30 neurons and the tangent sigmoid transfer function were used to simulate the performance of the ANN. All of the training simulations were performed with the same architecture: three-layer back-propagation network, with a tan-sigmoid transfer function in the hidden layer and a log-sigmoid transfer function in the output layer. The early stopping algorithm was used. Each simulation was run 20 times and the average values were recorded for each parameter.

Table 5.3 illustrates that the mean square error (MSE) for training and classification validation decreased as the number of neurons increased. The BP network performed the best when the number of hidden neurons was 30. Although the training set MSE for 30 hidden neurons did not improve much (0.0715 to 0.0712), the validation error improved from 19 to 17. Since the improvement was small, it was not necessary to test more hidden neurons. The BP network with 30 neurons in the hidden layer was used in subsequent analysis for the Enose data.

Table 5.3. BP network hidden neurons testing (Enose)

<table>
<thead>
<tr>
<th>Hidden neurons</th>
<th>Simulation repetition</th>
<th>Training MSE</th>
<th>Validation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>20</td>
<td>0.1670</td>
<td>31 (114)</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>0.1277</td>
<td>26 (114)</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>0.0808</td>
<td>19 (114)</td>
</tr>
<tr>
<td>25</td>
<td>20</td>
<td>0.0715</td>
<td>19 (114)</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>0.0712</td>
<td>17 (114)</td>
</tr>
</tbody>
</table>
Similar testing was conducted for the zNose™ data as shown in Table 5.4, and the optimal number of neurons in hidden layer for BP network was determined as 25. The learning rate for BP network was set to 0.1 for both the Enose and zNose™ data.

Table 5.4. BP network hidden neurons testing (zNose™)

<table>
<thead>
<tr>
<th>Hidden neuron</th>
<th>Simulation Repetition</th>
<th>Training MSE</th>
<th>Validation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>20</td>
<td>0.158</td>
<td>24.4 (114)</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>0.1556</td>
<td>22.8 (114)</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>0.1399</td>
<td>20 (114)</td>
</tr>
<tr>
<td>25</td>
<td>20</td>
<td>0.1362</td>
<td>18 (114)</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>0.1385</td>
<td>24 (114)</td>
</tr>
</tbody>
</table>

For the PNN, the number of neurons in the hidden layer was set to the size of the training data set, which was 342. The number of neurons in the hidden layer for the LVQ network was determined the same way as the BP network, and the number was set to 30. The architecture of the three neural networks is summarized in Figures 5.9-11.

In these figures, IW stands for input weight matrix, b stands for bias vector, and \( \text{dist} \) produces the dot product of input layer and input weight matrix.

![Figure 5.9. BP network architecture](image)

Figure 5.9. BP network architecture
Three ANN models classification performance comparison

The BP, PNN and LVQ neural network model classification performances were compared. The pooled data (456 data sets from three experiments) were classified as training subsets (342) and validation subsets (114), as shown in Table 5.2. The data extraction was performed in an alternate way: Each time 25% of the data was extracted as the validation data and the rest was training data. Four folds were formed after four extractions, and each data set was used as validation data. As shown in Figure 5.12, the output of the network was set to 1 for healthy apples and 0 as damaged apples. The output was rounded to integers 1 or 0 when they were counted.
The classification accuracy for the Enose data of four folds using three neural networks is compared in Figure 5.13. All of these values are averaged after running each model 10 times. It was observed that each neural network performed consistently for four folds data. Except for fold 2, the PNN performed slightly better or comparable with the BP, and the average performance of these two classifiers are equivalent (85.3% vs. 85.1% correct classification). In all four folds, LVQ produced less favorable results ranging from 72%-76% with average performance of 73.7%. Considering their processing speed, the BP typically required 100-200 epochs before being stopped early by validation error; the LVQ usually took 1000 epochs for training; the PNN did not go through the training routine, and processed even faster than the BP network. The Enose classification confusion matrices are shown in Table 5.5.

Figure 5.12. Network classification illustration
Figure 5.13. BP, PNN an LVQ model classification results for Enose data

Table 5.5. Classification confusion Table with BP, (PNN) and [LVQ] for Enose data

<table>
<thead>
<tr>
<th>Predicted healthy</th>
<th>Actual healthy</th>
<th>Actual damaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>48 (48) [41]</td>
<td>10 (8) [14]</td>
<td></td>
</tr>
<tr>
<td>9 (9) [16]</td>
<td>47 (49) [43]</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>57</td>
<td>57</td>
</tr>
</tbody>
</table>

In Figure 5.14, the BP network generated better classification results than PNN in three folds with the exception of fold 2 in which the PNN was slightly better than the BP (75% vs. 67%). These two classifiers performed equivalently on average with classification accuracy of 77% for the BP and 76.8% for PNN, which was slightly better than that of the LVQ (74.3%). The zNose™ ANN models generally did not perform as well as the Enose models. The zNose™ classification confusion matrices were shown in Table 5.6.
Figure 5.14. BP, PNN an LVQ model classification results for zNose™ data

Table 5.6. Classification confusion table with BP, (PNN) and [LVQ] for zNose™ data

<table>
<thead>
<tr>
<th></th>
<th>Actual healthy</th>
<th>Actual damaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted healthy</td>
<td>48 (47) [51]</td>
<td>16 (16) [23]</td>
</tr>
<tr>
<td>Predicted damaged</td>
<td>10 (10) [6]</td>
<td>41 (41) [34]</td>
</tr>
<tr>
<td>Total</td>
<td>57</td>
<td>57</td>
</tr>
</tbody>
</table>

5.3.3.4 Sensitivity and specificity

The performance of ANN models can also be evaluated by the sensitivity and specificity. The sensitivity is a measure of the ability of the classifier to detect damaged apples when they are present. The specificity of a network is the likelihood that the
absence of the damaged apples will be detected when they are absent. They are defined by the following equations:

\[
\text{Sensitivity} = \frac{TP}{TP + FN} \quad (5.5)
\]

\[
\text{Specificity} = \frac{TN}{TN + FP} \quad (5.6)
\]

where

TP is true positive which means the number of damaged apples that were correctly classified;

FN is false negative which means the number of good apples that were wrongly classified;

TN is true negative which means the number of good apples that were correctly classified;

FP is false positive which means the number of damaged apples that were wrongly classified.

Ideally, both the sensitivity and the specificity should be equal to 1 and their values are in the range of 0 to 1. Table 5.7 shows that the sensitivity and specificity values for the Enose data. The BP and PNN had almost identical sensitivity and specificity, which are in accordance with their classification results shown in Figure 5.13. It is clear that the BP and PNN outperformed the LVQ in the sensitivity and specificity indices. Table 5.8 shows these two indices for the zNose™ data. The BP and PNN generated the same values for these two indices, and their sensitivity (71.9%) is higher than that of the LVQ.
network (59.6%). However, the LVQ had better specificity (89.5%) than the other two networks (82.5%).

Table 5.7. Sensitivity and specificity for BP, PNN and LVQ networks with Enose data

<table>
<thead>
<tr>
<th>ANNs models</th>
<th>Sensitivity</th>
<th>Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP</td>
<td>84.5%</td>
<td>85.7%</td>
</tr>
<tr>
<td>PNN</td>
<td>84.5%</td>
<td>85.7%</td>
</tr>
<tr>
<td>LVQ</td>
<td>72.9%</td>
<td>74.5%</td>
</tr>
</tbody>
</table>

Table 5.8. Sensitivity and specificity for BP, PNN and LVQ networks with zNose™ data

<table>
<thead>
<tr>
<th>ANNs models</th>
<th>Sensitivity</th>
<th>Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP</td>
<td>71.9%</td>
<td>82.5%</td>
</tr>
<tr>
<td>PNN</td>
<td>71.9%</td>
<td>82.5%</td>
</tr>
<tr>
<td>LVQ</td>
<td>59.6%</td>
<td>89.5%</td>
</tr>
</tbody>
</table>

5.3.3.5 Generalization

It was necessary to assess how effectively the networks were able to classify damaged apples from healthy apples when the data set was not used for training the original neural network. In another words, it is important to evaluate whether the networks can acquire the knowledge necessary to classify new data.

This assessment was performed in three scenarios. In the first scenario, the patterns in sets 1 (data from September with 240 values) and 2 (data from June with 96 values) were used to train neural networks and the patterns in set 3 (data from March with 120 values) were used as new data to test the networks. As shown in Table 5.9, the overall classification accuracy from the BP, PNN and LVQ were 50%, 46% and 51% respectively. All three classifiers gave poor performance for good apple classification:
only 1 out of 60 was correctly classified, which might indicate that good apples in data set 3 are very similar to damaged apples in training data sets. The BP, PNN and LVQ networks provided much better classification accuracy for damaged apples with 98%, 90% and 100% correct classification, respectively.

For scenario 2, the patterns in sets 1 (data from September with 240 values) and 3 (data from March with 120 values) were used to train neural networks and the patterns in set 2 (data from June with 96 values) were used as new data to test the networks. Under this situation, 83% good apple classification accuracy was achieved for the BP networks and 100% for both the PNN and LVQ, which was considerably better than the first scenario. Although their performances on bad apple classification were degraded, their overall performances were still better than scenario 1: 65% for BP, 57% for PNN and 56% for LVQ. MSE was also reduced accordingly compared to scenario 1.

For scenario 3, the patterns in sets 2 (data from June with 96 values) and 3 (data from March with 120 values) were used to train neural networks and the patterns in set 1 (data from September with 240 values) were used as new data to test the networks. This presented a difficult situation since there were more testing data than training data. However, the results showed that networks still performed better than the first scenario. All three classifiers generated almost identical overall classification accuracies of 53% (BP and LVQ) and 52% (PNN). The MSE was slightly less than the first scenario.

ANN generalization results for zNose™ data are summarized in Table 5.10. It was observed that in all three scenarios, the three classifiers’ performances were much less favorable than in the Enose data case. However, a similar pattern was observed. All three networks performed best in the scenario 2. The LVQ performed better than the other two
except for the scenario 2. The poor performance of ANN generalization might be induced by large variation between three data sets sampled by the zNose™.

Table 5.9. Results of the generalization of BP, [PNN] and (LVQ) networks in damaged apple classification, in terms of patterns correctly classified/number of patterns (Enose)

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Good</th>
<th>Bad</th>
<th>Overall performance</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2 /3</td>
<td>[1/60]</td>
<td>59/60</td>
<td>50%</td>
<td>0.471</td>
</tr>
<tr>
<td></td>
<td>[54/60]</td>
<td>46%</td>
<td>[0.542]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1/60)</td>
<td>(60/60)</td>
<td>(51%)</td>
<td>(0.492)</td>
</tr>
<tr>
<td>1, 3 /2</td>
<td>[48/48]</td>
<td>22/48</td>
<td>65%</td>
<td>0.269</td>
</tr>
<tr>
<td></td>
<td>[7/48]</td>
<td>57%</td>
<td>[0.427]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(48/48)</td>
<td>(6/48)</td>
<td>(56%)</td>
<td>(0.438)</td>
</tr>
<tr>
<td>2, 3 /1</td>
<td>[112/120]</td>
<td>14/120</td>
<td>53%</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>[28/120]</td>
<td>52%</td>
<td>[0.483]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(102/120)</td>
<td>(25/120)</td>
<td>(53%)</td>
<td>(0.472)</td>
</tr>
</tbody>
</table>

Table 5.10. Results of the generalization of BP, [PNN] and (LVQ) networks in damaged apple classification, in terms of patterns correctly classified/number of patterns (zNose™)

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Good</th>
<th>Bad</th>
<th>Overall performance</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2 /3</td>
<td>[10/60]</td>
<td>3/60</td>
<td>13%</td>
<td>0.840</td>
</tr>
<tr>
<td></td>
<td>[1/60]</td>
<td>[9%]</td>
<td>[0.908]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(9/60)</td>
<td>(10/60)</td>
<td>(16%)</td>
<td>(0.839)</td>
</tr>
<tr>
<td>1, 3 /2</td>
<td>[14/48]</td>
<td>35/48</td>
<td>51%</td>
<td>0.339</td>
</tr>
<tr>
<td></td>
<td>[11/48]</td>
<td>[16%]</td>
<td>[0.563]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(6/48)</td>
<td>(22/48)</td>
<td>(29%)</td>
<td>(0.712)</td>
</tr>
<tr>
<td>2, 3 /1</td>
<td>[0/120]</td>
<td>12/120</td>
<td>5%</td>
<td>0.941</td>
</tr>
<tr>
<td></td>
<td>[0/120]</td>
<td>[5%]</td>
<td>[0.950]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(6/120)</td>
<td>(16/120)</td>
<td>(9%)</td>
<td>(0.908)</td>
</tr>
</tbody>
</table>

This generalization showed that the BP network performed slightly better than the PNN in all scenarios. The reason that the BP outperformed the PNN is because the BP used an early stopping technique to prevent over-fitting, which benefits network generalization. Various reasons may cause the relatively poor performance in the first situation, for instance, apples from data set 3 (March) might be harvested at a different maturity stage than the other two data sets.
5.3.3.6 Incremental learning

The ability of ANN models to be able to learn the new patterns without forgetting previously learned patterns is very important and attractive for practical applications. If the trained ANN model is exposed to new patterns, the ANN model needs to be re-trained, which is a time consuming and costly practice. The incremental learning is defined to measure the abilities of ANN models to remember previously trained patterns when training with new patterns. Since the PNN could not implement the “training” function, only BP and LVQ were evaluated for incremental learning.

In order to investigate the ability of the networks to learn new knowledge without forgetting previous knowledge, BP and LVQ networks were trained with the first data set as shown in Table 5.1. They were further trained using only the second data set and tested by the first and second data set. The purpose was to check whether these two networks could learn the patterns in the second data set without forgetting the patterns trained in the first data set. In the third step, the networks were further trained using the third data set and tested with the patterns from all three data sets.

The BP and LVQ network incremental learning performance for Enose data are summarized in Table 5.11. It was found that the BP network accuracy degraded from 95% to 66% when it was trained by the second data set and tested by data set 1 and 2. However, the overall performance remained almost constant from (66% vs. 64%) when it was trained by the third data set and tested by all three data sets. The LVQ network did not degrade as significantly as the BP did from step 1 to step 2 (86% vs. 78%). However, it degraded considerably from 78% to 63% when tested by all three data sets.
Incremental learning ability was evaluated for the zNose™ data (Table 5.12). Substantial degradation was observed for the BP network (from 99.6% to 54%) when it was tested by the first two data sets; the degradation was not noticeable (from 54% to 50%) when it was tested by three all data sets. The LVQ network did not degrade as much as the BP network (from 95% to 63% and to 61%), and it seemed to have better incremental learning capability than the BP network for the zNose™ data.

Table 5.11. BP and (LVQ) incremental learning for Enose data

<table>
<thead>
<tr>
<th></th>
<th>good</th>
<th>bad</th>
<th>all</th>
<th>Overall performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/1</td>
<td>111/120</td>
<td>(101/120)</td>
<td>228/240</td>
<td>95%</td>
</tr>
<tr>
<td>2/1,2</td>
<td>82/168</td>
<td>140/168</td>
<td>222/336</td>
<td>66%</td>
</tr>
<tr>
<td>3/1,2,3</td>
<td>116/228</td>
<td>(123/228)</td>
<td>293/456</td>
<td>64%</td>
</tr>
</tbody>
</table>

Table 5.12. BP and (LVQ) incremental learning for zNose™ data

<table>
<thead>
<tr>
<th></th>
<th>good</th>
<th>bad</th>
<th>all</th>
<th>Overall performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/1</td>
<td>120/120</td>
<td>(120/120)</td>
<td>239/240</td>
<td>99.6%</td>
</tr>
<tr>
<td>2/1,2</td>
<td>21/168</td>
<td>160/168</td>
<td>181/336</td>
<td>54%</td>
</tr>
<tr>
<td>3/1,2,3</td>
<td>0/228</td>
<td>228/228</td>
<td>228/456</td>
<td>50%</td>
</tr>
</tbody>
</table>

5.4 Conclusions

Apple headspace gas samples from three different sets were gathered by the electronic nose and zNose™ instruments. The aroma fingerprint made by the Enose and zNose™ showed different patterns for healthy apples and damaged apples. Further
clustering analysis made by PCA illustrated that healthy apples and damaged apples showed separate clustering patterns on individual days, but this clustering pattern was less evident when all data sets from different days were pooled into one group.

Data preprocessing and feature extraction were investigated for both the Enose and zNose™. PCA was used for feature extraction. The first five PCs were used for Enose data extraction; the first ten PCs were used to extract zNose™ spectral data.

BP, PNN and LVQ networks were built and compared for their capability to classify damaged apples from good apples. The classification results indicated that the BP and PNN achieved comparable superior classification accuracy (85.3% and 85.1% for Enose data, 77% and 76.8% for zNose™ data), and they outperformed the LVQ network (73.7% for Enose data and 74.3% for zNose™ data). From a computing speed perspective, the PNN was the fastest network, and the LVQ needed much more time (more than 1000 epochs) to finish one training. Processing speed for BP is in between the speed of PNN and LVQ. The BP and PNN networks achieved comparable success on sensitivity and specificity indices for both the Enose and zNose™ data.

The generalization of the trained networks to the new data was investigated. Experiments showed that the BP network performed slightly better than the PNN in all three scenarios because the BP was trained using an early stopping technique. All three classifiers worked best in scenario 3. The poor generalization performance of the three networks for the zNose™ data might be due to the large variance from three data sets sampled by the zNose™.

Incremental learning was performed and evaluated for both the BP and LVQ networks. It was found that the LVQ network had less degradation than the BP network.
when it was exposed to new patterns. This observation held for both the Enose and zNose\textsuperscript{TM} data.

In general, although the grouped data from different days and different months posed a challenging classification problem, three neural networks provided an acceptable non-linear classification approach. The classification results may be improved if apple headspace gas were sampled in a more uniform and controlled way, and the structure of ANN models can be further explored and optimized.
ABSTRACT

The high dimensionality of electronic nose data increases the difficulty of their use in classification models. Reducing this high dimensionality helps reduce variable numbers, potentially improve classification accuracy by removing irrelevant sensors, and reduce computation time and sensor cost. In this research, the C3nose 320 electronic nose was optimized for apple defect detection by selecting the most relevant of its 32 internal sensors using various selection methods. The contribution of each sensor was evaluated statistically by calculating the F-value. By keeping only the top 90% cumulative F-values, 25 sensors were selected and the classification error rate was 25.4%. Sequential forward/backward search methods reduced the minimum classification error rate to 6.1%. Two more heuristic optimization algorithms, genetic algorithm (GA) and the covariance matrix adaptation evolutionary strategy (CMAES), were applied and compared. Although both algorithms gave a best classification error rate of 4.4%, the average classification error rate of CMA over 30 random seed runs was 5.0% (s.d.=0.006).
which was better than 5.2% (s.d.=0.004) from the GA. The final optimal solution sets obtained by using an integer GA showed that including more sensors did not guarantee better classification performance. The best reduction in classification error rate was 10% while the number of sensors was reduced 75%. This study provided a robust and efficient optimization approach to reduce high data dimensionality of the electronic nose data, which substantially improved electronic nose performance in apple defect detection while potentially reducing the overall electronic nose cost for future specific applications.

6.1 Introduction

The electronic nose (Enose), accompanied by pattern recognition algorithms, is devised to classify smell patterns. Since the concept of the electronic nose was first introduced in 1982 (Persaud and Dodd, 1982), the applications of the electronic nose in various fields have grown exponentially. The electronic nose has been applied in the food industry (Schaller et al., 1998), medical diagnosis (Thaler et al., 2001), environmental monitoring (Ameer and Adeloju, 2005) and space exploration (Ryan et al., 2004). Among them, food quality control using the electronic nose is one of the most common application areas. Researchers used the electronic nose to predict fruit ripeness (Brezmes et al., 2001; Llobet et al., 1999), detect meat spoilage (Balasubramanian et al., 2004), and monitor fermentation process (Eklov et al., 1998). In this research, a commercially available instrument, the Cyranose 320 electronic nose, was used to detect apple defects.

The Cyranose 320 electronic nose contains 32 conducting polymer sensors which can function at ambient air temperature. It works based on the principle that the
resistance of chemiresistors increases when vapor-phase analytes absorb on their surface and disrupt the conductive pathways (Cyrano Science Inc., 2000). These 32 conducting polymer sensors are partially specific to certain chemicals and are capable of recognizing odors by using a pattern recognition system. However, in this apple defect detection application, these 32 sensors may not all be sensitive to volatile compounds emitted from healthy or damaged apples. If irrelevant sensors are included in pattern recognition models, they may not only introduce noise and reduce the classification accuracy, but increase the computation time as well.

Assuming an exhaustive search of all sensor selection possibilities is performed, for this 32 sensor problem, the number of all sensor combinations would be:

\[
\sum_{p=1}^{n} \frac{n!}{p!(n-p)!} \quad (n=32)
\]

(6.1)

This results in 4 billion possible design schemes, which makes the full exploration of the decision space too computationally expensive to be practical.

Some attempts have been made to select relevant sensors and reduce high dimensionality of the electronic nose data. Statistical methods, such as principal component analysis and multivariate analysis of variance (MANOVA), were used for sensor selection by evaluating each sensor’s contribution (Boilot et al., 2003; Ciosek et al., 2004). More recently, a robust, global heuristic search method, genetic algorithms (GA), was introduced for variable selection. The GA is an optimization and search technique based on the principles of genetics and natural selection (Haupt and Haupt, 2004). It was developed over the course of 1960s and 1970s (Holland, 1975) and gained more popularity since the late 1980s (Goldberg, 1989). Some researchers proposed to
use the GA for sensor selection and achieved encouraging results (Boilot et al., 2003; Kermani et al., 1999; Roussel et al., 2003a).

This paper introduces a state-of-the-art evolutionary algorithm developed by Hansen (2005), termed the covariance matrix adaptation evolutionary strategy (CMAES) to solve the problem of the Cyranose 320 sensor selection. It is capable of dealing with problems of highly non-linear, concave, and rugged search landscapes (Hansen, 2005). Different coding methods (binary coded and real number coded) were devised and compared for this specific application. Other sensor selection approaches such as F-value selection, SFS/SBS search, and simple GA were also assessed in this research and compared with CMA evolutionary strategy algorithm in terms of the effectiveness, efficiency, and reliability.

6.2 Methodology

6.2.1 Sampling process and data collection

Red ‘Delicious' apples were purchased from a local grocery store and stored in a refrigerator at 4-5 °C to inhibit respiration. Apple samples were kept in room air for 6 hours to reach the ambient air temperature before each test. The equilibrium time for headspace concentration was also 6 hours. Apples were maintained at room air temperature (20±1 °C) for 48 hours between each measurement. A 2 L glass jar was used as a headspace gas concentration chamber, sealed by a plastic cap with a Teflon septum. A 5 mm diameter hole was drilled at the center of the plastic cap. The Enose sampling needle could be inserted from this hole which was sealed by an inert silicon stopper after sampling. Apples were intentionally damaged by inducing a 10 mm deep cross-slice cut
on the top and these damaged apples were exposed to room air for deterioration development. The measurements were conducted every other day from day 4 to day 14 after the cut treatment. Other apples without the cut treatment were considered “healthy” apples.

The Cyranose 320 electronic nose (Smith Detection, Herts, UK) was used to collect headspace gas samples from healthy and damaged apples. Sampling was conducted in three different times: March, June, and September in 2005. Sampling time and number of samples for each group are shown in Table 6.1.

<table>
<thead>
<tr>
<th>Period of sampling</th>
<th>Dates of sampling</th>
<th>No. of replications</th>
<th>No. of samples</th>
<th>Data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>March, 2005</td>
<td>5,7,9,11,14</td>
<td>24</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>June, 2005</td>
<td>4,6,8,10</td>
<td>24</td>
<td>96</td>
<td>2</td>
</tr>
<tr>
<td>September, 2005</td>
<td>5,7,9,11,13</td>
<td>48</td>
<td>240</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>456</td>
<td>Pooled</td>
</tr>
</tbody>
</table>

### 6.2.2 Principal component analysis

Principal component analysis (PCA) is based on a linear projection of multidimensional data onto different coordinates based on maximum variance and minimum correlation. As a result, less significant components can be eliminated, reducing the data representation to only those responsible for the most significant contribution. PCA was used for two purposes in this study. The first goal was for feature extraction. The data from selected sensors after optimization were compressed by PCA and only the first five principal components, which represent 99.5% of the variances, were used for classification. The second purpose of using PCA is for data clustering.
analysis. The first three principal components were used to draw the 3-D PCA plot, which enables visual observation of the closeness of two classes.

6.2.3 PNN classification model

A probabilistic neural network (PNN) was developed as a classifier for apple classification and to compute the objective function for evolutionary algorithms. The PNN consists of three layers as shown in Figure 6.1. The first layer calculates distances from the testing input vectors to the training input vectors, and generates a vector measuring the distance between testing input and a training input. The second layer adds these contributions for each class of inputs and produces a vector of probabilities. Finally, a “compete” transfer function on the output of the second layer picks the maximum of these probabilities, and produces a 1 for “healthy apple” and a 0 for “damaged apple”. Previous experience proved that better classification accuracy was normally achieved if selected Enose sensor data were compressed by principal component analysis and the first five PCs (only five contributed more than 0.5% of total variances) were chosen before the data were fed into the classifier. Another reason to compress the data and use the first five principal components is that the PNN structure can be kept uniform, instead of being changed due to the different number of sensors selected in each run. The input neurons in the input layer was set to five for the first five principal components, and the number of neurons in the hidden layer was set to the size of the training data set, which was 342. Neurons in the output layer were set to two since there are two classes of interest. The total of 456 data vectors were divided into a training set (342 vectors) and a testing set (114 vectors). The classification error rate was obtained when the testing set
was processed by the trained PNN model. Since the goal of this study was to correctly separate damaged apples from healthy apples, the testing set classification error rate (Equation 6.2) was regarded as the "cost", which was to be minimized by evolutionary algorithms described in the following sections.

\[
\text{cost} = \frac{\text{misclassified}}{114}
\]  

(6.2)

Figure 6.1. PNN network architecture (IW is input weight; || dist || is a vector whose elements indicate how close the input is to the vectors of the training set)

**6.2.4 Sequential forward/backward search**

Sequential forward search (SFS) and sequential backward search (SBS) are two similar but reverse-process approaches to search the optimal settings by sequentially adding (forward) or removing (backward) variables. SFS starts from an empty space, and at each step the variable that provides the best improvement of the objective function is added to the feature space. This process is repeated until all sensors are selected. SBS works in reverse, which begins with the whole feature space and at each step the sensor that provides the least reduction of the objective function is removed from the selection. This process is repeated until all sensors are removed from the array.
These approaches are effective for reducing sensor numbers; however, they only explore a fraction of the whole search space since the order of removing or adding sensors also influences the search results.

### 6.2.5 Statistics method: F-value selection

For multivariate data, the F-value is defined as:

$$
F = \left(1 - \frac{\Lambda_{1/b}}{\Lambda_{1/b}}\right) \left(\frac{ab - c}{p(g - 1)}\right)
$$

(6.3)

where

$$
\Lambda = \frac{\left|E\right|}{\left|H + E\right|}
$$

(6.4)

$$
E = \sum_{i=1}^{g} \sum_{j=1}^{n_i} (Y_{ijk} - \bar{Y}_{ijk})(Y_{ijl} - \bar{Y}_{ijl})
$$

(6.5)

$$
H = \sum_{i=1}^{g} n_i (\bar{Y}_{ijk} - \bar{Y}_{ij})(\bar{Y}_{ijl} - \bar{Y}_{ijl})
$$

(6.6)

a, b and c are parameters related to N (sample size), g (group number) and p (number of sensors);

$Y_{ijk}$= Observation for sensor k from subject j in group i (1 or 2 represents good or bad);

$Y_{ij} = \begin{pmatrix} Y_{g1} \\ Y_{g2} \\ \vdots \\ Y_{gp} \end{pmatrix} =$ vector of variables (p=32) for subject j in group i;

$n_i =$ number of subjects in group i (g=2);
\[
\bar{y}_{i*} = \frac{1}{n_i} \sum_{j=1}^{n_i} y_{ij} = \begin{pmatrix}
\bar{y}_{i*1} \\
\bar{y}_{i*2} \\
\vdots \\
\bar{y}_{i*p}
\end{pmatrix}
\]

= sample mean vector for group \(i\) (good or damaged group);

\[
\bar{y}_{**} = \frac{1}{N} \sum_{i=1}^{k} \sum_{j=1}^{n_i} y_{ij} = \begin{pmatrix}
\bar{y}_{**1} \\
\bar{y}_{**2} \\
\vdots \\
\bar{y}_{**p}
\end{pmatrix}
\]

= grand mean vector for sensors 1 to \(p\) (\(p=32\)).

E is the error sum of squares and cross products; for \(k=l\), this measures the within-treatment variation for the \(k\)th variable, while for \(k \neq l\), this measures the dependence between variable \(k\) and \(l\) after taking into account the treatment. H is the hypothesis sum of squares and cross products; for \(k=l\), this measures the between-treatment variation for the \(k\)th variable, while for \(k \neq l\), this measures the dependence between variable \(k\) and \(l\) across the treatments. If \(H\) is large relative to \(E\), it means variation induced by treatment is greater than that from error.

### 6.2.6 Genetic algorithms (GAs)

The genetic algorithm (GA) is a randomized search algorithm. It is inspired by the process of natural selection and performs a global random search on a population of solutions. The GA allows a population composed of many chromosomes to evolve to reach a point where the “fitness” is maximized (or the cost function is minimized). Selection, crossover, and mutation are three main operators in genetic algorithms. The selection gives the chromosome with the lowest cost the greatest probability of mating, to
bias the population to converge towards the best solutions. Mating is implemented by
swapping and combining “parents” genes to create offspring. The crossover and selection
operators make the GA globally search promising regions of a problem space. Mutation
randomly alters a certain percentage of the genes that compose population members.
Selection and mutation allow GA to locally explore a cost surface.

A binary coded GA and an integer coded GA were developed for this sensor
selection problem. For binary coded GA, each chromosome consists of 32 variables to
represent 32 sensors and each variable was either 1 (this sensor was selected) or 0 (this
sensor was not selected). For integer coded GA, each chromosome consists of an integer
number with pre-defined length, e.g., 12-gene long chromosome means only 12 repeated
variables will be selected from the 32 total variables and each gene is an integer number
ranging from 1 to 32. Non-repeated variables were first generated, and during the
evolution each of the 12 variables was compared. The repeated ones were replaced by
randomly generated numbers ranging from 1 to 32 until all 12 variables were non-
repeated.

The initial population provides a beginning search space. The ideal initial
population should be evenly distributed in the whole search space and as representative
as possible. The following mating and mutation would be based on this initial population.
This population needs to be large enough to achieve good representation, but not too
large which would result in unnecessary computation cost. The size of the initial
population was set as 16 and each chromosome was randomly generated. Larger
population sizes were tried and did not give better searching results. Elitism, which is the
principle of “survival of the fittest”, was used in the GA. In each generation, the cost and
associated chromosomes are ranked from lowest to highest cost. Only the best (lowest cost) are selected to continue for mating and mutation while the rest were discarded. The natural selection rate and mutation rate were determined by trial and error and set to 0.5 and 0.1 respectively. The “roulette wheel” selection approach was used for mating pair selection. It chooses two chromosomes to mate based on their fitness; the lowest cost has the greatest probability of mating. Single point crossover was performed during evolution. Random mutation changes certain bits in the population of chromosomes. This technique can introduce traits not in the original population and keeps the GA from converging too fast or being trapped in a local minimum before sampling the entire space. The stopping criterion was when the maximum generation reaches 200. The PNN model was used to calculate the objective function and the classification error rate was measured as the cost which was expected to be minimized.

6.2.7 CMA evolutionary strategy

As a state-of-the-art version of the evolutionary strategy, the covariance matrix adaptation evolutionary strategy (CMAES) is a heuristic optimization algorithm. The initial population is generated by sampling a normal distribution with a user specified mean value and standard deviation for each decision variable. Offspring generation, selection and recombination, covariance matrix adaptation, and step size control are four key operators in the process of evolution and are introduced below (Hansen, 2005). CMAES can be described as a randomized black box search whose computation flowchart is shown in Figure 2.
6.2.7.1 Offspring generation

A new population is sampled from a normal distribution defined by:

\[ x^{(g+1)}_k \sim N(m^{(g)}_k, (\sigma^{(g)}_k)^2 C^{(g)}) \]  

\( k=1, \ldots, \lambda \)
Where
\( \lambda \) is the population size; \( x^{(g+1)}_k \) is the \( k \)th sampled individual of generation \((g+1)\);
\( N(m^{(g)}, \sigma^{(g)} C^{(g)}) \) is a multivariate normal distribution in generation \((g)\); \( m^{(g)} \in R^n \), is the mean value of decision variables in generation \((g)\); \( \sigma^{(g)} \in R_+ \), is the “overall” standard deviation or step size at generation \(g\); \( C^{(g)} \in R^{n \times n} \), is the covariance matrix at generation \(g\).

### 6.2.7.2 Selection and recombination

Selection and recombination are used to determine the new mean value of the distribution for generation \((g+1)\), which is a weighted average of \( \mu \) selected individuals from the population.

\[
m^{(g+1)} = \sum_{i=1}^{\mu} w_i x^{(g+1)}_{i, \lambda} \quad (6.8)
\]

\[
\sum_{i=1}^{\mu} w_i = 1, w_1 \geq w_2 \geq \ldots w_\mu \quad (6.9)
\]

Where
\( x^{(g+1)}_{i, \lambda} \) is the \( i \)th best individual in generation \((g+1)\). The first \( \mu \) individuals in the fitness ranking are chosen for recombination. The \( \mu \) variable is also called the parent number. Different weights (\( w_{i=1,\ldots,\mu} \in R_+ \)) can be assigned to each selected individual to enhance selective pressure.

\[
\mu_{eff} = \left( \sum_{i=1}^{\mu} w_i^2 \right)^{-1} \quad (6.10)
\]
\( \mu_{\text{eff}} \) is defined as variance effective selection mass and it is determined by the recombination weights \( w_i \). The value of \( \mu_{\text{eff}} \) is in the range of \([1, \mu]\). When the population size is given, smaller \( \mu_{\text{eff}} \) means higher selective pressure.

### 6.2.7.3 Covariance matrix adaptation

The covariance matrix defines the shape of the variable distribution and it consists of three parts, calculated by:

\[
p_c^{(g+1)} = (1 - c_c) p_c^{(g)} + h_\sigma^{(g+1)} \frac{c_c (2 - c_c) \mu_{\text{eff}} m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}
\]

\[
C^{(g+1)} = (1 - c_{\text{cov}}) C^{(g)} + \frac{c_{\text{cov}}}{\mu_{\text{cov}}} (p_c^{(g+1)} p_c^{(g+1)^T} + \delta(h_\sigma^{(g+1)}) C^{(g)}) + c_{\text{cov}} (1 - \frac{1}{\mu_{\text{cov}}}) \sum_{i=1}^{\mu} \frac{w_i}{(\sigma^{(g)})^2} (x_i^{(g+1)} - m^{(g)}) (x_i^{(g+1)} - m^{(g)})^T
\]

where

- \( p_c^{(g+1)} \) is the evolution path which is used to record correlations between consecutive evolution steps;
- \( c_{\text{cov}} \) is the learning rate for covariance matrix updates;
- \( c_c \) is the learning rate for evolution path updates;
- \( \mu_{\text{cov}} \) is the weighting between the evolution path update and estimator of distribution update.
6.2.7.4 Step size control

The step size $\sigma^{(g)}$ of each generation is updated by using a cumulative path length control method. A conjugated evolution path is constructed according to the following equations:

$$p^{(g+1)}_\sigma = (1 - c_\sigma)p^{(g)}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)}\mu_{\text{eff}} C^{(g)} \frac{1}{2} m^{(g+1)} - m^{(g)}$$

(13)

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma \left\| p^{(g+1)}_\sigma \right\|}{d_\sigma N(0, I)} - 1\right)$$

(14)

Where

$c_\sigma$ is the learning rate for cumulative step size control; $d_\sigma$ is damping parameter for step size update.

6.2.7.5 Default values and stop criteria

Algorithm parameters are always important for search efficiency, global optimization quality, and algorithm reliability. In CMAES, the following parameters were used: $\lambda, \mu, w_i, c_{\text{cov}}, c_c, \mu_{\text{cov}}, c_\sigma$ and $d_\sigma$. Among them, population size $\lambda$, parent number $\mu$ and recombination weight $w_i$ are given in the algorithm (Hansen, 2005). The default values are defined in the following equations and other parameters can be derived using these three parameters:

$$\lambda = 4 + [3 \ln n]$$

(6.15)

$$\mu = \lfloor \frac{\lambda}{2} \rfloor$$

(6.16)
\[ w_i = \frac{\ln(\mu + 1) - \ln i}{\mu \ln(\mu + 1) - \sum_{j=1}^{\mu} \ln j} \quad \text{for } i = 1, \ldots, \mu \] (6.17)

Where

\[ n \] is the number of decision variables.

The stop criteria for the Enose sensor selection application is when the maximum standard deviation of the decision variables is smaller than 0.25.

### 6.3 Computational experiment

Genetic algorithms with two different coding methods, binary and integer, were developed and compared for sensor selection. The binary GA has a straightforward representation for this problem because each sensor was represented by a binary number with 1 representing “to select” and 0 representing “to remove”. There were a total of 32 variables corresponding to the number of sensors. This coding method has an obvious advantage of easily accomplished crossover and mutation operators, but the number of sensors selected from this coding method was different in each independent run. The integer coded GA is a special type of real number coded GA with each variable being an integer number representing the number of the selected sensor. This approach has the advantage of controlling the number of sensors selected by defining the length of the chromosome \textit{a priori}. In this study, the length of the chromosome of integer coded GA was set to 12. The probabilistic neural network (PNN) was used as classifier for both approaches. The adjusted algorithm parameters found to be most effective are shown in Table 6.2. Since GA is a random search method, different sensors were selected at each
iteration with different classification error rates. To evaluate the system more objectively, 30 random seed runs were performed, and the average and best performance results were evaluated.

<table>
<thead>
<tr>
<th>Table 6.2. Binary and integer coded GA parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Binary coded GA</strong></td>
</tr>
<tr>
<td>Population size</td>
</tr>
<tr>
<td>Mutation rate</td>
</tr>
<tr>
<td>Length of chromosome</td>
</tr>
<tr>
<td>Maximum generation</td>
</tr>
<tr>
<td>Selection method</td>
</tr>
<tr>
<td>Selection rate</td>
</tr>
</tbody>
</table>

CMAES is designed to solve continuous number problems. However, by making some adjustments, this algorithm can also be adapted to solve binary coded problems. For binary coded CMAES, the initial population was randomly generated as real numbers in the range of 0 to 1, and the population was manipulated by the evolution strategy operators thereafter. In calculating fitness values, the variable was forced to be 1 if the variable value is greater than 0.5, which means this sensor was selected; otherwise, it was forced to be 0 which means this sensor was not selected. The default algorithm parameters which were calculated using equations stated in section 6.2.7 are listed in Table 6.3.

To fully exploit the advantage of CMAES for solving continuous number problems, a real number coded scheme was also developed. Each variable was divided into 4 segments, and each segment represented one of four possibilities of two sensor selections as defined in Equation 6.18. For instance, if x falls into range [0.25, 0.5), it represents the state of [1,0]: the first sensor was selected and the second was not selected.
By doing this, each variable can represent four sensor selection possibilities of two sensors and hence the 32 sensors can be represented by 16 variables, which reduced the variable number by 50% compared to the binary coded scheme. The search was terminated when the maximum standard deviation of the decision variables was smaller than 0.25.

Table 6.3. CMAES algorithm parameters

<table>
<thead>
<tr>
<th></th>
<th>Binary coded</th>
<th>Real number coded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of sensors</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Number of variables</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>Population size</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>Parent number</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>Recombination weights</td>
<td>[0.3448 0.2299 0.1626 0.1149 0.0779 0.0477 0.0221]</td>
<td>[0.3818 0.2458 0.1663 0.1098 0.0660 0.0303]</td>
</tr>
</tbody>
</table>

6.4 Results and discussion

6.4.1 PCA plot

The 3-D PCA plot of individual day data and pooled data is presented in Figure 6.3. Day 9 data was selected as a representative individual day, and data collected in September from day 5 to day 13 as pooled data. On the individual day, healthy apples and damaged apples formed two obvious groups and can be easily separated; however, for the pooled data, the two classes overlapped and no clear plane separates these two classes.
6.4.2 SFS/SBS search result

The sequential forward search (SFS) process is shown in Figure 6.4. It was observed that the classification accuracy reached 90.4%, when the first five sensors (sensor 5, 6, 27, 21, and 26) were added; the classification accuracy increased to the maximum of 93.9% when the four additional sensors were added (sensor 10, 2, 12, and 17). With any more sensors, performance did not improve. Interestingly, certain sensors deteriorated the performance of the classifier, such as sensors 31, 23, 24.

The sequential backward search (SBS) produced a similar pattern as shown in Figure 6.4. System performance was continuously improved after sequentially removing sensors 31, 7, 1, 22 and 11. System performance reached the peak when only eight
sensors were left (6, 5, 14, 27, 24, 26, 30, and 28) with a classification accuracy of 93.9%. Continuously removing sensors from these eight decreased the classification accuracy from 93.9% to 78%. By using SFS, the best classification system performance was found to be 93.9% while the minimum sensors used were nine. SBS found that by using only eight sensors, the classification accuracy of the system could reach a peak of 94%.

Figure 6.4. The SFS (top) and SBS (bottom) search process for the Enose sensor selection

6.4.3 F-value selection

The F-value is an index to measure the importance of one variable in multivariate analysis of variance (MANOVA). Selecting sensors with large F-values will result in lower classification error rates. The gml procedure from SAS (SAS Inc., NC) was used to calculate the F-value for each of the 32 sensors (Figure 6.5). The 32 sensors were ranked
by their F-value and their cumulative F-value calculated. Using the top 90% F-values, 25 sensors were correspondingly selected by this procedure. The removed sensors were 6, 10, 17, 22, 23, 26, and 31. By using the data from the selected 25 sensors, the classification accuracy of the system was 74.6%.

![Figure 6.5. F-values for 32 sensors](image)

**Figure 6.5. F-values for 32 sensors**

### 6.4.4 Sensor selection by genetic algorithms (GAs)

As shown in Table 6.4, both binary coded and integer coded GA resulted in a minimum classification error rate of 0.044. The average performance of binary coded GA was slightly better than integer coded GA with the average cost, or classification error rate, of 0.052 versus 0.059.
Table 6.4. Comparison of binary and integer coded GA over 30 random seed runs

<table>
<thead>
<tr>
<th>Classification error rate</th>
<th>Binary coded</th>
<th>Integer coded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.052</td>
<td>0.059</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0.0039</td>
<td>0.0069</td>
</tr>
<tr>
<td>Best results searched</td>
<td>0.044</td>
<td>0.044</td>
</tr>
<tr>
<td>Number of function evaluations</td>
<td>3200</td>
<td>3000</td>
</tr>
</tbody>
</table>

The typical binary coded and integer coded GA convergence plots with the best fitness value and average fitness value in each generation are shown in Figure 6.6. Because the elitism was introduced in the genetic algorithm, the best fitness value decreased monotonically in the plot; the population average decreased sharply within the first few generations, but fluctuated without much decrease as the evolution progressed. It was also observed that both binary coded GA and integer coded GA converged quickly. They reached the minimum error rate within 15 generations, and the performance was not improved in the rest of the generations.

Figure 6.6. Binary coded GA (a) and integer coded GA (b) optimization convergence plots
6.4.5 CMAES sensor selection

6.4.5.1 Comparison of binary and real number coded CMAES

The best fitness value of each generation versus number of function evaluations for binary and real number coded CMAES are shown in Figure 6.7. Because the elitism was not used in the CMAES algorithm, the best fitness value in one generation was not necessarily better than the previous generation. Instead of decreasing monotonically, which was the case in GA, the best fitness value for the CMAES fluctuated during evolution. Sometimes, the best fitness value in all generations was not in the final generation because of this fluctuation.

As shown in Figure 6.7, the best fitness value of 0.044 was reached after 826 function evaluations for binary coded, and the same error rate was found by real number coded after 962 function evaluations. The best fitness value in the first generation by random generation was 0.097 and it decreased by 5.3% because of the CMAES optimization search. Corresponding to the best fitness value, 12 sensors were selected by the binary coded method: 2, 5, 6, 7, 12, 13, 19, 22, 25, 26, 27, and 32, whereas 18 sensors were selected by the real number coded method: 2, 3, 5, 6, 7, 9, 12, 13, 14, 16, 17, 20, 23, 24, 26, 27, 28, and 30.
Figure 6.7. Evolution history comparison between binary coded and real number coded

The 20 random seed runs were conducted and the average performances were compared (Table 6.5). Binary coded CMAES generated better results than the real number coded CMAES, although they both found the best result of 0.044 error rate at least once in 20 random seed runs. The binary coded CMAES performed a little better than the real number coded CMAES because using one variable to represent each sensor allows a more effective search than using each variable to represent two or more sensors. Subsequently, binary coded CMAES was further explored.

Table 6.5. Binary and real number coded CMAES searching results after 20 random seed runs

<table>
<thead>
<tr>
<th></th>
<th>Binary coded</th>
<th>Real number coded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average classification error rate</td>
<td>0.0517</td>
<td>0.0589</td>
</tr>
<tr>
<td>S.D of classification error rate</td>
<td>0.007</td>
<td>0.008</td>
</tr>
<tr>
<td>Average number of function evaluations</td>
<td>1569</td>
<td>815</td>
</tr>
<tr>
<td>Best result</td>
<td>0.044</td>
<td>0.044</td>
</tr>
</tbody>
</table>
6.4.5.2 CMAES parametric study

Although the default parameters gave a relative satisfactory performance, changing the population size $\lambda$ was recommended by Hansen (Hansen, 2005). In this section, different population sizes were tested to investigate the influence of population size on the final search result. Population sizes tested here were: $\lambda=9, 14, 24$ and $27$. The corresponding parent number was determined by the equation: $\mu = \lambda / 2$. Each population size was run ten times and the evolution history of average performance for each population size is shown in Figure 6.8. The number of function evaluations and error rate for each population size are listed in Table 6.6.

Different population sizes led to different optimized error rates. Generally speaking, increasing population size improves the search performance by increasing the chance of finding the minimum error rate. In this case, the optimized error rate was decreased from 5.4% when the population size is 9, to 4.7% when the population size is 27. However, this improvement was at the cost of using more function evaluations. The average number of function evaluations was increased from 1487 for a population size of 9 to 3261 for a population size of 27. A larger population size ($\lambda = 27$) achieved a better average search result than the default population size ($\lambda = 14$). Although it takes more nfe's, consideration of the computation time in this problem is not paramount and the better optimization result is more desirable than saving computation time. Therefore, the larger population size ($\lambda = 27$) was chosen.
Figure 6.8. Evolution history of different population sizes

Table 6.6. Comparison of different population size ($\lambda$) performance

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>Average number of function evaluations ($nfe$)</th>
<th>Standard deviation of nfe</th>
<th>Average error rate</th>
<th>Standard deviation of error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1487</td>
<td>497.4</td>
<td>0.054</td>
<td>0.008723</td>
</tr>
<tr>
<td>14</td>
<td>1569</td>
<td>825.4</td>
<td>0.050</td>
<td>0.006472</td>
</tr>
<tr>
<td>24</td>
<td>2575</td>
<td>1481.6</td>
<td>0.050</td>
<td>0.005921</td>
</tr>
<tr>
<td>27</td>
<td>3261</td>
<td>6494.5</td>
<td>0.047</td>
<td>0.004237</td>
</tr>
</tbody>
</table>

6.4.5.3 CMAES algorithm ($\lambda = 27$) reliability analysis

Since a population size of 27 generated the best result, it was chosen for reliability analysis by running 30 different random number seeds. Due to the inconsistency of the best fitness value in the last generation and in all generations, the best classification error rate and the minimum classification error rate in the last generation are compared in Table 6.7. The average value and standard deviation of the function evaluations were also recorded. In 30 random number seed runs, there are only two cases where the best minimum error rate is different from the minimum error rate in the last generation, so the
performance of the two categories is very close. It was observed that the best minimum classification error rate was 4.83% and the standard deviation was 0.0045, which was small. In general, CMAES with a population size of 27, using other default values, lead to reliable and consistent performance.

Table 6.7. Reliability analysis of $\lambda=27$ (30 random number seeds)

<table>
<thead>
<tr>
<th>30 random number seeds</th>
<th>Last generation result</th>
<th>Best result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Classification error rate</td>
<td>Function evaluations</td>
</tr>
<tr>
<td>Mean</td>
<td>0.0503</td>
<td>1338.4</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0.0065</td>
<td>696.8</td>
</tr>
</tbody>
</table>

In Table 6.8, ten representative sensor selection schemes from a total of 30 different random number seed runs ($\lambda = 27$) are recorded. Although independent runs of CMAES with a population size of 27 achieved relatively similar minimum classification error rates (from 0.044 to 0.070), no two selections had the same sensors even for the same classification error rate. This result proved that the 32 Enose internal sensors are correlated with each other, and the same classification error rate could be achieved by using different sensors.

Another observation from this table is that not all runs searched the minimum classification error rate of 0.044 (the best fitness value found in all runs). Some searches were terminated because of premature convergence before reaching the minimum classification error rate. For example, in number 3 and number 8 runs, both error rates were 0.044 and ten sensors were selected, but among these ten sensors, only eight of them were the same and two were different. In number 3 run, sensor combinations of 16, 29 and 5, 6, 7, 12, 19, 26, 27, and 32 achieved the same classification error rate as the sensor combinations of 2, 22 and 5, 6, 7, 12, 19, 26, 27, and 32 in number 8 run. If the
best sensor selection is defined as the minimum error rate with the least sensors selected, number 16 run provided the best selection. Only seven sensors were selected, 5, 6, 22, 25, 26, 27 and 32, while the minimum classification error rate of 0.044 was achieved.

Table 6.8. Ten representative sensor selection schemes from 30 independent runs

<table>
<thead>
<tr>
<th>Run number</th>
<th>Error rate</th>
<th>Sensor numbers</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.044</td>
<td>12</td>
<td>2 4 5 6 10 13 14 24 25 27 30 32</td>
</tr>
<tr>
<td>2</td>
<td>0.053</td>
<td>18</td>
<td>2 4 5 6 7 8 12 16 19 20 21 24 25 26 27 28 30</td>
</tr>
<tr>
<td>3</td>
<td>0.044</td>
<td>10</td>
<td>5 6 7 12 16 19 26 27 29 32</td>
</tr>
<tr>
<td>4</td>
<td>0.053</td>
<td>14</td>
<td>2 3 5 6 9 12 14 19 20 24 26 27 28 30</td>
</tr>
<tr>
<td>5</td>
<td>0.044</td>
<td>7</td>
<td>5 6 22 25 26 27 32</td>
</tr>
<tr>
<td>6</td>
<td>0.053</td>
<td>15</td>
<td>2 3 5 6 7 12 14 19 20 24 26 27 28 29 30</td>
</tr>
<tr>
<td>7</td>
<td>0.070</td>
<td>19</td>
<td>2 4 5 6 8 9 12 13 17 19 20 22 24 26 27 28 30 32</td>
</tr>
<tr>
<td>8</td>
<td>0.044</td>
<td>10</td>
<td>2 5 6 7 12 19 22 26 27 32</td>
</tr>
<tr>
<td>9</td>
<td>0.062</td>
<td>16</td>
<td>2 5 6 7 8 10 13 14 19 22 24 26 27 29 30 32</td>
</tr>
<tr>
<td>10</td>
<td>0.053</td>
<td>17</td>
<td>2 4 5 6 8 13 14 16 18 20 23 24 25 27 28 30 32</td>
</tr>
</tbody>
</table>

Although sensors selected in all 30 independent runs were different, some sensors were always selected or more often selected than others. In order to observe this pattern, a sensor selection distribution histogram is shown in Figure 6.9. Sensors 5 and 6 were selected all 30 times, and eight sensors were selected more than 20 times in the 30 runs: 2, 5, 6, 14, 24, 27, 30, and 32. This indicates these eight sensors are very important in classification of damaged apples. One interesting result is that in contrast to the Enose users manual (Cyrano Science Inc., 2000) which suggested that sensor 5 and 6 should be removed from input data due to their sensitivity to polar vapor, these two sensors contributed significantly to damaged apple classification and should be included in each classification.
6.4.6 Optimal solution sets

If the classification error rate and number of sensors selected are two conflicting objectives to be minimized, it becomes a two-objective optimization problem. As few sensors as possible are used to reach as low a classification error rate as possible. The relationship between the number of sensors selected and the classification error rate can be obtained by calculating classification error rate using different number of sensors from 1 to 32. Integer coded GA was applied due to its ability to optimize classification error rates by using a predefined number of sensors. Figure 6.10 shows an interesting characteristic of this problem: The system classification error rate did not decrease monotonically as the number of sensors increased. Instead, including all 32 sensors deteriorated the system performance. The classification error rate was 14%, even worse than using only three sensors (classification error rate of 12.3%). In fact, the system
performance was relative satisfactory using sensor numbers ranging from 4 to 31. Increasing the number of sensors did not improve the classification performance, and the classification error rate fluctuated from 0.044 to 0.070 when the number of sensors increased from 8 to 31. If the goal is to achieve the minimum classification error rate by using as few sensors as possible, using seven sensors (sensor 5, 6, 22, 25, 26, 27, and 32) can reach the minimum classification error rate of 4.4% with the sensors' dimensionality reduced by 75%.

![Figure 6.10. Relationship between number of sensors and the classification error rate](image)

6.5 Conclusion

The optimization of the Cyranose 320 electronic nose to classify damaged apples from healthy apples was studied in this research. The state-of-the-art CMA evolutionary strategy was applied and compared with genetic algorithm optimization, F-value selection and SBS/SFS selections.
Both CMAES and genetic algorithms found the minimum classification error rate of 0.044, which was better than the results from the F-value selection (0.254) and SBS/SFS selection (0.061). Considering the average performance over 30 random seed runs, CMAES achieved more satisfactory performance than genetic algorithms in optimization quality, with an average classification error rate of 0.0497 versus 0.059. CMAES was further fine-tuned by adjusting its default population size. It was observed that a better search quality was obtained by increasing the population size from a default value of 14 to 27. The average classification error rate was improved from 0.054 to 0.047 over 10 random seeds runs. The CMAES algorithm with a population size of 27 also showed good reliability. Lastly, optimal solutions were obtained to illustrate the tradeoff between the classification error rate and the number of sensors selected. More sensors did not necessarily result in improved classification performance. Using four sensors or 31 sensors did not generate much different results. The minimum classification error rate could be found by using only eight sensors.

This research explored different approaches for electronic nose sensor selection and system optimization for apple defect detection. The best system classification performance was achieved when the data dimensionality was reduced by 75%. In the future, by using this optimization method, electronic nose manufacturers could assemble only those relevant sensors for specific applications, which will not only reduce sensor cost, but minimize computation time and improve the classification accuracy as well.
CHAPTER 7

A COMPARATIVE STUDY OF THREE EVOLUTIONARY ALGORITHMS FOR SURFACE ACOUSTIC WAVE SENSOR WAVELENGTH SELECTION

It is common sense to take a method and try it. If it fails, admit it frankly and try another. But above all, try something.

Franklin D. Roosevelt (1882-1945)

ABSTRACT

A surface acoustic wave sensor (the zNose™) was utilized to detect fruit defects by measuring and analyzing the volatile compounds emitted by apples. The zNose™ generates a spectrum with 512 wavelength values. This large number of variables not only increases the processing time, but reduces the classification accuracy due to irrelevant information and noise. In this study, three evolutionary techniques: genetic algorithms (GA), covariance matrix adaptation evolutionary strategy (CMAES), and differential evolution (DE) algorithms, were investigated to select the most relevant wavelengths and reduce data dimensionality of a surface acoustic wave sensor for apple defect detection. Three algorithms were compared for their search quality, search efficiency, and data dimensionality reduction. The whole spectrum, which spans 512 wavelength values, was divided into different windows: 16, 32, and 64. These three different discretization schemes were tested by the three techniques. Both CMAES and DE yielded the best prediction accuracy with the 64 windows scenario, and GA produced
comparable results with 32 windows and 64 windows, which were better than 16 windows. These results suggested that the finer the spectrum was discretized, the better the classification accuracy obtained. The results also showed that CMAES is the most efficient search algorithm with comparable search quality as DE. Three algorithms were further fine-tuned by adjusting their population size which influences the search space. The parametric study was conducted only for the 64-window case. It was observed that algorithms with larger population size gave better search results. For CMAES, the average cost for ten random seed runs was 0.0289 with the best search cost of 0.0263 for population $2\lambda$. Differential evolution (DE) produced slightly better search results but at the cost of reducing search efficiency. All three algorithms can effectively reduce data dimensionality by 50%, which in turn reduces the computation time.

7.1 Introduction

Fruit production in the US accounts for $70 billion each year (USDA-NASS, 2006). However, fruits are easily perishable commodities and fresh produce departments typically experience more than 10% loss of revenue due to fruit quality deterioration (NE-179 Project report, 2001). Even worse, defects on the surface of fruits can create sites for pathogens to develop and consequently pose risks for human health. It is estimated that there were 428 cases of foodborne illnesses induced by contaminated fresh fruits in 2003 in the US and fresh fruits have become the second largest vehicle for foodborne diseases (Centers for Disease Control and Prevention, 1998). In this research, a surface acoustic wave sensor, the zNose™ (Electronic Sensor Technology, 2001), was
utilized to measure the headspace gas from apples and detect physically damaged apples by using pattern recognition algorithms.

A surface acoustic wave (SAW) sensor generates spectral data which span 512 wavelength values during 10 seconds sampling time. Conventionally, spectral data were analyzed by multivariate calibration techniques such as principal component analysis (PCA) and partial least squares (PLS) (Roger and Bellon-Maurel, 2000). Chemists and spectroscopists extract information by using PCA or PLS to predict solution concentrations or properties of samples (Broadhurst et al., 1997). Recent research (Goicoechea and Olivieri, 2002) indicated that the prediction results may be improved by selecting only the relevant wavelengths. During the sampling process, there are some factors that may add irrelevant information to spectral data, such as environmental noise, temperature changes, and other interferences. These factors are not useful for explaining the variations in the given properties of the object of interest, and the presence of this information only reduces the prediction accuracy.

Since a SAW sensor generates a spectrum every 0.02 seconds during the 10-second cycle, a large number of variables (512) are created. An exhaustive search of all combinations from these variables is too computationally expensive (it is an astronomical number, more than hundreds of billions) to undertake. Researchers developed algorithms to select relevant variables, such as forward stepwise regression and correlellogram analysis (Osborne and Fearn, 1986). More recently, researchers explored heuristic optimization methods, such as genetic algorithms (Broadhurst et al., 1997; Goicoechea and Olivieri, 2002; Goldberg, 1989; Lucasius et al., 1994; Roger and Bellon-Maurel, 2000) and simulated annealing (Ingber, 1993). Genetic algorithms were proven to be an
effective heuristic searching method for variable selection and classification performance enhancement.

In this study, two state-of-the-art algorithms, covariance matrix adaptation evolutionary strategy (CMAES) (Hansen, 2005) and differential evolution (DE) (Storn and Price, 1995) were applied for zNose™ wavelength selection and compared to genetic algorithms (GA). CMAES is capable of dealing with problems of high non-linear, concave, and rugged search landscapes (Hansen, 2005). DE is a newly developed heuristic approach for minimizing possible nonlinear and non differentiable continuous space functions (Storn and Price, 1995). It has been proven that the DE converges faster than Adaptive Simulated Annealing (Ingber, 1993). Moreover, this method requires few control variables and is more robust (Storn and Price, 1995).

In the following sections, three algorithms (GA, CMAES and DE) were compared for their searching quality (the minimum cost), searching efficiency (number of function evaluations) and data dimensionality reduction (number of wavelengths selected). Three algorithms were further fine-tuned by adjusting their population size, which has a large influence on searching space. A recommendation of choices of algorithms and zNose™ spectral data dimensionality reduction are given.

7.2 Materials and Methods

7.2.1 Apple volatile sample collection

Red ‘Delicious' apples were purchased from a local grocery store. Apples were intentionally damaged by inducing a 10 mm deep cross-slice cut at the apple surface. The damaged apples were exposed to room air for deterioration development. The
measurements were conducted every other day from day 4 to day 14 after the cut treatment. Other apples without the cut treatment were considered as “healthy” apples.

Each individual apple was stored in a 2 L glass jar for accumulation of headspace gas, which was sealed by a plastic cap with a Teflon septum. A 5 mm diameter hole was drilled at the center of the plastic cap. The zNose™ (Electronic sensor technology, Newsberry, CA) sampling needle was inserted from this opening to collect apple headspace gas, and this opening was sealed by an inert silicon stopper after sampling.

Apple headspace gas samples were collected during three different times in 2005: March, June, and September. Each time, both healthy apples and damaged apples were monitored from the earliest time of 4 days after treatment until the latest time of 14 days after treatment. Sampling time and number of samples for each group are shown in Table 7.1. A total of 456 data vectors were used for pattern recognition algorithm development. A typical zNose™ smellprint for both healthy apples and damaged apples is shown in Figure 7.1.

<table>
<thead>
<tr>
<th>Period of sampling</th>
<th>Days of sampling</th>
<th>No. of replications</th>
<th>No. of samples</th>
<th>Data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>March, 2005</td>
<td>5, 7, 9, 11, 14</td>
<td>24</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>June, 2005</td>
<td>4, 6, 8, 10</td>
<td>24</td>
<td>96</td>
<td>2</td>
</tr>
<tr>
<td>September, 2005</td>
<td>5, 7, 9, 11, 13</td>
<td>48</td>
<td>240</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>456</td>
<td>Pooled</td>
</tr>
</tbody>
</table>
7.2.2 Test case development

One difficulty for wavelength selection is that there are 512 values in each spectrum. If each wavelength value is treated as one variable, the 512 gene-long chromosome is too long for any evolutionary algorithms to compute. One solution to this difficulty is to discretize the whole spectrum into a number of windows or bands. Each window was treated as one variable, so the number of variables was reduced to the number of windows. In Figure 7.2, the zNose™ spectrum was discretized into 16 windows, and hence there are only 16 variables to select, which greatly reduces the length of the chromosome. After evolutionary algorithm selection, the wavelength values within selected windows were kept unchanged instead of being averaged, and the reduced wavelengths were then fed into classification models.
In this study, three different discretizing schemes were compared. The whole spectrum was divided into 16, 32 and 64 windows with the increasing resolution of 32, 16 and 8 wavelength values in each window. Since the goal was to classify damaged apples from healthy apples, the classification error rate (Equation 7.1) was treated as the cost function. Procedures were then used to minimize the cost function. The combination of variables that gave the minimal classification error rate was the desired selection.

\[
\text{Cost function} = \frac{\text{misclassifications}}{\text{testing set}}
\]  

(7.1)

7.2.3 PNN

To calculate the cost function for evolutionary algorithms, a partial least square (PLS) classification model and probabilistic neural network (PNN) model were developed by MATLAB Chemometrics and NN Toolbox (The Mathworks Inc., Natick, MA), respectively. The PNN model showed better classification accuracy than the PLS (Li and
Heinemann, 2006a), so the PNN model was used as the classifier and for computing the objective function thereafter.

The total 456 data vectors were divided into a training set and testing set to avoid the over-fitting problem that typically occurs in artificial neural network applications. One fourth of the data (114 vectors) were treated as the testing data set, and three fourths of the data (342 vectors) were treated as the training set. The PNN consisted of three layers as shown in Figure 7.3. Previous experience proved that better classification accuracy was usually achieved if selected zNose™ data were compressed by principal component analysis and the first ten principal components (which contributed more than 0.5% of total variances) were chosen before the data were fed into the classifier (Li et al., 2006). Another reason to compress the data and use the first ten principal components is that the PNN structure can be kept uniform by doing this, instead of being changed due to different number of wavelengths selected in each run. The neurons in the input layer was set to 10 for the first ten principal components obtained from selected zNose™ wavelengths, and the number of neurons in the hidden layer was set to the size of the training data set (342). Neurons in the output layer were set to two since there were two classes of interest.

![Figure 7.3. PNN network architecture](image-url)
7.3 Algorithm comparison

In this research, the performance of the genetic algorithms (GA), the covariance matrix adaptation matrix evolutionary strategy (CAMES), and the differential evolutionary (DE) were compared for their capacity to reduce data dimensionality (wavelength selection) for acoustic wave sensors. All three algorithms are heuristic optimization approaches, and they explore the whole search space stochastically. The key differences between these three algorithms are compared in the following sections.

7.3.1 Simple GA

The genetic algorithm (GA) is a randomized search algorithm. It is inspired by the process of natural selection and performs a global random search on a population of solutions. The selection, cross over and mutation are three main operators in genetic algorithms. The flowchart of the GA optimization process is shown in Figure 7.4. Based on the nature of this problem, binary coding was used for wavelength selection. Each window was represented by one binary number; 1 means this window was selected and 0 means this window was not selected. The length of each chromosome was the number of the windows.

The initial population provides a beginning search space, which is expected to be evenly distributed in the whole search space and as representative as possible. The selection, mating, and mutation operators are based on this initial population. This population should not be too small to achieve good representation and not be too large which might result in unnecessary computation cost. The size of the initial population was set to 20 and was randomly generated. Elitism, which is the principle of “survival of
the fittest”, was used in the genetic algorithm. In each generation, the cost and associated chromosomes are ranked from lowest cost to highest cost. Only the best individual with the lowest cost is selected to continue for mating and mutation while the rest were discarded. The natural selection rate was set to 0.5. The roulette wheel selection approach was used for mating pair selection. It chooses two chromosomes to mate based on their fitness; the lowest cost has the greatest probability of mating. Single point crossover was performed during evolution. Random mutation changes certain bits in the population of chromosomes. This technique can introduce traits not in the original population and keeps the GA from converging too fast or being trapped in a local minimum before sampling the entire space. The mutation rate was chosen as 0.1. The stopping criterion was when the maximum generation reaches 200.
7.3.2 CMA evolutionary strategy

CMA evolutionary strategy (CMAES) is a state-of-the-art heuristic optimization algorithm. The initial population is generated by sampling a normal distribution with a user-specified mean value and standard deviation of each decision variable. Offspring generation, selection and recombination, covariance matrix adaptation, and step size control are four key operators in the process of evolution, and the mathematic expressions
of these steps can be referred to (Hansen, 2005). CMAES can be described as a randomized black box search and the algorithm flowchart is shown in Figure 7.5.

![Algorithm Flowchart](image)

Figure 7.5. CMAES algorithm flow chart

Algorithm parameters are always important for search efficiency, global optimization quality, and algorithm reliability. In CMAES, the following parameters were used: \( \lambda, \mu, w, c_{\text{cov}}, c_{\text{c}}, \mu_{\text{cov}}, c_{\text{e}}, \text{ and } d_{\sigma} \). Among them, population size \( \lambda \), parent
number $\mu$ and recombination weight $w_i$ can be set to default values which were defined in the following equations, and other parameter can be derived by these three parameters.

$$\lambda = 4 + [3 \ln n] \quad (7.2)$$

$$\mu = [\lambda / 2] \quad (7.3)$$

$$w_i = \frac{\ln(\mu + 1) - \ln i}{\mu \ln(\mu + 1) - \sum_{j=1}^{\mu} \ln j} \text{ for } i=1,\ldots,\mu \quad (7.4)$$

The stop criteria for this zNose$^\text{TM}$ sensor selection application was when the maximum standard deviation of the decision variables was smaller than 0.25.

### 7.3.3 Differential evolution (DE)

DE is a novel parallel direct search method which was developed by Rainer Storn and Kenneth Price (Storn and Price, 1995) in the mid-nineties. It is a heuristic approach for minimizing possible nonlinear and non differentiable continuous space functions. The crucial idea behind DE is a scheme for generating trial parameter vectors. Basically, DE adds the weighted difference between two population vectors to a third vector, which is then crossed-over with the target vector $S_1$ and subsequently produces a new trial vector $u$. If the resulting trial vector $u$ has a lower objective function value than the target vector $S_1$, then it will replace $S_1$ and enter the next generation, otherwise, it will be discarded and $S_1$ will survive. The best parameter vector $X_{\text{best},G}$ is evaluated for every generation $G$ to keep track of the progress that is made during the minimization process. This process is shown in Figure 7.6.
The initial population is generated randomly assuming a uniform probability distribution for all random decisions.

For each vector \( x_{i,G} \), \( i=0,1,2,\ldots,NP-1 \), a vector \( v \) is generated according to

\[
v = X_{r1,G} + F \cdot (X_{r2,G} - X_{r3,G})
\]  

(7.5)

where

\( r_1, r_2, r_3 \) are chosen randomly from the interval \([0,NP-1]\) and are different from the target index \( i \); \( F \) is a real and constant factor which controls the amplification of the differential variation \( (X_{r2,G} - X_{r3,G}) \); \( NP \) is the population number; \( G \) is the generation number.

In this research, the MATLAB source code developed by Storn and Price (Storn and Price, 1995) was modified and tested for this specific problem.
7.3.4 Metrics of performance

To compare the three different evolutionary algorithms, three metrics of performance were evaluated: 1) mean and best values of objective functions after 10
random seed runs, which measured the searching quality; 2) number of function evaluations for searching efficiency measurement; 3) number of windows selected for data dimensionality reduction effectiveness measurement.

Since all three algorithms are heuristic optimization approaches, each independent run may produce different results depending on the initial random seeds. To better measure the robustness of each algorithm, the average performance of each algorithm over 10 independent runs were considered as the search quality metric. In addition to the average performance, the best found result is also an important index to measure the searching power of each algorithm. Searching efficiency was measured by the number of function evaluations, which was approximately the number of generations multiplied by the population size, instead of the number of generations which varies depending on the population size. The larger the number of function evaluations (nfe), the higher the CPU time consumption. Data dimensionality reduction is another metric to measure the efficacy of the searching algorithm; less data dimensionality with comparable classification error rate was desired.

7.4 Computational experiment

In this study, the surface acoustic wave sensor spectrum was divided into 16, 32 and 64 windows, corresponding to 32, 16 and 8 wavelength values in each window. Each window-dividing scheme was tested by three algorithms. The GA, CMAES and DE were first compared using their default settings which were commonly recommended from literature (Hansen, 2005; Storn and Price, 1995), summarized in Table 7.2. After selecting the optimal window-dividing scheme, the three algorithms were further
explored by fine-tuning their parameters. Since population size is the most influential parameter for evolution search, different population sizes were tested to provide the optimal parameter settings for three algorithms.

In order to accurately assess the reliability of each algorithm, 10 random seeds were used for the 10 random seed trial runs for each algorithm. The random seeds were generated based on the current clock time. The performance of each algorithm was compared not only by their average cost value over 10 independent runs, but the best cost among 10 independent runs as well.

Table 7.2. Summary of algorithm parameters (with default settings for population size)

<table>
<thead>
<tr>
<th>Parameters</th>
<th>GA</th>
<th>CMA-ES</th>
<th>DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>N=20</td>
<td>N=λ=12,14,16</td>
<td>N=10D</td>
</tr>
<tr>
<td>Termination criteria</td>
<td>24,000 evaluations</td>
<td>Improvement &lt;10%</td>
<td>100 generations</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>0.8</td>
<td>NA</td>
<td>1</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>0.15</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Variable representation</td>
<td>Binary</td>
<td>Real</td>
<td>Real</td>
</tr>
<tr>
<td>Length of chromosome</td>
<td>16, 32, 64</td>
<td>16, 32, 64</td>
<td>16, 32, 64</td>
</tr>
<tr>
<td>Selection rate (selection pressure)</td>
<td>0.5</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Elitism</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Recombination weights</td>
<td>NA</td>
<td>[0.3818,0.2458,0.1663, 0.1098,0.0660,0.0302]</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[0.3448,0.2299,0.1626, 0.1149,0.0779,0.0477, 0.0221]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[0.3151,0.2157,0.1575, 0.1163,0.0843,0.0581, 0.0360,0.0169]</td>
<td></td>
</tr>
<tr>
<td>Parent number</td>
<td>NA</td>
<td>8,16,32</td>
<td>NA</td>
</tr>
</tbody>
</table>

Note: D is length of chromosome;
All three algorithms were programmed by MATLAB (The Mathworks Inc., Natick, MA). Due to the large computational cost, these algorithms were executed on the LION-XO and Hammer Cluster at Penn State’s High Performance Computing (HPC) Center. The LION-XO PC Cluster has the following hardware resources in each node:

- Sun SunFire v20z 1U Rackmount Box
- Dual 2.4 GHz AMD Opteron Processors
- 8 GB of ECC RAM

The Hammer Cluster can interactively execute MATLAB code through the GUI mode instead of batch mode. It has the following hardware resources in each node:

- Sun SunFire v40z 3U Rackmount Boxes
- Quad 2.6 GHz AMD Opteron Processors
- 32 GB of ECC RAM/584 GB of ECC RAM

7.5 Results

7.5.1 Discretization schemes comparison

The whole zNose™ spectrum was divided into 16, 32 and 64 windows and the GA, CMAES and DE were tested on each of these three discretization schemes. The control parameters for these three algorithms are listed in section 4.

Tables 7.3-5 present the search results by GA, CMAES and DE for the three different discretization schemes. Figures 7.7-9 show runtime convergence plots for 16 windows, 32 windows and 64 windows schemes executed by the GA, CMAES and DE, respectively.
Table 7.3: GA results summary (default population size=20)

<table>
<thead>
<tr>
<th>Discretize scheme</th>
<th>Mean</th>
<th>Std.</th>
<th>nfe</th>
<th>Best results</th>
<th>Data dimension reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows=16</td>
<td>0.0526</td>
<td>0</td>
<td>4000</td>
<td>0.0526</td>
<td>44%</td>
</tr>
<tr>
<td>Windows=32</td>
<td>0.0456</td>
<td>0.0108</td>
<td>4000</td>
<td>0.0263</td>
<td>50%</td>
</tr>
<tr>
<td>Windows=64</td>
<td>0.0509</td>
<td>0.0074</td>
<td>4000</td>
<td>0.0351</td>
<td>50%</td>
</tr>
</tbody>
</table>

Using the default population size (N=20), the genetic algorithm generated the best average performance over ten random seeds runs with the 32 windows scheme, as shown in Table 7.3. The average classification error rate was 0.0456 and the best classification error rate among 10 random seeds runs was 0.0263. The 64-window scheme was superior to the 16-window scheme for both the average objective function and best results. The algorithms were stopped after 200 generations. Since the population size of 20 was used, a total of 4000 function evaluations (nfe) were executed before each search was terminated. Both the 64-window and 32-window schemes reduced the initial input data dimensionality by 50%, while the 16-window scheme reduced the number of wavelengths by 44%. Figure 7.7 illustrates the typical searching history of genetic algorithms over 200 generations with three discretization schemes.

Figure 7.7. GA evolutionary history
Table 7.4 shows the results of three discretization schemes tested by the CMA evolutionary strategy. CMAES determines the default population size ($\lambda$) based on the number of variables to be optimized (Equation 7.2), so the population sizes corresponding to 16-window, 32-window and 64 window schemes were 12, 14, and 16 respectively. In contrast to GA, the stopping criteria for CMAES was when the maximum standard deviation of the decision variables is smaller than 0.25, so a different number of function evaluations were used for each independent run. It was observed that the CMAES 64-window scheme provided the least classification error rate. The average cost (classification error rate) was 0.034 and the best result among 10 independent runs was 0.0263. The CMAES 32-window scheme also found the least cost of 0.0263, but its average performance over 10 runs was inferior to the 64-window scheme. Both the 64-window and 32-window schemes reduced the input data dimensionality by 50%, which was slightly higher than the 44% from 16-window scheme. The relatively low standard deviation for the three schemes indicated that these algorithms had relatively stable performance over 10 runs. It was shown that the number of function evaluations increased twice as the number of variables doubled. Figure 7.9 illustrates the evolutionary history of the three schemes, from which it can be observed that CMAES resulted in a smaller classification error rate using the 64-window scheme, but took more time to do so.

Table 7.4: CMAES results summary (default $\lambda=12,14,16$ for windows 16, 32, and 64 respectively)

<table>
<thead>
<tr>
<th>Discretize scheme</th>
<th>Mean</th>
<th>Std.</th>
<th>nfe</th>
<th>Best results</th>
<th>Data dimension reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows=16</td>
<td>0.0632</td>
<td>0.01</td>
<td>572</td>
<td>0.0526</td>
<td>44%</td>
</tr>
<tr>
<td>Windows=32</td>
<td>0.0579</td>
<td>0.02</td>
<td>1028</td>
<td>0.0263</td>
<td>50%</td>
</tr>
<tr>
<td>Windows=64</td>
<td>0.0342</td>
<td>0.01</td>
<td>2050</td>
<td>0.0263</td>
<td>50%</td>
</tr>
</tbody>
</table>
Figure 7.8. CMAES evolutionary history

Differential evolution (DE) discretization scheme search results are shown in Table 7.5. Since the stopping criteria for DE is a maximum of 100 generations, and the default population size is $10D$ (D is the number of variables), the number of function evaluations were predetermined for each test. It was observed that the 64-window scheme resulted in best average performance and best found results. The best found result of 0.0175 is better than the 0.0263 value from CMAES and GA. However, many more function evaluations were needed for DE due to its large population size. The low standard deviation over 10 independent runs indicated that the repeatability of DE is good. All three discretization schemes reduced data dimensionality by 44%-50%, which was the same as the GA and CMAES.

Table 7.5: DE results summary (default population size $N=160$, 320 and 640 for windows 16, 32 and 64 respectively)

<table>
<thead>
<tr>
<th>Discretize scheme</th>
<th>Mean</th>
<th>Std.</th>
<th>nfe</th>
<th>Best results</th>
<th>Data dimension reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows=16</td>
<td>0.0535</td>
<td>0.0028</td>
<td>16000</td>
<td>0.0526</td>
<td>44%</td>
</tr>
<tr>
<td>Windows=32</td>
<td>0.0373</td>
<td>0.0044</td>
<td>32000</td>
<td>0.0351</td>
<td>50%</td>
</tr>
<tr>
<td>Windows=64</td>
<td>0.0246</td>
<td>0.0037</td>
<td>64000</td>
<td>0.0175</td>
<td>50%</td>
</tr>
</tbody>
</table>
Figure 7.9 illustrates the evolutionary history of DE through three different discretization schemes. The search quality was significantly improved by increasing the search variables, i.e. the number of windows.

![Figure 7.9. DE evolutionary history](image)

7.5.2 Parametric study for 64-window scheme

Since the lowest classification error rate was usually obtained by searching in the 64-window discretization scheme, this discretization scheme was selected for further parametric study. Population size is the control parameter that most influences the search space and convergence results, so thereafter, population size was manipulated for the three algorithms to optimize the control parameters.

For GA, three population sizes were tested for the 64-window scheme, i.e. 20 (the default value), 60, and 120. For CMAES, population sizes of 9, 16 (default value) and 32 were tested. For DE, population sizes of 320, 640 (default value) and 960 were tested.
Since the default population size for the 64-window scheme was tested in the previous section, two more settings were tested in this section and compared. It was observed that the best search results were obtained with large population size $N=120$ for GA, and $\lambda=32$ for CMAES, but for DE, the population size of 640 yielded better results than $N=960$ and $N=320$ (Table 7.6) both in terms of average performance and best results. If the best scenario from each algorithm is selected and compared, it can be seen that the CMAES and DE performed better than the GA, and the DE performed slightly better than the CMAES, in terms of average performance. Considering the best found result, the DE achieved lower error (0.0175) than the other two (0.0263), which proved the superior searching power of DE. However, considering the search efficiency, CMAES is the best of the three; its number of function evaluations (nfe=1836) were less than one tenth of GA (24000) and one thirtieth of the DE (64000). All three algorithms had good repeatability according to their small standard deviation over 10 random seed runs. Their performance on reducing data dimensionality was close; input wavelengths were reduced almost half varying from 48% to 52%.

Table 7.6: Parametric study of three algorithms for 64 windows discretization scheme

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$N$</th>
<th>$\lambda$</th>
<th>Mean</th>
<th>Std.</th>
<th>nfe</th>
<th>Best results</th>
<th>Data reduction</th>
<th>dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA: N=20</td>
<td>0.0509</td>
<td>0.0074</td>
<td>4000</td>
<td>0.0351</td>
<td>50%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GA: N=60</td>
<td>0.0439</td>
<td>0.0083</td>
<td>12000</td>
<td>0.0351</td>
<td>48%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GA: N=120</td>
<td>0.0377</td>
<td>0.0065</td>
<td>24000</td>
<td>0.0263</td>
<td>52%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMAES: $\lambda=9$</td>
<td>0.0491</td>
<td>0.0192</td>
<td>2180</td>
<td>0.0263</td>
<td>51%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMAES: $\lambda=16$</td>
<td>0.0342</td>
<td>0.0097</td>
<td>2050</td>
<td>0.0263</td>
<td>51%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMAES: $\lambda=32$</td>
<td>0.0289</td>
<td>0.0059</td>
<td>1836</td>
<td>0.0263</td>
<td>50%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DE: N=320</td>
<td>0.0281</td>
<td>0.0037</td>
<td>32000</td>
<td>0.0263</td>
<td>49%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DE: N=640</td>
<td>0.0246</td>
<td>0.0037</td>
<td>64000</td>
<td>0.0175</td>
<td>50%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DE: N=960</td>
<td>0.0249</td>
<td>0.0036</td>
<td>96000</td>
<td>0.0263</td>
<td>50%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figures 7.10-12 are parametric comparisons over different population sizes for the three algorithms. Since the elitism was introduced for the GA and DE, the search cost for each generation decreased monotonically. The CMA evolutionary strategy did not use elitism and hence the evolutionary curve was more ragged, but they generally followed the decreasing trend. It was observed that the fitness value was not improved after 100 generations for GA, and the search performance was not improved after 40000 function evaluations. Instead, the flexible stopping criteria helped the CMAES stop when the search performance did not improve after a certain number of function evaluations, which substantially saved computation time.

![Graph](image.png)

Figure 7.10. GA parametric study
Figure 7.11. CMAES parametric study

Figure 7.12. DE parametric study

7.5.3 Optimal spectral data selection

Wavelength selection using three discretization schemes are illustrated in Figure 7.13. The selected wavelengths illustrated were the best search results from each
discretization scheme. Classification error rates from these three wavelength selections were 0.0526, 0.0351 and 0.0175. It was observed that although different wavelengths were selected from different discretization schemes, some wavelengths were selected more often than others. For example, wavelengths between 30-50, 200-250, 400-420 and 475-512 were more relevant for damaged apple classification. The wavelengths between 450-480 were not selected in the 16 and 32-window schemes, but by selecting these wavelengths, the classification performance was improved as shown in the 64-window schemes. From these three plots, it was also observed that the reason the classification performance was improved was that the whole spectrum was discretized more finely, allowing the optimization approach more flexibility to search more variable combinations.

![Diagram of wavelength selection for 16, 32, and 64 windows](image)

Figure 7.13. zNose™ wavelength selection illustration for 16, 32 and 64 windows schemes (Note: the solid blue line represents selected wavelengths, the dashed red line shows removed wavelengths)
The number of times a window was selected by the CMAES algorithm over 10 independent runs for the 64-window scheme is shown in Figure 7.14. If wavelengths being selected more than 5 times are considered important, there were 31 windows in this category. They were mainly located in the area of windows 4-8, windows 18-21, windows 30-34, some windows from 40-50, and windows 51-64. Wavelengths located in these areas were key for the differences between damaged apples and healthy apples.

Figure 7.14. Frequency of selection of wavelengths in 64-window scheme

7.6 Discussions

From results above, we can reach the following observations:

1) Discretizing the whole spectrum into more windows (more variables) yielded better search results by the CMAES and DE algorithms. This fact does not hold for the GA with the population size of 20. The plausible reason is that the population size is too small to exploit the advantage of large number of variables, so the 32-variable scheme
yielded a slightly better result than the 64-variable scheme with a population size of 20. However, if the population size is set to 120, the 64-window scheme did perform better (average cost value=0.0377) than the 32-window scheme (average cost value=0.0428). More variables provided bigger search space for exploring, making it easier for algorithms to find the minimum cost value. The best-ever-found result with the 16-window scheme for all three algorithms was 0.0526. This number was improved to 0.026 for the 32-window scheme, and 0.0175 for the 64-window scheme.

2) Parametric studies supported the previous claim (De Jong, 1975) that better search results were usually obtained by using a large population size. For both the GA and CMAES, the search results were improved by using population sizes of 120 and $2\lambda$ (double default setting according to the variable number), respectively. The search performance for DE was improved by using a population size of $10D$ ($D$ is the number of variables) instead of $5D$, although it was not improved by increasing to $15D$. However, it should be noted that the improved search performance was obtained at the cost of search efficiency, i.e. both GA and DE took two times of number of function evaluations when the population size was doubled. The DE achieved the best minimum classification error rate (0.0175) among all three algorithms, largely because of the much larger population size. It took an extremely long time (64000 NFEs) to get the results. Considering the relatively small population size of CMAES even after doubling its default setting, CMAES achieved satisfactory search results with one tenth the computation time of the GA and less than one thirtieth the computation time of the DE.

3) The overall evaluation of three algorithms showed that both the CMAES and DE had superior wavelengths searching capacity compared to the GA. The simple GA
could not explore the full searching space due to its limitations on single crossover, roulette wheel selection, and mutation. The DE has good search quality, but consumed extremely large computation time compared to the CMAES. If search efficiency was also a consideration, the DE is not a good choice. The CMAES has the impressive search quality and search efficiency not only because of its inherited searching algorithm, but also its flexible stopping criteria; when the maximum standard deviation of the variables is smaller than 0.25, it can stop the search process when the minimum cost value is reached. Similar measures could be used for the GA and DE stopping criteria in the hope to improving search efficiency.

4) Although the heuristic optimization generated different wavelength selections each time, a statistical count of wavelength selection over 10 random seed runs by CMAES with the 64-window scheme showed that certain wavelengths were more often selected than others. These areas were considered to have the key information to differentiate damaged apples from healthy apples. By only using these relevant variables, the input data dimensionality could be reduced by 50%, which significantly reduced further data processing time.

7.7 Conclusions

In this research, a comparative study of three evolutionary algorithms (the GA, CMAES and DE) on the zNose™ wavelength selection was conducted. The zNose™ spectral data were discretized into 16, 32 and 64 windows, respectively, so the 512 variables in each spectrum could be reduced to 16, 32 and 64. Binary coding was used for the GA, and pseudo-real number coding was used for the CMAES and DE. The 64-
window scheme gave the best search results due to finer discretization and a fuller searching space, although this discretization scheme required more computation time. Three searching algorithms were further fine-tuned to determine the optimal control parameters for the 64-window scheme. It was found that the GA with a population size 120, CMAES with 32 and DE with 640 gave the best search results. The CMAES and DE achieved comparable and better average performance than GA, and the DE reached the minimum classification error rate of 0.0175, which was unmatchable from the other two algorithms. Considering the search efficiency and computation time, the CMAES was the most efficient algorithm; the number of function evaluations it used was less than one tenth of the GA and one thirtieth of the DE. All three algorithms could reduce data dimensionality 50% by selecting only the 50% relevant wavelengths, while improving the classification performance from 10.5% misclassified to a minimum of 1.8%.

This research provides a useful methodology for spectral data processing and variable selection which not only improves pattern recognition models’ performance by removing irrelevant information, but reduces the computation time by reducing input data dimensionality. These algorithms with a PNN classification model greatly enhanced the capability of the zNose™ for apple defect detection.
ABSTRACT

The Cyranose 320 electronic nose (Enose) and zNose™ are two instruments used to detect volatile profiles. In this research, feature level and decision level multisensor data fusion models, incorporated with covariance matrix adaptation evolutionary strategy (CMAES), were developed to combine the Enose and zNose™ data to improve performance for damaged apple classification over using the single instruments alone. Principal component analysis (PCA) was used for feature extraction and probabilistic neural networks (PNN) were developed as the classifier. Three feature-based fusion schemes were compared. Dynamic selective fusion achieved an average 1.8% and a best 0% classification error rate (be sure to be consistent in the classification results. I will look for these also) in a total of 30 independent runs. The static selective fusion approach resulted in a 6.1% classification error rate, which was not as good as using individual sensors (4.2% for the Enose and 2.6% for the zNose™) if only selected features were applied. Simply adding the Enose and zNose features without selection (non-selective
fusion) worsened the classification performance with a 32.5% classification error rate. This indicated that the feature selection using the CMAES is an indispensable process in multisensor data fusion, especially if multiple sources of sensors contain much irrelevant or redundant information. At the decision level, Bayesian network fusion achieved better performance than two individual sensors, with 11% error rate vs. 13% error rate for the Enose and 20% error rate for the zNose™, when soft evidence from the BP network was used. Bayesian network fusion did not produce better results than the two individual sensors when hard evidence and prior knowledge were used. It was proven that both the feature level fusion with the CMAES optimization algorithms and decision level fusion with soft evidence improved system classification performance. This methodology can also be applied to other sensor fusion applications.

8.1 Introduction

Apples (*Malus domestica*, Borkh) are one of the most commonly consumed fruits in the United States and the world at large. The United States is the second largest apple producing country with nearly 5 million tons production and 1.7 billion dollars revenue in 2004 (USDA-NASS, 2006). Pennsylvania is the 5th largest U.S. apple producing state, accounting for roughly 4-5% of national production (USDA-FAS, 2005).

Apples are usually stored for 6-10 months before arriving at grocery stores and being consumed by customers. During this long process of storage, spoilage and diseases may occur in apples, including the most common apple postharvest diseases: *Botrytis cinerea* Pers., *Penicillium expansum* Link, *Mucor piriformis* (Vikram et al., 2004). It is estimated that typically more than 10% of apples are lost due to spoilage during this
process (NE-179 Project report, 2001). Controlled atmosphere (CA) technology has been widely adopted for apple storage by reducing the temperature, and adjusting oxygen and carbon dioxide levels to inhibit the metabolic activities of apples. Under this confined storage condition, it is impossible for humans to enter the storage room and visually inspect apple conditions, which may consequently induce economic losses. In other cases, retailers want to guarantee that each bag of apples contains good quality individuals, devoid of spoilage and disease. However, some spoiled apples are occluded by healthy apples within a plastic bag and may not be detected visually. From the perspective of both storage managers and retailers, there is a need to develop a non-invasive, sensitive, and fast method to detect apple spoilage and diseases in confined storage rooms and package bags to reduce unexpected losses.

Various studies have proven that compositional changes in volatiles occur during fruit ripening, and vary depending on the presence of diseases and physical damage (Simon et al., 1996). By detecting these changes, deterioration in apples can be detected and differentiated from healthy apples. The electronic nose, which was dubbed in 1982 (Persaud and Dodd, 1982), has been widely used in food quality control, medical diagnosis, and homeland security (Schaller et al., 1998; Schiffman et al., 1997; Thaler et al., 2001). Several research groups have applied the Enose to predict fruit (apple, pear, and banana) ripeness (Brezmes et al., 2001; Llobet et al., 1999; Oshita et al., 2000). It has also been used for quality sorting of blueberries (Simon et al., 1996), spoilage identification of beef (Balasubramanian et al., 2004), peanut off-flavor detection (Osborn et al., 2001), sausage fermentation monitoring (Eklov et al., 1998), grain quality inspection (Jonsson et al., 1997) and other food products (Benady et al., 1995; Gardner et
al., 1992). Unlike analytical chemical instruments such as gas chromatography and mass spectrometry (GC-MS), the electronic nose does not detect and identify single volatiles, but differentiates smell patterns of vapor mixtures by using pattern recognition algorithms. Its processing time is also much faster than GC-MS. This characteristic gives the Enose an edge over GC-MS in certain applications when concern is not about the specific volatile compounds, but the overall smell patterns, such as food quality inspection and class differentiation.

Multi-sensor data fusion techniques try to combine data from multiple sensors, to get a better interpretation of the target than using individual sensors alone, if these sensors can provide complementary information (Hall, 1997). Usually, more sources of data provide more information and achieve better performance. For instance, humans and animals use multiple senses to improve their chances of survival. They use not only their vision system, but hearing, smell, and taste to determine if a product is edible. Typically, selection of good sensors and proper fusion processing techniques are two key components of a successful multi-sensor data fusion problem. This concept was first developed by the Department of Defense (DoD) and used for the location, characterization and identification of weapons and military units (Hall and Llinas, 2001). It has also been widely used in nonmilitary applications such as the implementation of robotics, medical diagnosis using multiple instruments, and environmental monitoring such as location of earthquakes, hurricanes, and other natural disasters (Hall, 1997). Mathematical algorithms, which are used to implement data fusion, are drawn from traditional disciplines such as digital signal processing, statistics, control theory, and artificial intelligence. They include Kalman filtering, clustering analysis, artificial neural
networks, Bayesian inference, Dempster-Shafer’s method, etc. (Hall and McMullen, 2004). Some attempts have been made to apply multisensor data fusion methods to combine different sensors and improve the performance of fruit quality inspection systems (Ozer et al., 1995; Roussel et al., 2003a; Roussel et al., 2003b; Steinmetz et al., 1996; Steinmetz et al., 1999a).

Two volatile sensing instruments, the Cyranose 320 electronic nose and zNose™, were applied to detect apple spoilage in this research. Both of these instruments have previously been used separately for food quality evaluation (Lammertyn et al., 2004; Li et al., 2005). In this research, the data from these two instruments were combined to improve the system performance for classification by developing multi-sensor data fusion models.

In this work, two different levels of data fusion models were explored. The data from two commercial volatile sensing instruments (the Cyranose 320 electronic nose and zNose™) were combined at the feature level and the decision level. In the feature level fusion, features were extracted from the Enose and zNose™ using principal component analysis (PCA) and these extracted features were fed into artificial neural networks (ANN) for classification. Since previous studies (Li and Heinemann, 2006b; Li et al., 2006) showed that both the Enose and zNose™ have redundant sensors, at this level, the optimization method covariance matrix adaptation evolutionary strategy (CMAES) was applied to select relevant sensors that provide complementary information and to optimize the fusion model. Three different feature-based fusion models were implemented and compared. In the decision level fusion, Bayesian network was used to fuse classification results after these two instruments made a declaration of identity.
8.2 Methods

8.2.1 Data measurement

The Cyranose 320 electronic nose (Smith Detection, Herts, UK) consists of 32 internal thin film carbon black polymer composite sensors, which can function in ambient air temperature. The resistance from these conducting chemiresistors increases when vapor-phase analytes absorb on the surface and disrupt the conductive pathways (Cyrano Science Inc., 2000). These 32 composite polymer sensors are non-specific to a wide range of volatile compounds and are capable of recognizing odors with a pattern recognition system. The zNose™ (Electronic Sensor Technology, Newbury Park, CA) consists of one capillary column and one surface acoustic wave sensor. Volatile compounds that pass through the capillary column are separated based on their different solubility and enter the SAW sensor at different times. The oscillating frequency from the SAW sensor changes due to the mass change caused by volatile compounds and results in a frequency shift with respect to elution time. Both the Enose and zNose™ signals are shown in Figure 8.1.

Figure 8.1. Enose and zNose™ “smellprints”
Red ‘Delicious’ apples were purchased from a local grocery store and were intentionally damaged by inducing a 10 mm deep cross-slice cut on the top. These damaged apples were exposed to room air for deterioration development. The measurements were conducted every other day from day 4 to day 14 after the cut treatment. Other apples without the cut treatment were considered “healthy” apples. Apple samples were kept in room air for 6 hours to reach the ambient air temperature before each test. The equilibrium time for headspace concentration was also 6 hours. Apples were maintained at room air temperature (20±1 °C) for 48 hours between each measurement. A 2 L glass jar was used as a headspace gas concentration chamber, sealed by a plastic cap with a Teflon septum.

The Enose sampled volatile compounds emitted by apples by inserting a 50 mm long snout needle into the 5 mm hole in the lid of the glass jar. These measurements can be stored in the Enose and downloaded to the computer by an RS 232 cable. The zNose™ was equipped with a 5 cm long sampling needle at the inlet, which is inserted into the concentration chamber for sampling. The sampling time was 10 s, during which the gas sample was released from the trap inside the system and carried over the column (DB-5) in a helium flow of 3 cm³/min. The zNose™ was baked for 5 s after each data sampling period to clean the SAW detector. One blank system purge run was conducted between each sample measurement to attain a stable baseline.

Sampling was conducted at three different times: March, June, and September, 2005. The sampling time and number of samples for each group are shown in Table 8.1.
Table 8.1. Sampling protocol for the Enose

<table>
<thead>
<tr>
<th>Period of sampling</th>
<th>Dates of sampling</th>
<th>No. of replications</th>
<th>No. of samples</th>
<th>Data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>March, 2005</td>
<td>5,7,9,11,14</td>
<td>24</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>June, 2005</td>
<td>4,6,8,10</td>
<td>24</td>
<td>96</td>
<td>2</td>
</tr>
<tr>
<td>September, 2005</td>
<td>5,7,9,11,13</td>
<td>48</td>
<td>240</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>456</td>
<td>Pooled</td>
</tr>
</tbody>
</table>

8.2.2 Data fusion schemes

Multisensor fusion provides a collaborative approach to improve classification accuracy by using multiple sensors. Four steps were followed to accomplish multisensor data fusion:

1) Acquire apple headspace volatile smellprints using the Enose and zNose™;

2) Select and apply the proper multisensor data fusion method. Bayesian classifiers and ANN classification method were applied at feature level and decision level;

3) Evaluate the multisensor data fusion models by comparing their performance to the previous individual sensor classification models;

4) The proposed sensor fusion method was accepted, rejected, or refined based on classification error rate.

Two levels of data fusion architecture were investigated in this paper (Figure 8.2):

1) Feature level fusion: features were first extracted by using principal component analysis (PCA) from each source of data. These features were concatenated into a single feature vector, which in turn is used as input to an identity declaration technique artificial neural network;
2) Decision level fusion: each sensor performed an identity declaration process based only on its own single-source data. The identity declarations provided by the individual sensors were combined using the Bayesian inference decision level fusion technique.

The PCA was used for feature extraction, probabilistic neural networks, and backpropagation (BP) networks. Bayesian inference was used for data fusion and classification. Covariance matrix adaptation evolutionary strategy (CMAES) was used for feature selection from the Enose and zNose™ and to optimize data fusion models. In the sections that follow, these algorithms are introduced.

Figure 8.2. Feature level (a) and decision level (b) fusion schemes
8.2.3 Principal component analysis

Principal component analysis (PCA) is a linear projection of multidimensional data onto different coordinates based on maximum variance and minimum correlation. As a result, less significant components can be eliminated, reducing the data representation to only those responsible for the most significant contribution. PCA was used for feature extraction. The data were processed by PCA and only the principal components that explain more than 0.5% variances were selected. Based on this rule, the first five principal components were extracted from the Enose data, the first ten principal components were extracted from the zNose™ data, and the first six principal components were selected from fused raw data of the Enose and zNose™.

8.2.4 PNN and BP networks

A probabilistic neural network (PNN) was designed to develop feature-based fusion models. The PNN consists of an input layer, radial basis layer, competitive layer and output layer (Figure 8.3). The input layer combines the extracted features from the Enose and zNose™, and presents them to the network. The radial basis layer calculates distances between the testing input vectors and the training input vectors, and generates a vector measuring the distance between testing input and a training input. The competitive layer adds these contributions for each class of inputs and produces a vector of probabilities and uses a competitive transfer function to pick the maximum of these probabilities, returning a 1 for the winner whose probability is the maximal and a 0 for the other class. The number of elements in the input layer was set to the total number of principal components extracted from the Enose and zNose™. The number of neurons in
the hidden layer was set to the size of the training data set. Neurons in the output layer were set to two, which represents two classes: healthy and damaged apples. The total of 456 data vectors were divided into a training set (342 vectors) and a testing set (114 vectors). The classification error rate was obtained when the testing set was processed by the trained PNN model (Equation (8.1)). It was also used as the cost function when the evolutionary algorithm was carried out.

\[
\text{classification error rate} = \frac{\text{number of misclassifications}}{\text{number of testing set samples}} \quad (8.1)
\]

The Back Propagation (BP) network is known for its ability to generalize well on a wide variety of problems. This network is generally robust, although one drawback is that the training is slow. A three-layer (input, hidden, and output layer) BP network was designed in this study as shown in Figure 8.4. The extracted principal components which explain more than 0.5% variances were fed into input neurons. The number of hidden layer neurons was determined by trial-and-error, and set at 30. The output layer has two neurons to represent healthy and damaged apples. For more details, please refer to reference (Li and Heinemann, 2006a).

![Figure 8.3. PNN network architecture (IW is input weights; \(|| \text{dist} ||\) is a vector whose elements indicate how close the input is to the vectors of the training set; LW is layer weights)](image-url)
8.2.5 CMAES

As a state-of-the-art version of the evolutionary strategy, the covariance matrix adaptation evolutionary strategy (CMAES) is a heuristic optimization algorithm which can be used to select the most relevant features from a multivariate space (Hansen, 2005). Previous studies (Li and Heinemann, 2006b; Li et al., 2006) have proven that both the Enose and zNose™ have redundant information, and better performance could be achieved by selecting the most relevant and complementary features. For the Enose, 32 sensors were considered decision variables to select; for the zNose™, the continuous chromatograph was discretized into 64 windows and each window was treated as one decision variable to select, which greatly reduces the number of decision variables.

The initial population is generated by sampling a normal distribution with a user-specified mean value and standard deviation for each decision variable. Offspring generation, selection and recombination, covariance matrix adaptation, and step size control are four key operators in the process of evolution (Hansen, 2005). CMAES can be described as a randomized black box search whose computation flowchart is shown in Figure 8.5.

Figure 8.4. BP network architecture (IW is input weights; LW is layer weights; b is bias)
Figure 8.5. CMAES algorithm flow chart

Algorithm parameters are always important for search efficiency, global optimization quality, and algorithm reliability. In CMAES, the following parameters were used: $\lambda, \mu, w_i, c_{cov}, c_c, c_{cov}, c_\sigma$ and $d_\sigma$. Among them, population size $\lambda$, parent number $\mu$ and recombination weight $w_i$ are given in the algorithm (Hansen, 2005). The
default values are defined in the following equations and other parameters can be derived using these three parameters:

\[ \lambda = 4 + [3 \ln n] \]  \hspace{1cm} (8.2)

\[ \mu = [\lambda / 2] \]  \hspace{1cm} (8.3)

\[ w_i = \frac{\ln(\mu + 1) - \ln i}{\mu \ln(\mu + 1) - \sum_{j=1}^{\mu} \ln j} \] for \( i = 1, \ldots, \mu \)  \hspace{1cm} (8.4)

Where

\( n \) is the number of decision variables.

The stop criteria for the Enose sensor selection application is when the maximum standard deviation of the decision variables is smaller than 0.25 or the cost value reaches its global minimum 0.

8.2.6 Bayesian network fusion

Equation (8.5) and (8.6) are two basic Bayes rules, and in data fusion applications, Equation (8.6) can be expressed as (8.7):

\[ P(A/ B) = \frac{P(B/A)P(A)}{P(B)} \]  \hspace{1cm} (8.5)

\[ P(A/ B) = \frac{P(B/A)P(A)}{\sum_i P(B/ A_i)P(A_i)} \]  \hspace{1cm} (8.6)

where \( P(A/ B) \) is the probability of \( A \) conditioned on the occurrence of \( B \).

\[ P(O_j/ data) = \frac{P(data/ O_j)P(O_j)}{P(data)} \]  \hspace{1cm} (8.7)

\( P(O_j) \) = prior Object\(_j\) distribution probability
\[ P(data) = \sum_j P(data / Oj)P(Oj) \]

Since the individual sensor reports rely on different working principles, these probabilities can be considered conditionally independent:

\[ P(data / Oj) = P(E, Z / Oj) = P(E / Oj)P(Z / Oj) \]  \hspace{1cm} (8.8)

where \( P(data|Oj) \) is the posterior probability of observing the data given the objects \( O_j \) are present.

\( O_1=\text{good}; \ O_2=\text{bad}; \)

\( P(O_1)=P(O_2)=0.5; \)

Finally, Bayes’s rule can be used as:

\[
P(Oj / data) = P(Oj / E, Z) = \frac{P(E, Z / Oj)P(Oj)}{P(E, Z)} = \frac{P(E / Oj)P(Z / Oj)P(Oj)}{P(E / O1)P(Z / O1)P(O1) + P(E / O2)P(Z / O2)P(O2)} \]  \hspace{1cm} (8.9)

In Bayesian network, \( P(E,Z) \) term in equation (8.9) can be derived as:

\[
P(E,Z) = \sum_j P(E, Z / Oj)P(Oj) = P(E, Z / \text{good})P(\text{good}) + P(E, Z / \text{bad})P(\text{bad}) = \]

\[
P(E / \text{good})P(Z / \text{good})P(En / E)P(Zn / Z)P(\text{good}) + \]

\[
P(E / \text{bad})P(Z / \text{bad})P(En / E)P(Zn / Z)P(\text{bad}) \]  \hspace{1cm} (8.10)

The Bayesian fusion flowchart and Bayesian network structure are shown in Figure 8.6 and Figure 8.7.
Figure 8.6. Bayesian fusion flowchart

Enose raw data → BP network classicization → Bayesian Network fusion → Output

zNose raw data → BP network classicization → Bayesian Network fusion → Output

Apple (Ag/Ab)

P(Eg/Ab) = .85
P(Eb/Ag) = .15
P(Eg/Ab) = .15
P(Eb/Ab) = .85

Enose (Eg/Eb) → Enose ANN (g/b)

zNose (Zg/Zb) → zNose ANN (g/b)

P(Zg/Ag) = .91
P(Zb/Ag) = .09
P(Zg/Ab) = .43
P(Zb/Ab) = .57

Instantiated State (0, 1) or soft evidence, e.g. (.81, .19)

Figure 8.7. Bayesian Network structure
8.3 Computational experiment

8.3.1 CMAES coding methods comparison

If each Enose sensor (32 total) and zNose™ wavelength window (64 total) is considered as one variable, there are 96 variables to select. The chromosome (which consists of decision variables) will be too long if binary coding methods are used. To reduce chromosome length (i.e. number of decision variables) and at the same time fully exploit the advantage of CMAES for solving continuous number problems, a real number coding scheme was developed. Two different length real number coding schemes for the CMAES were developed and compared.

8.3.1.1 48-variable scheme

Each variable is evenly divided into four segments, and each segment represented one out of four possibilities of two sensors selected, as defined in Equation (8.11). For instance, if \( x \) falls into the range of \([0.25, 0.5)\), it represents the state of \([1,0]\) that the first sensor was selected and the second was not selected. By doing this, each variable can represent four sensor selection possibilities of two sensors, and a total of 96 decision variables can be represented by 16 real number variables. This reduced the variable number by 50% compared to binary coding.

\[
x \in \begin{cases} 
[0,0.25) \Rightarrow [0,0] \\
[0.25,0.50) \Rightarrow [1,0] \\
[0.50,0.75) \Rightarrow [0,1] \\
[0.75,1.00] \Rightarrow [1,1] 
\end{cases} \tag{8.11}
\]
8.3.1.2 24-variable coding scheme

Similarly, a 24-variable coding scheme can be carried out by using only one real number variable to represent 16 possible selections for four sensors (Equation (8.12)).

\[
\begin{align*}
&0.0000, 0.0625) \Rightarrow [0, 0, 0, 0] \\
&0.0625, 0.1250) \Rightarrow [0, 0, 1, 0] \\
&0.0125, 0.1875) \Rightarrow [0, 0, 1, 0] \\
&0.1875, 0.2500) \Rightarrow [0, 1, 1, 0] \\
&0.2500, 0.3125) \Rightarrow [0, 1, 0, 1] \\
&0.3125, 0.3750) \Rightarrow [0, 1, 0, 1] \\
&0.3750, 0.4375) \Rightarrow [0, 1, 1, 0] \\
&0.4375, 0.5000) \Rightarrow [0, 1, 1, 1] \\
&0.5000, 0.5625) \Rightarrow [1, 0, 0, 0] \\
&0.5625, 0.6250) \Rightarrow [1, 0, 0, 1] \\
&0.6250, 0.6875) \Rightarrow [1, 0, 1, 0] \\
&0.6875, 0.7500) \Rightarrow [1, 0, 1, 1] \\
&0.7500, 0.8125) \Rightarrow [1, 1, 0, 0] \\
&0.8125, 0.8750) \Rightarrow [1, 1, 0, 1] \\
&0.8750, 0.9375) \Rightarrow [1, 1, 1, 0] \\
&0.9375, 1.0000) \Rightarrow [1, 1, 1, 1]
\end{align*}
\]  

(8.12)

By doing this, the number of decision variables can be further reduced from 48 to 24, which can reduce population size and computation time.

8.3.2 Three feature level fusion schemes comparison

At the feature level, three schemes were proposed and tested as shown in Figure 8.8.

1) Non-selective fusion scheme:

This scheme uses all raw data without data dimension reduction.
2) Static selective fusion scheme:

This scheme is different from the first scheme in that it uses only partial raw data which were previously selected by CMAES. The seven sensors and 256 wavelength values were selected for the Enose and zNose™ individually by the CMAES.

3). Dynamic selective fusion scheme:

This is a dynamic selection fusion scheme, which uses the CMAES to dynamically select sensors from the Enose and wavelengths from the zNose™ simultaneously instead of separately.

(a)

![Diagram](image)

(b)

![Diagram](image)

(c)

![Diagram](image)

Figure 8.8. Three ANN fusion schemes: (a) non-selective fusion; (b) static selective fusion; (c) selective fusion.
8.4 Results

8.4.1 Two coding methods comparison

Two real number coding methods (48-variable and 24-variable) were executed by the CMAES and their search performances were compared. Figure 8.9 shows a typical search history for a 48-variable coding method. It is shown that the CMAES with the 48-variable real number coding can effectively reduce the classification error rate from 0.16 at the beginning to 0 at the end of the search.

![Figure 8.9. A typical 48-variable search history by the CMAES](image)

Since the evolutionary strategy is a heuristic search method and each run may return a different search result, two coding methods were executed 30 times each with 30 random seeds. Their average performance parameters are listed in Table 8.2. Three performance parameters were compared: best fitness value or minimum cost value which is a measure of search quality, the number of function evaluations (nfe) which is a measure of search efficiency, and the number of sensors selected from the Enose and
zNose™, which is a measure of dimensionality reduction. It is observed that both coding methods can reduce the classification error rate to 0 which appeared three times in both cases. Their 30-time run average performances are comparable, 1.7 versus 1.8 minimum error rate. Figure 8.10 is an illustration of selected wavelength windows from the zNose™ and sensors from the Enose.

Table 8.2. 48-variable coded CMAES for sensor fusion searching (over 30 random runs)

<table>
<thead>
<tr>
<th>No.</th>
<th>48-variable coding method</th>
<th>24-variable coding method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimum cost</td>
<td>NFE</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3152</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3452</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3602</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3622</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4082</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4172</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>4172</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>3182</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>3962</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>3662</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>3272</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>3962</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>3902</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>5462</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>3782</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>4682</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>4532</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>4232</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>3632</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>4232</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>3962</td>
</tr>
<tr>
<td>22</td>
<td>4</td>
<td>3392</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>5012</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>4472</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
<td>3092</td>
</tr>
<tr>
<td>26</td>
<td>2</td>
<td>3992</td>
</tr>
<tr>
<td>27</td>
<td>2</td>
<td>3962</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>2972</td>
</tr>
<tr>
<td>29</td>
<td>2</td>
<td>3572</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>3392</td>
</tr>
</tbody>
</table>

| Average | 1.7 | 3886 | 16 | 31 | 1.8 | 4503 | 19 | 31 |
For the best scenario, when the classification error rate was 0 and the least sensors were selected from both the Enose and zNose™, the following sensors/wavelength windows were selected:

- Selected sensors from the Enose (22): 3  5  6  8  10  11  12  13  14  15  16  17  18  19  20  21  24  26  28  30  31  32
- Selected wavelength windows from the zNose™ (29): 1  2  5  6  7  8  11  12  13  18  19  20  22  23  24  28  34  45  46  47  50  54  57  59  60  61  62  63  64

Figure 8.10. Selected sensors from the zNose™ (top) and Enose (bottom)
8.4.2 Feature level data fusion comparison

Based on the methods presented above, three feature level data fusion schemes were executed and compared (Table 8.3). In the non-selective scheme, all 32 sensors from the Enose and 64 zNose™ spectral windows were used separately and jointly for development of classification models. In this case, the sensor fusion model which combines all sensors did not yield a better performance (32.5% error rate) than individual sensor models (15% and 23% error for the Enose and zNose™, respectively). In the static selective fusion scheme, the Enose and zNose™ sensors which were selected separately, were jointly used in the sensor fusion model. The classification error rate was reduced from 32.5% to 6.1%;. In the third scenario, all 32 sensors from the Enose and 64 windows from the zNose™ were coded in one chromosome and dynamically selected using the CMAES and the selected sensors were fed into the PNN-based sensor fusion model. The results were encouraging; the dynamic selective sensor fusion model achieved an average 1.5% error rate in 30 independent runs, which outperformed the individual sensor classification models.

Table 8.3. Classification error rate comparison between sensor fusion schemes and individual sensors

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Individual sensor</th>
<th>Sensor fusion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Enose</td>
<td>zNose™</td>
</tr>
<tr>
<td>Non-selective</td>
<td>15%</td>
<td>23%</td>
</tr>
<tr>
<td>Dynamic selective (for</td>
<td>4.2%</td>
<td>2.6%</td>
</tr>
<tr>
<td>individual sensors)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static selective (for</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>fusion)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic selective (for sensor</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>fusion)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8.4.3 Decision level Bayesian network fusion

At the decision level, a Bayesian network, based on Bayesian decision rules, was developed. The Bayesian network utilizes two sources of information to make its fusion decisions: the classification results obtained independently from the Enose and zNose™ classifier, and two classifiers’ previous performance (also called prior knowledge). The classification results can either be binary numbers (1 or 0) as generated by the PNN or a real number indicating the probability of the sample belonging to a certain class as produced by the BP network. The Enose and zNose™ classifiers may have classification decisions that agree or conflict. In the situation where both classifiers make the same wrong decisions, there is no way to use Bayesian fusion to improve the classification results. In the situation of decision conflict between two classifiers, there must be one correct and one wrong, and the Bayesian network fusion is designed to reduce this part of error based on prior knowledge or probabilistic outputs. In this section, both the PNN and BP classifiers’ results were fused by Bayesian network using hard evidence (binary outputs) and soft evidence (probabilistic outputs).

8.4.3.1 PNN hard evidence Bayesian fusion

Both the Enose and zNose™ PNN classifiers produced binary number outputs: 1 means it belongs to this element (good or bad), 0 means otherwise. These binary outputs are called “hard evidence” in order to differentiate from the BP network probabilistic outputs, which are called “soft evidence”. Table 8.4 shows Bayesian fusion performances in two situations: 14% error rate in the non-selective situation and 3.5%
error rate in the static selective fusion. These results are only as good as the performance of the individual Enose or zNose\textsuperscript{TM}, but not better.

### Table 8.4. Bayesian fusion from PNN hard evidences

<table>
<thead>
<tr>
<th></th>
<th>Enose</th>
<th>zNose\textsuperscript{TM}</th>
<th>Bayesian fusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non selective</td>
<td>14%</td>
<td>15%</td>
<td>14%</td>
</tr>
<tr>
<td>Static selective*</td>
<td>4.4%</td>
<td>3.5%</td>
<td>3.5%</td>
</tr>
</tbody>
</table>

* static selective sensors used:
Enose seven sensors: 5, 6, 22, 25, 26, 27, 32;
zNose\textsuperscript{TM} 37 windows: 5, 6, 7, 8, 12, 16, 17, 19, 20, 23, 25, 26, 28, 29, 30, 32, 34, 36, 40, 42, 45, 47, 49, 52, 53, 54, 55, 56, 57, 61, 62, 63, 64;

### 8.4.3.2 BP soft/hard evidence Bayesian fusion

Both the Enose and zNose\textsuperscript{TM} BP classifiers produced probabilistic outputs, which indicate the probability of the sample belonging to certain classes. By using this information, the Bayesian fusion was expected to reduce the error rate. To determine if prior knowledge and soft evidence should be used, three fusion schemes were tested:

(a) Fusion 1: soft evidence with prior Enose and zNose\textsuperscript{TM} performances;

(b) Fusion 2: soft evidence without prior Enose and zNose\textsuperscript{TM} performance;

(c) Fusion 3: hard evidence with prior Enose and zNose\textsuperscript{TM} performance.

The simulation results in Table 8.5 show that the fusion 1 and fusion 2 schemes performed equally well with 11% error rate, which was better than the individual sensors (13% and 20% error rate for the Enose and zNose\textsuperscript{TM} respectively). However, fusion 3, which used only hard evidence (transformed from probability to binary number) and prior performance of the Enose and zNose\textsuperscript{TM} provided similar results as the enose by itself (13%).
Table 8.5: Three Bayesian fusion schemes performance comparison

<table>
<thead>
<tr>
<th></th>
<th>Enose</th>
<th>zNose™</th>
<th>Fusion 1</th>
<th>Fusion 2</th>
<th>Fusion 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error rate</td>
<td>15/114 (13%)</td>
<td>22/114 (20%)</td>
<td>13/114 (11%)</td>
<td>13/114 (11%)</td>
<td>15/114 (13%)</td>
</tr>
<tr>
<td>Error 1*</td>
<td>25</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>Error 2*</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

*Error 1 is defined as the number of errors caused by the Enose and zNose™ decisions conflict;  
*Error 2 is defined as the number of errors caused by the Enose and zNose™ when both made the wrong decision.

8.5 Discussions

Both 48-variable and 24-variable real number coded CMAES achieved comparably good results by reducing the classification error rate from 16% at the beginning to 1.7~1.8% after 3300~4500 function evaluations, if 30 independent runs were executed and the average performance was considered. Both coding methods found the global minimum error rate 0 three times in 30 total independent runs. However, the 48-variable real number coded CMAES was more efficient than its counterpart, using only 3386 function evaluations, 25% less than that of 24-variable real number coded CMAES. Considering the dimensionality reduction, both schemes reduced zNose™ chromatograph windows from 64 to 31 (52% reduction). However, the 48-variable scheme reduced more data dimensionality (50%) for the Enose than its counterpart (41%). Thus, the 48-variable scheme was recommended for feature selection in this data fusion application.

Based on the feature selection results, both the Enose and zNose™ have redundant sensors. However, the zNose™ raw data have more noise and irrelevant information than that of the Enose. The zNose™ data are more compressible, e.g. using
all 512 wavelengths gave a classification error rate of 23%, but after reducing its data dimensionality by 50%, the error rate was reduced to 2.6% (20% improvement). For the Enose, the number of sensors was reduced 40-50% while the classification error rate was reduced 10%. This indicates that the zNose™ has more irrelevant sensors, which adds noise and worsens the classification result, while the Enose has more sensors that are highly correlated with each other and redundant.

Since both the Enose and zNose™ have irrelevant and redundant information, simply adding the Enose and zNose™ raw data together in a feature-based fusion scheme does not improve classification performance. Instead, it worsened the classification performance, e.g. the 32.5% error rate from the fused data is higher than 15% from the Enose and 23% from the zNose™ individually.

In feature-based fusion scheme 2, using separately selected sensors and fusing them together (using seven sensors from the Enose and 32 windows from the zNose™) greatly improved the performance, reducing the classification error rate from 32.5% to 6.1%. Nevertheless, this improvement does not make sensor fusion superior than using the Enose or zNose™ alone. Both sensors achieved a lower classification error rate (4.2% and 2.6% respectively) than the “static selective sensor fusion scheme” if only the most relevant features selected by the CMAES were used.

In feature-based fusion scheme 3, 32 internal sensors and 64 chromatograph windows were considered and selected simultaneously by the CMAES, and a minimum classification error rate of 0 was achieved. This result is better than using either of these two instruments by themselves. The “data complementary effect” is the contributor to this improvement. In scheme 2, although optimized sensors were selected from the
Enose and zNose™ separately and they worked well individually, these sensors may not have provided complementary information when they were put together. In contrast, the dynamic selective sensor fusion selected only the sensors providing complementary information and consequently resulted in a better classification performance than using the Enose or zNose™ alone. The average classification error rate using the dynamic selective fusion scheme over 30 independent runs was 1.7%, and the best-ever-found classification error rate was 0%, which is better than the best results obtained from the two instruments individually.

At the decision level, a Bayesian network fusion was used to process data from PNN classifiers and BP classifiers. Using the hard evidence from PNN and prior knowledge of the Enose and zNose™ performance gave results only as good as one of sensors, but not better. This is because when decisions from two classifiers are in conflict, e.g., (1,0) and (0,1), they do not provide any useful information for making correct decisions. Adding prior performance knowledge helps to make the decision, but at the same time it gives bias to a particular sensor and only one sensor’s performance can be trusted. Thus, the performance can only be as good as one of sensors. However, the Bayesian network fusion model achieved better results when the soft evidence from the BP network was utilized whether or not prior performance knowledge was used. This is because soft evidence provides probability information that one sample belongs to a certain class. Generally, when more information is used, better results are achieved.

Comparing the feature level and decision level data fusion models, the feature level data fusion achieved better performance (minimum 1.5% error rate) than the decision level data fusion (minimum 3.5% for PNN fusion and 11% for BP fusion). The
feature level fusion performed better than individual sensors in the dynamic selective fusion, but not in non-selective and static selective fusion. The decision level fusion’s performance depends on the performances of the two sensors. By using soft evidence from BP classifiers, Bayesian network fusion can improve the individual sensors’ performance by 2%. These results supported the claim (Hall and McMullen, 2004) that generally better accuracy is obtained by fusing information closer to the source. With higher level fusion more information is lost, although at lower level fusion, more noise would be added to the model. Using and optimization method such as evolutionary strategy to select the most relevant features and remove redundant sensors and noise is a good choice for feature level fusion.

8.6 Conclusions

In this project, different levels of multisensor data fusion models, used to combine the Enose and zNose™ for apple defect detection, were developed and compared. In the feature level fusion, the covariance matrix adaptation evolutionary strategy, an optimization algorithm, was incorporated into the fusion process and used for feature selection. Two real number coding methods were constructed and compared. Based on whether or not the feature selection was carried out, three feature-based fusion schemes were developed and compared. Computational experiments showed that 48-variable coding performed slightly better than 24-variable coding in the CMAES optimization when search quality, search efficiency, and dimensionality reduction were considered. Among three feature level fusion schemes, the dynamic selective fusion outperformed the other two schemes, with a best case of 0% and average performance of 1.8%
classification error rate in total 30 independent runs. This is because the dynamic selective fusion method selected sensors with complementary information from the two instruments, while the static selective fusion method used only selected sensors that worked best for the individual instruments but not for the combined sensors. In the decision level fusion, Bayesian network fusion using soft evidence from the BP network improved individual sensor’s performance by 2%, while using hard evidence and prior performance only gave results as good as one of the two instruments. The dynamic selective fusion model provided the best performance, which supports the claim that in multisensor data fusion, generally, a better result is obtained at lower level data fusion which keeps much more original information.
CHAPTER 9
MODEL VALIDATION

Willing is not enough; we must do. Knowing is not enough; we must apply.

*Bruce Lee (1940-1973)*

The calibration models established in previous chapters performed well using data collected in the year 2005. To validate these sensor fusion models, i.e., testing on unseen data sets, new samples were taken in 2006. For this experiment, one bad apple was placed amongst three good apples. Calibration models for individual instruments and multisensor data fusion developed using data from the 2005 samples were applied to the new measurements. The purpose of this validation study was to simulate a more realistic situation, such as what might be found in a produce department at a grocery store. A certain number of spoiled apples may be hidden in a group of good apples in a plastic bag, which is impossible for vision or other techniques to detect.

9.1 Materials and methods

The experiments were conducted in June and July 2006 to collect new apple headspace samples. One bad apple was placed amongst three good apples in a 4 L glass jar. The testing period was from day 6 to day 16 after the damaged apples were cut. In total, 98 samples were collected, 35 with four good apples and 63 with one bad apple amongst three good apples (Table 9.1).
Table 9.1 Validation data set collection

<table>
<thead>
<tr>
<th>Date</th>
<th>6/29</th>
<th>7/1</th>
<th>7/6</th>
<th>7/8</th>
<th>7/14</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>12</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>11</td>
<td>35</td>
</tr>
<tr>
<td>Bad</td>
<td>12</td>
<td>15</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>63</td>
</tr>
<tr>
<td>Total</td>
<td>24</td>
<td>15</td>
<td>24</td>
<td>12</td>
<td>24</td>
<td>98</td>
</tr>
</tbody>
</table>

Validation was conducted on calibration models for the Enose and zNose™ individually and for multisensor data fusion models at both the feature and decision levels. The PNN was used for feature level fusion (ANN fusion) and the BP network was used to produce real number classification outputs, which were then used for decision level fusion (Bayesian fusion). Both PNN and BP networks were trained using 456 data vectors collected in 2005, and validated using 98 data vectors collected in 2006. Validation was conducted in the following two ways:

1) Validation of individual sensor models: Enose PNN and BP models, zNose™ PNN and BP models;

2) Validation of multisensor data fusion models:
   2.1) Feature level fusion (ANN fusion) models;
   2.2) Decision level fusion (Bayesian fusion) models.

All models were tested in three scenarios by using different number of sensors: 1) using all sensors, 2) previously selected sensors (selected in Chapter 6 and 7), and 3) adaptively selected sensors using the new data set.

9.2 Validation results

PNN models for individual sensors and feature level data fusions were tested and the correct classification rates are listed in Table 9.2. BP network classification models
for individual sensors and decision level fusion using Bayesian network were performed and the correct classification rates are compared in Table 9.3.

“All sensors” means that 32 sensors from the Enose and 512 wavelength values from the zNose™ were used, without any dimensionality reduction; “previously selected sensors” refers to those sensors selected through evolutionary algorithms in chapter 6 and 7 using the data set from 2005; “adaptively selected sensors” means using evolutionary algorithms (CMAES) to adaptively select sensors and wavelength values from the Enose and zNose™ using the new data set.

Table 9.2. Comparison of PNN and feature level fusion models correct classification rate

<table>
<thead>
<tr>
<th></th>
<th>All sensors</th>
<th>Previously selected sensors</th>
<th>Adaptively selected sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual sensor models</td>
<td>Enose PNN 68%</td>
<td>68% a</td>
<td>81% b</td>
</tr>
<tr>
<td>Sensor fusion models</td>
<td>zNose PNN 73%</td>
<td>82% a</td>
<td>93% b</td>
</tr>
<tr>
<td></td>
<td>ANN 66%</td>
<td>81% c</td>
<td>97% d</td>
</tr>
</tbody>
</table>

Table 9.3. Comparison of BP and Bayesian fusion models correct classification rate

<table>
<thead>
<tr>
<th></th>
<th>All sensors</th>
<th>Previously selected sensors</th>
<th>Adaptively selected sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual sensor models</td>
<td>Enose BP 79%</td>
<td>81% a</td>
<td>86% c</td>
</tr>
<tr>
<td>Sensor fusion models</td>
<td>zNose BP 80%</td>
<td>82% a</td>
<td>86% c</td>
</tr>
<tr>
<td></td>
<td>Bayesian 85% e</td>
<td>78% e</td>
<td>91% e</td>
</tr>
<tr>
<td></td>
<td>Fusion 82% f</td>
<td>82% f</td>
<td>88% f</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80% g</td>
<td>82% g</td>
</tr>
</tbody>
</table>

Note: Please refer to Appendix G for detailed information about the selected sensors and wavelength windows from a to g.

a Previously selected sensors and wavelength windows (as shown in Appendix G)
b Adaptively selected sensors and wavelength windows for calibration models for Enose and zNose™ individually (as shown in Appendix G)
c Previously selected sensors for feature level (ANN) fusion model (as shown in Appendix G)
d Adaptively selected sensors and wavelength windows for feature level (ANN) fusion model (as shown in Appendix G)
e Soft evidence with prior knowledge (prior knowledge was obtained in Chapter 8)
f Soft evidence without prior knowledge
9.3 Observations and summary

From results above, the following observations are made:

1) Looking across the rows in Tables 9.2 and 9.3, when different data dimensionality schemes were applied for different calibration models, it was found that adaptively selected sensors gave significantly improved performance at both feature level fusion (97%) and decision level fusion (91%); using previously selected sensors gave improved results for feature level fusion (81% vs. 66%) and comparable results for decision level fusion (82% vs. 85%).

2) Looking down the columns in Tables 9.2 and 9.3, when calibration models for individual sensors and multisensor data fusion were compared, Bayesian fusion at the decision level gave consistently superior performances compared to PNN and BP models for the Enose and zNose™ individually; however, ANN fusion models at the feature level only outperformed individual sensor models in the “adaptively selected sensors” scheme, not in the other two schemes. These results support the claim made in Chapter 8 that simply adding all Enose and zNose™ sensors together in an ANN fusion model increases more noise in the data and does not yield better results. Using “previously selected sensors”, the ANN fusion model gave a better result (81%) than using the Enose alone (68%) but comparable to the zNose™ PNN model (82%). This discrepancy may be caused by sensor drift and sample variation: those previously selected sensors may not provide complementary information when they experience drift and a new data set was tested. However, by using adaptively selected sensors,
the ANN fusion model gave a correct classification rate of 97%, better than any of the individual sensors (81% for Enose and 93% for zNose™). This indicates that the CMAES algorithm can adaptively select the most relevant sensors from a new data set and correct the sensor drift to some extent.

3) It was found that the zNose™ consistently outperformed the Enose in all test cases, although this was not true when calibration models were trained and tested using the data from the same calendar year. It suggests that the zNose™, which has a standard calibration method, is more stable and robust than the Enose, which was not calibrated during sampling.

4) Other possible error sources include training and testing samples that were collected from two different seasons with different harvesting time. This may influence their volatile profiles. Furthermore, testing on one bad apple amongst three good apples is also more challenging than testing on individual apples, since the amount of volatiles emitted by three good apples may dominate volatiles emitted by one bad apple.

5) In Table 9.3, when three schemes of Bayesian fusion were compared, it is found that generally the first situation (soft evidence with prior knowledge of performance) gives the best results; the soft evidence without prior knowledge and hard evidence with prior knowledge gave the second and third best results. However, in the scheme of using “previously selected sensors”, the soft evidence without prior knowledge and hard evidence with prior knowledge gave better results than soft evidence with prior knowledge. This discrepancy indicates that Bayesian fusion may not perform desirably if the prior knowledge does not fit the current data well.
CHAPTER 10
CONCLUSIONS AND FUTURE RESEARCH SUGGESTIONS

The longest road in the world is the one between aspiration to achievement.

Anonymous

10.1 Conclusions

This study developed multisensor data fusion algorithms to integrate two artificial noses (the Enose and zNose™) for physically damaged and fungi diseased apple detection. Artificial neural networks (ANN) and evolutionary algorithms (EA) were applied for pattern classification and data dimensionality reduction, both of which improved the Enose and zNose™ performance. More specifically, the following conclusions regarding this project were reached:

1a) Gas chromatography and mass spectrometry (GC-MS) experiments found that initially, more volatile compounds were detected from healthy apples than damaged apples. However, more complicated volatile profiles showed in the damaged apples’ chromatograph 6-9 days after treatment. Volatile compound quantity from healthy apples did not vary much from day 2 to day 9, while it increased more than 120% for damaged apples from day 2 to day 9. Qualitatively, some volatile compounds that were found in the damaged apples’ headspace gas were not present in the healthy apples’ headspace gas. These results were also verified by relevant literature (Vikram et al., 2004).
1b) Preliminary experiments found that both the Enose and zNose sensors respond to volatile compounds emitted by apples and both the Enose and zNose’s responses to two classes of apples are statistically different (p-value<0.0001). Differences in physical damage had some effect on volatile compound emission: apples subjected to two or three cuts generated volatile profiles that were significantly different from healthy apples. However, varying the orientation of damaged apples did not generate significant differences in the volatile profile.

2a) Statistical models were developed for the Enose and zNose™ separately. The high dimensional data were first compressed by PCA or PLS, and classified by linear discriminant analysis (LDA) and canonical variance analysis (CVA). When data from individual days were analyzed, good results were generally obtained. PLS-LDA produced the best results with 96% classification accuracy using zNose™ data and 85% accuracy for the Enose data when only the March 2005 data were tested.

2b) Statistical models did not perform well when data sets collected from different months were pooled and tested. In order to solve this high non-linear problem, artificial neural networks (ANN)-based pattern classification models were developed for the Enose and zNose™, respectively. Three ANN models (back-propagation, probabilistic, and learning vector quantification networks) were compared based on their classification accuracy, sensitivity and specificity for both the Enose and zNose™. For the Enose data, the BP and PNN achieved comparable classification performance with 85.3% and 85% accuracy rate, which were better than the LVQ classification rate of 73.7%. For the zNose™ data, three ANN models had similar performance which was less favorable than the Enose, with classification rates of 77%, 76.8% and 74.3%. The principal components
which explained more than 0.5% variations were extracted from the Enose and zNose™ data and used as neural network inputs.

3a) The high dimensionality problem of both Enose and zNose™ data was investigated in chapter six and seven. Reducing this high dimensionality helps reduce variable numbers, improve classification accuracy, and reduce computation time and sensor cost as well. For the Enose data, various dimensionality reduction approaches were studied and compared. Although both statistical approaches (PCA loadings and F-values) and sequential forward/backward search methods could reduce data dimensionality by 22% and 72% and at the same time keep the classification error rate to 25% and 6.1%, they only searched limited feature space. Two heuristic optimization algorithms, genetic algorithms (GA) and the covariance matrix adaptation evolutionary strategy (CMAES), provided a more powerful method to search the whole feature space. Both of these evolutionary algorithms reduced data dimensionality by 78% and improved the Enose classification performance by 10% compared to using all sensors. It was also found that using more sensors does not guarantee better classification performance; the Enose 32 sensors are highly correlated with each other, the same classification accuracy can be achieved by using different sensors and their combinations.

3b) The zNose™ spectral data feature selection was investigated by three evolutionary algorithms: genetic algorithms (GA), CMAES and differential evolution (DE). The whole spectrum, which spans 512 wavelength values, was divided into different windows: 16, 32 and 64, which were treated as independent variables. Both CMAES and DE yielded the best prediction accuracy with 64 windows, and GA produced comparable results with 32 windows and 64 windows, which were better than results from 16
windows. It suggested that the finer the spectrum was discretized, the better the classification accuracy. Simulations showed that the CMAES is the most efficient search algorithm (consumes fewer function evaluations) with comparable searching quality as differential evolution. Further fine tuning the algorithms found that algorithms with a larger population size gave better search results. For CMAES, the average cost (error rate) for ten random seed runs was 0.0289 with the best search cost of 0.0263 for a population size of $2\lambda$. DE produced a slightly better search quality but at a much higher search efficiency cost. All three algorithms effectively reduced the zNose™ data dimensionality by 50%, which in turn reduces the computation time.

4a) Feature level and decision level multisensor data fusion models were developed to combine the Enose and zNose™ data, to achieve better performance on damaged apple classification compared to using single sensors alone. At the feature level, the probabilistic neural networks were used to fuse extracted principal components from the Enose and zNose™. Three feature-based fusion models were developed and compared. The dynamic selective fusion achieved an average error rate of 1.8% and a best error rate of 0% in 30 independent runs. The static selective sensor fusion gave a 6.1% error rate which was not as good as using individual sensors (4.2% for the Enose and 2.6% for the zNose™) when both sensors used only selected features. In the non-selective fusion model, simply adding the Enose and zNose™ data without selection worsened the classification performance with a 32.5% error rate, which indicated that the feature selection using the CMAES is an indispensable step in ANN-based multi-sensor data fusion, especially when multiple sources of sensors contain irrelevant or redundant information.
4b) At the decision level, classification decisions made by the Enose and zNose™ independently were combined by Bayesian inference. Different use of the hard evidence (binary number classification results), soft evidence (real number probabilistic output) and prior performance knowledge (Enose and zNose™ classifiers’ prior performances) were investigated. Bayesian network fusion achieved better performance than individual sensors with the maximum 9% improvement of the error rate when soft evidence from the BP network was used with or without using prior performance knowledge. However, the Bayesian network did not produce better results than two sensors when hard evidence and prior knowledge were used. It was proven that both the feature level fusion with the CMAES optimization algorithms and decision level fusion with soft evidence improved system classification performance.

5a) Trained models were tested on new data sets, which were collected by measuring one bad amongst three healthy apples in a 4 L concentration chamber. Sensor fusion models could achieve 81% and 82% classification accuracy at the feature level and decision level; when selected sensors were updated, the classification accuracy of sensor fusion models could be improved to 97% at the feature level and 91% at the decision level.

5b) This research developed a system for non-destructive detection of fruit quality using two volatile detection instruments. It provides a methodology of using multiple sensors to detect fruit quality and safety when these defects are invisible or infected fruits are in a confined storage room where the application of vision and other technologies are impossible. This technology has shown promise to reduce fruit losses in fresh produce department and protect consumers’ interests as well.
10.2 Future research suggestions

Due to the time limit of the Ph.D. program and the scope limit of this dissertation, several issues that were not addressed in this thesis are expected to be explored in the future research:

1) An effective calibration method needs to be designed for the electronic nose in order to counter the sensor drift effect and generate repeatable results.

2) Future research is expected to test other concentration chambers such as a plastic bag, which is usually used to contain apples in a supermarket. It is ideal if the apple headspace gas could be collected without using a concentration chamber, which is a more realistic scenario in the real world.

3) More needs to be learned about physical properties of the Cyranose 320 internal 32 sensors so that specific sensors can be chosen that are sensitive to the key volatiles, and compare this method with the heuristic optimization/search method.

4) A GUI (graphic user interface) can be designed to integrate all functions from data collections, data preprocessing, pattern classification, feature selection to sensor fusion. These algorithms can also be integrated into commercial artificial noses.

5) In this thesis, only classification error rate was chosen as the single objective to optimize. In future research, different sensors’ costs can also be considered as another objective, which can be optimized by using evolutionary multi-objective optimization.

6) In the future, this system can be applied together with the analytical techniques which give more specificity.
7) The application of this system and methodology can be extended to other fruit cultivars and fruit surface pathogen detection in the future.
REFERENCES


Li, C., and P. Heinemann. 2006a. ANN integrated electronic nose system for apple quality evaluation. *Transactions of the ASABE (submitted for publication).*

Li, C., and P. Heinemann. 2006b. A comparative study of three evolutionary algorithms for a surface acoustic wave sensor wavelength selection. *Sensors and actuators B: Chemical (submitted for publication).*


Li, C., P. Heinemann, and P. Reed. 2006. Using genetic algorithms (GAs) and CMA evolutionary strategy to optimize electronic nose sensor selection. *Transactions of the ASABE (submitted for publication).*


Mielle, P. 1996. 'Electronic noses': Towards the objective instrumental characterization of food aroma. *Trends in Food Science & Technology* 7(12): 432.


NE-179 Project report. 2001. Assuring fruit and vegetable product quality and safety through the handling and marketing chain.


Waltz, E. "The principles and practice of image and spatial data fusion." *Proc. 8th Natl. Data Fusion Conf.*, Dallas, TX.


% this program is to build a BP network for Enose data classification
% 10/10/2005

clear all %clear previous variables;
%load data for input and target;
load eall.txt;
load target.txt;

%row is variables; column is repetition;
p=eall';
t=target';

%data preprocessing using PCA;
[pn, minp, maxp]=premnmx(p);
[ptrans, transMat]=prepca(pn, 0.0005);

%check processed matrix size;
ptrans=pn;
tn=t;
[R, Q]=size(ptrans)

%divide data into three sets: training, validation, and testing;
iitst=2:4:Q;
iival=4:4:Q;
iitr=[1:4:Q 3:4:Q];
val.P=ptrans(:,iival);val.T=tn(:,iival);
test.P=ptrans(:,iitst);test.T=tn(:,iitst);
ptr=ptrans(:,iitr);ttr=tn(:,iitr);

%build up a feed forward BP network, structure: R-10-2; training function:
%trainlm
net=newff(minmax(ptr), [21 2], {'tansig' 'purelin'}, 'trainrp');

%set training parameters: epochs and show how many times after
net.trainParam.show=10;
net.trainParam.epochs=200;
net.trainParam.goal=0.001;
net.performFcn='mse';
net=init(net); %initialize networks each time;
%train networks; with validation and test data sets;
tic
[n, tr, Y1, E1] = train(net, ptr, ttr, [], [], val, test);
toc %get how long training takes

%classification accuracy statistics
E1_row1=E1(1,:);
E1_round=round(E1_row1);
num_error1=sum(abs(E1_round))
correct_rate1=1-4*num_error1/Q
% simulate networks with testing data set;
[Y2,Pf,Af,E2,perf]=sim(net,test.P);
error=Y2-test.T;

% error statistics
E2_row1=error(1,:);
E2_row2=error(2,:);
E2_round1=round(E2_row1);
E2_round2=round(E2_row2);
E2_num_error1=sum(abs(E2_round1));
E2_num_error2=sum(abs(E2_round2));
E2_correctrate1=1-E2_num_error1/Q
E2_correctrate2=1-E2_num_error2/Q

% training and validation plot
plot(tr.epoch,tr.perf,tr.epoch,tr.vperf)
legend('Training','Validation');
ylabel('Squared error');xlabel('Epoch');

% this program is to build a LVQ network to classify the Enose data
% 10/2005

clear all % clear previous variables;
% load data for input and target;
load eall.txt;
load targetlvq.txt;

% row is input; column is repetition;
p=eall';
c=targetlvq';
t=ind2vec(c);

% data preprocessing using PCA;
[pn, minp, maxp]=premnmx(p);
[ptrans, transMat]=prepca(pn,0.005);

% check processed matrix size;
R,Q=size(ptrans);

divide data into four folds

% fold 1
iitst=1:4:Q;

% fold 2
iitst=2:4:Q;
iitr=[1:4:Q 3:4:Q 4:4:Q];

% fold 3
iitst=3:4:Q;
iitr=[1:4:Q 2:4:Q 4:4:Q];

% fold 4
iitst=4:4:Q;
iitr=[1:4:Q 2:4:Q 3:4:Q];
% % fold4
% iitst=4:4:Q;
% iiitr=[1:4:Q 2:4:Q 3:4:Q];

% divide data into three sets: training, and testing sets;
test.P=ptrans(:,iitst);test.T=tn(:,iitst);
ptr=ptrans(:,iiitr);ttr=tn(:,iiitr);

% build up a LVQ network
net=newlvq(minmax(ptr),30,[0.5 0.5],0.05);

% set training parameters: epochs and show how many times after
net.trainParam.show=25;
net.trainParam.epochs=500;
net.trainParam.goal=0.1;
net.performFcn='mse';

% 10 times independent runs
for pass=1:10
    pass=pass
    net=init(net); % initialize networks each time;
    net=train(net,ptr,ttr);
    Y1=sim(net,test.P);
    Yc=vec2ind(Y1);
    error=Y1-test.T;

    [R1,Q1]=size(test.T);
    all_error(pass)=sum(abs(round(error(1,:))));
    good_error(pass)=sum(abs(round(error(1,1:Q1/2))));
    bad_error(pass)=sum(abs(round(error(1,Q1/2+1:Q1))));

    vmse(pass)=sum(error(1,:).^2)/Q1
end

% calculate average error rate from 10 runs
ave_all_error=sum(all_error)/10
ave_good_error=sum(good_error)/10
ave_bad_error=sum(bad_error)/10
ave_vmse=sum(vmse)/10

% this program is to develop a PNN network for Enose data classification
% 10/2005
clear all % clear previous variables;
load data for input and target;
load eall.txt;
load target.txt;

% row is input; column is repetition;
p=eall';
t=target';
%data preprocessing using PCA;
[pn, ps]=mapminmax(p);
[ptrans, transMat]=processpca(pn,0.005);

%check processed matrix size;
R=;
[R, Q]=size(ptrans)

%divide data into four folds
%fold 1
iitst=1:4:Q;

% fold2
% iitst=2:4:Q;
% iitr=[1:4:Q 3:4:Q 4:4:Q];

% fold3
% iitst=3:4:Q;
% iitr=[1:4:Q 2:4:Q 4:4:Q];

% fold4
% iitst=4:4:Q;
% iitr=[1:4:Q 2:4:Q 3:4:Q];

%divide data into training and testing set;
test.P=ptrans(:,iitst);test.T=tn(:,iitst);
ptr=ptrans(:,iitr);ttr=tn(:,iitr);

%build up a PNN network
net=newpnn(ptr, ttr);

n=5; % 5 independent runs
for pass=1:n
    randn('state', sum(100*clock))
    net=init(net);
    Y1=sim(net, ptr);
    train_error=Y1-ttr;

    [R0, Q0]=size(ttr);
    train_all_error(pass)=sum(abs(round(train_error(1,:))))
    train_good_error(pass)=sum(abs(round(train_error(1,1:Q0/2))))
    train_bad_error(pass)=sum(abs(round(train_error(1,Q0/2+1:Q0))))
    train_vmse(pass)=sum(train_error(1,:).^2)/Q0

    Y2=sim(net, test.P);
    test_error=Y2-test.T;

    [R1, Q1]=size(test.P);
    test_all_error(pass)=sum(abs(round(test_error(1,:))))
    test_good_error(pass)=sum(abs(round(test_error(1,1:Q1/2))))
    test_bad_error(pass)=sum(abs(round(test_error(1,Q1/2+1:Q1))))
    test_vmse(pass)=sum(test_error(1,:).^2)/Q1

265
end

test_test_ave_all_error=sum(test_all_error)/n
test_ave_good_error=sum(test_good_error)/n
test_ave_bad_error=sum(test_bad_error)/n
test_ave_vmse=sum(test_vmse)/n

%plot real output and target comparison plot;
time = 1:length(test.P);
plot(time,test.T(1,:),'--',time,Y2(1,:))
axis([0 length(time) -0.5 1.5])
title('Classification performance comparison')
xlabel('Time Step')
ylabel('Target -- Output ---')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% this program is to design a BP network for zNose data processing;
% 10/2005
% clear all %clear previous variables;
% load data for input and target;
load zall.txt;
load ztarget3.txt;

% row is input;column is repetition;
p=zall';
t=ztarget3';

%-----use all sensors but 8 windows that were statistically selected
% only once or none;
b1=randperm(64);
b1=sort(b1);
b1(2)=[];b1(24)=[];b1(25)=[];b1(35)=[];b1(36)=[];b1(38)=[];b1(41)=[];b1(44)=[];
zall_new=[];
nsensor=length(b1);
for j=1:nsensor
zall_new=cat(1,zall_new,p(((b1(j)-1)*Nt+1):b1(j)*Nt,:));%Nt=32, then 16
windows, otherwise...
end
p=zall_new;

%data preprocessing using PCA;
[pn, minp, maxp]=premnmx(p);
[ptrans,transMat]=prepca(pn,0.005);

%check processed matrix size;
 tn=t;
 [R,Q]=size(pn);

 % fold 1
 iival=1:4:Q;
val.P=ptrans(:,iival);val.T=tn(:,iival);
ptr=ptrans(:,iitr);ttr=tn(:,iitr);

% build up a feed forward BP network, structure: R-10-2; training function: trainlm
net=newff(minmax(ptr),[25 2],{'tansig' 'logsig'},'trainrp');

% set training parameters: epochs and show how many times after
net.trainParam.show=10;
net.trainParam.epochs=1000;
net.trainParam.goal=0.001;
net.performFcn='mse';

% train and test networks for 20 independent runs;
for pass=1:20
    net=init(net); % initialize networks each time;
    [net,Tr,Y1,E1]=train(net,ptr,ttr,[],[],val);
    plot(Tr.epoch,Tr.perf,Tr.epoch,Tr.vperf)
    legend('Traning','Validation',-1);
    ylabel('Squared error');xlabel('Epoch');
    Y2=sim(net,val.P);
    error=Y2-val.T;
    [R1,Q1]=size(val.P);
    all_error(pass)=sum(abs(round(error(1,:))))
    good_error(pass)=sum(abs(round(error(1,1:Q1/2))))
    bad_error(pass)=sum(abs(round(error(1,Q1/2+1:Q1))))
    vmse(pass)=sum(error(1,:).^2)/Q1
end

% average error rate statistics
ave_all_error=sum(all_error)/pass
ave_good_error=sum(good_error)/pass
ave_bad_error=sum(bad_error)/pass
ave_vmse=sum(vmse)/pass

% plot real output and target comparison plot;
time = 1:length(val.P);
plot(time,val.T(1,:),'--',time,Y2(1,:))
axis([0 length(time) -0.5 1.5])
title('BP classification performance comparison')
xlabel('Time Step')
ylabel('Target -- Output ---')
% this program is to develop a LVQ network for zNose data processing
% 10/2005
clear all %clear previous variables;
%load data for input and target;
load zall.txt;
load ztarget3.txt;
%row is input; column is repetition;
p=zall';
t=ztarget3';

%------use all sensors but 8 windows that were statistically selected
only once or none;
b1=randperm(64);
b1=sort(b1);
b1(2)=[];b1(24)=[];b1(25)=[];b1(35)=[];b1(36)=[];b1(38)=[];b1(41)=[];
b1(44)=[];
zall_new=[];
nsensor=length(b1);
for j=1:nsensor
    zall_new=cat(1,zall_new,p(((b1(j)-1)*Nt+1):b1(j)*Nt,:));
    %Nt=32, then 16 windows, otherwise...
end
p=zall_new;

%data preprocessing using PCA;
[pn, meanp, stdp, tn, meant, stdt]=prestd(p,t);
[ptrans, minp, maxp]=premnmx(pn);

%check processed matrix size;

%divide data into four folds
% fold1
iitst=1:4:Q;

% fold2
iitst=2:4:Q;
iitr=[1:4:Q 3:4:Q 4:4:Q];

% fold3
iitst=3:4:Q;
iitr=[1:4:Q 2:4:Q 4:4:Q];

% fold4
iitst=4:4:Q;
iitr=[1:4:Q 2:4:Q 3:4:Q];
%divide data into training and testing sets;
test.P=ptrans(:,iitst);test.T=tn(:,iitst);
ptr=ptrans(:,iitr);ttr=tn(:,iitr);

%build up a new LVQ network
net=newlvq(minmax(ptr),30,[0.5 0.5],0.05);

%set traning parameters: epochs and show how many times after
net.trainParam.show=25;
net.trainParam.epochs=80;
net.trainParam.goal=0.1;
net.performFcn='mse';

for pass=1:10  % test for 10 independent runs;
    pass=pass
    net=init(net);%initialize networks each time;
    net=train(net,ptr,ttr);
    Y1=sim(net,test.P);
    error=Y1-test.T;

    [R1,Q1]=size(test.T);
    all_error(pass)=sum(abs(round(error(1,:))))
    good_error(pass)=sum(abs(round(error(1,1:Q1/2))))
    bad_error(pass)=sum(abs(round(error(1,Q1/2+1:Q1))))
    vmse(pass)=sum(error(1,:).^2)/Q1
end

% average error rate statistics
ave_all_error=sum(all_error)/pass
ave_good_error=sum(good_error)/pass
ave_bad_error=sum(bad_error)/pass
ave_vmse=sum(vmse)/pass

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% this program is to develop a PNN for zNose data processing
% 10/2005
clear all %clear previous variables;
%load data for input and target;
load zall.txt;
load ztarget3.txt;

%row is input;column is repetition;
p=zall';
t=ztarget3';

%------use all sensors but 8 windows that were statistically selected
only once or none;
b1=randperm(64);
b1=sort(b1);
b1(2)=[];b1(24)=[];b1(25)=[];b1(35)=[];b1(36)=[];b1(38)=[];b1(41)=[];b1 (44)=[];
zall_new=[];
nsensor=length(b1);
for j=1:nsensor
zall_new=cat(1,zall_new,p(((b1(j)-1)*Nt+1):b1(j)*Nt,:));%Nt=32, then 16 windows, otherwise...
end
p=zall_new;
%data preprocessing using PCA;
[pn, meanp, stdp, tn, meant, stdt]=prestd(p, t);
[ptrans, minp, maxp]=premmx(pn, 0.005);

%check processed matrix size;

tn=t;
[R, Q]=size(ptrans)

%divide data into four folds
%fold 1
iitst=1:4:Q;
%
%fold2
% iitst=2:4:Q;
% iitr=[1:4:Q 3:4:Q 4:4:Q];
%
%fold3
% iitst=3:4:Q;
% iitr=[1:4:Q 2:4:Q 4:4:Q];
%
%fold4
% iitst=4:4:Q;
% iitr=[1:4:Q 2:4:Q 3:4:Q];

%divide data into training and testing sets;
test.P=ptrans(:, iitst); test.T=tn(:, iitst);
ptr=ptrans(:, iitr); ttr=tn(:, iitr);

%build up a PNN
net=newpnn(ptr, ttr); %
% ten independent runs;
for pass=1:10
  net=init(net);
  Y1=sim(net, ptr);
  train_error=Y1-ttr;
  [R0, Q0]=size(ttr);
  train_all_error(pass)=sum(abs(round(train_error(1,:),1)))
  train_good_error(pass)=sum(abs(round(train_error(1,1:Q0/2,:)),1))
  train_bad_error(pass)=sum(abs(round(train_error(1,Q0/2+1:Q0,:)),1))
  train_vmse(pass)=sum(train_error(1,:),1)^2)/Q0
  Y2=sim(net, test.P);
  test_error=Y2-test.T;
  [R1, Q1]=size(test.P);
  test_all_error(pass)=sum(abs(round(test_error(1,:),1)))
  test_good_error(pass)=sum(abs(round(test_error(1,1:Q1/2,:)),1))
  test_bad_error(pass)=sum(abs(round(test_error(1,Q1/2+1:Q1,:)),1))
end
test_bad_error(pass)=sum(abs(round(test_error(1,Q1/2+1:Q1))))
end

test_vmse(pass)=sum(test_error(1,:).^2)/Q1

% average error rate statistics;
test_test_ave_all_error=sum(test_all_error)/10
test_ave_good_error=sum(test_good_error)/10
test_ave_bad_error=sum(test_bad_error)/10
test_ave_vmse=sum(test_vmse)/10
Appendix B

% this program is a binary genetic algorithms for zNose windows
% selection;
% Changying, 2/1/2006;

clear
for repeat=1:10 %compute 10 times independently;

% ________________________________________________________________
%                       I. Setup the GA
ff='zpnn';      % objective function
npar=32;            % number of optimization variables
% ________________________________________________________________
%                     II. Stopping criteria
maxit=200;                  % max number of iterations
mincost=3;          % minimum cost
% ________________________________________________________________
%                       III. GA parameters
popsize=20;                % set population size
mutrate=.15;               % set mutation rate
selection=0.5;             % fraction of population kept
nbits=1;                         % number of bits in each parameter
Nt=nbits*npar;                   % total number of bits in a chromosome
keep=floor(selection*popsize);   % #population members that survive
% _____________________________________________________________
%               Create the initial population
iga=0;                  % generation counter initialized
pop=round(rand(popsize,Nt));        % random population of 1s and 0s
tic
cost=feval(ff,popsize,pop);               % calculates population cost
using ff
toc

[cost,ind]=sort(cost);             % min cost in element 1
pop=pop(ind,:);  % sorts population with lowest cost first
minc(l)=min(cost);            % minc contains min of population
meanc(l)=mean(cost);            % meanc contains mean of population
% _____________________________________________________________
%               Iterate through generations
while iga<maxit
    iga=iga+1;                        % increments generation counter
% _____________________________________________________________
%               Pair and mate
M=ceil((popsize-keep)/2);            % number of matings
prob=flipud([1:keep]'/sum([1:keep]));   % weights chromosomes based
upon position in list
odds=[0 cumsum(prob(1:keep))'];};  % probability distribution function
pick1=rand(1,M);                    % mate #1 ?
pick2=rand(1,M);                    % mate #2 ?
%% ma and pa contain the indicies of the chromosomes that will mate
ic=1;
while ic<=M
    for id=2:keep+1
        if pick1(ic)<=odds(id) & pick1(ic)>odds(id-1)
            ma(ic)=id-1;
            end % if
        if pick2(ic)<=odds(id) & pick2(ic)>odds(id-1)
            pa(ic)=id-1;
            end % if
        end % id
    ic=ic+1;
end % while

%% _______________________________________________________________________
% Performs mating using single point crossover
ix=1:2:keep;                                          % index of mate #1
xp=ceil(rand(1,M)*(Nt-1));                           % crossover point
pop(keep+ix,:)=[pop(ma,1:xp) pop(pa,xp+1:Nt)];        % first offspring
pop(keep+ix+1,:)=[pop(pa,1:xp) pop(ma,xp+1:Nt)];      % second offspring

%% _______________________________________________________________________
% Mutate the population
nmut=ceil((popsize-1)*Nt*mutrate);                   % total number of mutations
mrow=ceil(rand(1,nmut)*(popsize-1))+1;               % row to mutate
mcol=ceil(rand(1,nmut)*Nt);                          % column to mutate
for ii=1:nmut
    pop(mrow(ii),mcol(ii))=abs(pop(mrow(ii),mcol(ii))-1); % toggles
end % ii

%% _______________________________________________________________________
% The population is re-evaluated for cost
tic
cost(2:popsize)=feval(ff,popsize-1,pop(2:popsize,:));% from no.2, not all pop;
toc

%% _______________________________________________________________________
% Sort the costs and associated parameters
[cost,ind]=sort(cost);
pop=pop(ind,:);

%% _______________________________________________________________________
% Do statistics for a single nonaveraging run
minc(iga+1)=min(cost);
meanc(iga+1)=mean(cost);

%% _______________________________________________________________________
% Stopping criteria
if iga>maxit | cost(1)<mincost
    break
end

[iga cost(1)]
end iga

% Displays the output
day=clock;
disp(datestr(datenum(day(1),day(2),day(3),day(4),day(5),day(6)),0))
disp(['optimized function is ' ff])
format short

disp(['popsize = ' num2str(popsize) ' mutrate = ' num2str(mutrate) ' #
par = ' num2str(npar)])
disp(['#generations= ' num2str(iga) ' best cost= ' num2str(cost(1))])
disp(['best solution'])
disp([num2str(pop(1,:))])
disp('binary genetic algorithm')
disp(['each parameter represented by ' num2str(nbits) ' bits'])
figure(24)
iters=0:length(minc)-1;
plot(iters,minc,iters,meanc,'--');
xlabel('generation');ylabel('cost');
text(0,minc(1),'best');text(1,minc(2),'population average')

% count how many sensors are selected; and what position they are;
best_windows=pop(1,:);
[best_windows_sorted ind]=sort(best_windows);
sensor_number=sum(best_windows);

disp(['the number of selected wavelengths in repeat:' num2str(repeat)])
sensor_number
disp('the selected wavelengths are:')
b=ind(Nt-sensor_number+1:Nt) %Nt is the length of the chromosome;

%this is the repeat of 'num2str(repeat))

fid=fopen(['znoserun' num2str(repeat) '.txt'],'wt');
fprintf(fid,'% 6.2f %12.8f %6.2f 
',b,iga,cost(1));
close(fid);
end repeat

%this is the function to calculate cost for GA using PNN classifier;
%Binary code;
%zNose data;
%2/10/06

function test_all_error=zpnn(pnum,pop)
[m,n]=size(pop);
Nt=512/n;

%load zall matrix;and target matrix;
load zallraw.txt;
load ztarget3.txt;
zall=zallraw;
t=ztarg3;'

% manipulate sensors/turn on selected sensors; e.g., x=0000...0001, turn on
% the first sensor

[pop1 ind]=sort(pop,2); % sort pop two dimensions;
nsensor=sum(pop,2); % count how many sensors are turned on;
[row column]=size(pop);
% calculate costs for pnum schemes
for i=1:pnum;
    z=[]; % initialize z matrix each time;
a=ind(i,column-nsensor(i)+1:column);
    %---force wavelength points be less than 456;
    if column==16
      if nsensor>=15
        nsensor=14;
      end
    elseif column==32
      if nsensor>=29
        nsensor=28;
      end
    elseif column==64
      if nsensor>=58
        nsensor=57;
      end
    end
    for j=1:nsensor(i)
        z=cat(1,z,zall(((a(j)-1)*Nt+1):a(j)*Nt,:));%Nt=32, then 16
        windows, otherwise...
        %Nt=16, 32 windows;
    end
  end
  %-----------------------------
  for j=1:n sensor(i)
    z=cat(1,z,zall(((a(j)-1)*Nt+1):a(j)*Nt,:)));%Nt=32, then 16
    windows, otherwise...
    %Nt=16,32 windows;
  end
  
  %-----------------------------
  % data preprocessing using PCA; -- can i not use pca, while to use
  raw
  % spectrum data directly?
  [pn, ps]=mapminmax(z);
  [ptrans,transMat]=processpca(pn,0.005);
  
  tn=t;
  [R,Q]=size(ptrans);

  % extract training data set and testing data set;
  iitst=1:4:Q;

  % divide data into two sets: training and testing;
  test.P=ptrans(:,iitst);test.T=tn(:,iitst);
  ptr=ptrans(:,iitr);ttr=tn(:,iitr);

  % build up new PNN
net=newpnn(ptr,ttr);

net=init(net);
%simulate using training set
Y1=sim(net,ptr);
train_error=Y1-ttr;

%simulate using testing data;
Y2=sim(net,test.P);
test_error=Y2-test.T;

%count error numbers;
test_all_error(i)=sum(abs(round(test_error(1,:))));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%this program is to run CMAES for real number coding; 16 digits; each
represent 2
% sensors;
% 4/2006
clear;
for repeat=1:10  % 10 independent runs;
opts.StopFitness=0; % stop criteria
opts.LBounds = 0; opts.UBounds = 1;

lambda=24  % set population size;
opts.PopSize=lambda;

fitfun='epnn6';
xstart=rand(16,1);% set xstart from range -1 to 1;

insigma=0.5;

[xmin,fmin,counteval,stopflag]=cmaes07(fitfun,xstart,insigma,opts)
disp(['this is the repeat: ' num2str(repeat)])
end % end repeat;

% [xmin, ... % minimum search point of last iteration
% fmin, ... % function value of xmin
% counteval, ... % number of function evaluations done
% stopflag, ... % stop criterion reached
% ] = cmaes( ...
% fitfun, ... % name of objective/fitness function
% xstart, ... % objective variables initial point, determines N
% insigma, ... % initial coordinate wise standard deviation(s)
% inopts) % options struct, see defopts below

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%this program is to calculate cost for CMAES using PNN classifier;
% real number coding
function test_all_error=epnn(x)
% x is a 32 x 1 vector; one population, we need to transpose it to 1 x 32
%%%%%-----------------
load eall.txt;
load zallraw.txt;
load etest100.txt;
load etrain98.txt;

ep=eall';
etest_new=etest100(:,[1:63 66:100]);
etrain_new=etrain98(:,[1:48 51:98]);

etarget(:,1)=cat(1,ones(228,1),zeros(228,1),ones(48,1),
zeros(48,1),ones(35,1),zeros(63,1));
etarget(:,2)=cat(1,zeros(228,1),ones(228,1),zeros(48,1),
one(48,1),zeros(35,1),ones(63,1));
eall=horzcat(ep,etrain_new,etest_new);  % append all data(2005, 2006 training and testing together)

%%%%%-----------------
%row is input;column is repetition;
p=eall;  % 32 x 650
t=etarget';% 2 x 650
x=x';
[row column]=size(x);
%row is population size 1; colum is chromosome length;

S_new=0;
%round x elements; to 1 or 0;
for m=1:column
    if x(m)<=0.25
        S=[0 0];
    else if x(m)>0.25 && x(m)<=0.5
        S=[1 0];
    else if x(m)>0.5 && x(m)<=0.75
        S=[0 1];
    else S=[1 1];
    end
end
S_new=[S_new S];
end

%manipulate sensors/turn on selected sensors; e.g., x=0000...0001, turn
%on the first sensor
x1=S_new(2:33);
[pop1 ind]=sort(x1,2); %sort x
nsensor=sum(x1,2);  % count how many sensors are turned on;

% caculate costs for pnum schemes
p=eall;  % 32 x 650 reload initial data;
a = ind(32 - nsensor + 1:32); % get the index number (position) of on sensors
p = p(a,:); % got new data matrix after select specific sensors;

% data preprocessing using PCA;
[pn, minp, maxp] = premnmx(p);
[ptrans, transMat] = prepca(pn, 0.005);

%ptrans=p;
tn=t;
[R, Q] = size(ptrans);

% divide data into training and testing sets;
iitr=1:456;
iitst=456+97:650;

% divide data into two sets: training and testing;
test.P = ptrans(:, iitst); test.T = tn(:, iitst);
ptr = ptrans(:, iitr); ttr = tn(:, iitr);

% build up new PNN
net = newpnn(ptr, ttr);

% randn('seed', 19273647);
net = init(net);

% simulate using training set
Y1 = sim(net, ptr);
train_error = Y1 - ttr;

% simulate using testing data;
Y2 = sim(net, test.P);
test_error = Y2 - test.T;

% count error numbers;
test_all_error = sum(abs(round(test_error(1,:))));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% this program is for enose data sensor selection using differential
% evolution optimizer (DE);
% 4/2006

clear;

% VTR        "Value To Reach" (stop when ofunc < VTR)
VTR = 0;

% D        number of parameters of the objective function
D = 32;

% XVmin, XVmax    vector of lower and bounds of initial population
% the algorithm seems to work well only if [XVmin, XVmax] covers the region where the global minimum is expected
% *** note: these are no bound constraints!! ***
XVmin = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
XVmax = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];

% y     problem data vector (remains fixed during optimization)
y=[];

% NP    number of population members: 10x D
NP = 320;

% itermax maximum number of iterations (generations)
itermax = 50;

% F     DE-stepsize F ex [0, 2]
F = 0.8;

% CR    crossover probability constant ex [0, 1]
CR = 1;

% strategy 1 --> DE/best/1/exp      6 --> DE/best/1/bin
%           2 --> DE/rand/1/exp      7 --> DE/rand/1/bin
%           3 --> DE/rand-to-best/1/exp  8 --> DE/rand-to-
% best/1/bin
%           4 --> DE/best/2/exp      9 --> DE/best/2/bin
%           5 --> DE/rand/2/exp      else DE/rand/2/bin

strategy = 7;

% refresh intermediate output will be produced after "refresh" iterations. No intermediate output will be produced if refresh is < 1
refresh = 10; % refresh every 10 iterations;

[x,f,nf] = devec3('epnn4',VTR,D,XVmin,XVmax,y,NP,itermax,F,CR,strategy,refresh)

runs=10; % number of independent runs;

for i=1:runs
    tic
    disp(['the run number is: ' num2str(i)]) % the number of independent runs;
    disp(['current time is: ' num2str(clock)])
    [x,f,nf] = devec3('epnn4',VTR,D,XVmin,XVmax,y,NP,itermax,F,CR,strategy,refresh)
    toc % calculate each run time;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% cost function for DE
% this is for Differential Evolution (DE) cost function

279
function test_all_error=epnn(x,y)
%load data for input and target;
load eall.txt;
load target.txt;

%row is input; column is repetition;
p=eall';
t=target';

[row colum]=size(x); %row is population size 1; column is chromosome length;

%round x elements; to 1 or 0;
for m=1:colum
    if x(m)>0.5
        x(m)=1;
    else
        x(m)=0;
    end
end

[popl ind]=sort(x,2); %sort x
nsensor=sum(x,2); %count how many sensors are turned on;

%calculate costs for pnum schemes
p=eall'; %reload initial data;
a=ind(colum-nsensor+1:colum); %get teh index number (position) of on sensors
p=p(a,:); %got new data matrix after select specific sensors;

%data preprocessing using PCA;
[pn, minp, maxp]=premnmmx(p);
[ptrans, transMat]=pre pca(pn,0.005);

%ptrans=p;
tn=t;
[R,Q]=size(ptrans);

%extract training and testing sets;
iitst=4:4:Q;
iitr=[1:4:Q 2:4:Q 3:4:Q];
test.P=ptrans(:,iitst); test.T=tn(:,iitst);
ptr=ptrans(:,iitr);ttr=tn(:,iitr);

%build up new PNN
net=newpnn(ptr,ttr);
    net=init(net);
%simulate using training set
Y1=sim(net,ptr);
train_error=Y1-ttr;

%simulate using testing data;
Y2=sim(net,test.P);
test_error=Y2-test.T;

test_all_error=sum(abs(round(test_error(1,:))));

% this program is DE optimizer
% modified from Rainer Storn;
% 4/2006
function [bestmem,bestval,nfeval] =
devect3(fname,VTR,D,XVmin,XVmax,y,NP,itermax,F,CR,strategy,refresh);
% Output arguments:
% ----------------
% bestmem parameter vector with best solution
% bestval best objective function value
% nfeval number of function evaluations
%
% Input arguments:
% ----------------
% fname string naming a function f(x,y) to minimize
% VTR "Value To Reach". devect3 will stop its minimization
% if either the maximum number of iterations "itermax"
% is reached or the best parameter vector "bestmem"
% has found a value f(bestmem,y) <= VTR.
% D number of parameters of the objective function
% XVmin vector of lower bounds XVmin(1) ... XVmin(D)
% of initial population
% *** note: these are not bound constraints!! ***
% XVmax vector of upper bounds XVmax(1) ... XVmax(D)
% of initial population
% y problem data vector (must remain fixed during the
% minimization)
% NP number of population members
% itermax maximum number of iterations (generations)
% F DE-stepsize F from interval [0, 2]
% CR crossover probability constant from interval [0, 1]
% strategy 1 --> DE/best/1/exp 6 --> DE/best/1/bin
% 2 --> DE/rand/1/exp 7 --> DE/rand/1/bin
% 3 --> DE/rand-to-best/1/exp 8 --> DE/rand-to-
% best/1/bin
% 4 --> DE/best/2/exp 9 --> DE/best/2/bin
% 5 --> DE/rand/2/exp else DE/rand/2/bin
% Experiments suggest that /bin likes to have a slightly
% larger CR than /exp.
% refresh intermediate output will be produced after "refresh"
% iterations. No intermediate output will be produced
% if refresh is < 1
% About devec3.m
% ----------------
% Differential Evolution for MATLAB
% Copyright (C) 1996, 1997 R. Storn
% International Computer Science Institute (ICSI)
% 1947 Center Street, Suite 600

281
% devec is a vectorized variant of DE which, however, has a
% property which differs from the original version of DE:
% 1) The random selection of vectors is performed by shuffling the
% population array. Hence a certain vector can't be chosen twice
% in the same term of the perturbation expression.

% Due to the vectorized expressions devec3 executes fairly fast
% in MATLAB's interpreter environment.

% This program is free software; you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 1, or (at your option)
% any later version.

% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
% GNU General Public License for more details. A copy of the GNU
% General Public License can be obtained from the
% Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139,
% USA.

%-----Check input variables--------------------------------------------
-
err=[];
if nargin<1, error('devec3 1st argument must be function name'); else
  if exist(fname)<1; err(1,length(err)+1)=1; end; end;
if nargin<2, VTR = 1.e-6; else
  if length(VTR)~=1; err(1,length(err)+1)=2; end; end;
if nargin<3, D = 2; else
  if length(D)~=1; err(1,length(err)+1)=3; end; end;
if nargin<4, XVmin = [-2 -2];else
  if length(XVmin)~=D; err(1,length(err)+1)=4; end; end;
if nargin<5, XVmax = [2 2]; else
  if length(XVmax)~=D; err(1,length(err)+1)=5; end; end;
if nargin<6, y=[];end;
if nargin<7, NP = 10*D; else
  if length(NP)~=1; err(1,length(err)+1)=7; end; end;
if nargin<8, itermax = 200; else
  if length(itermax)~=1; err(1,length(err)+1)=8; end; end;
if nargin<9, P = 0.8; else
  if length(P)~=1; err(1,length(err)+1)=9; end; end;
if nargin<10, CR = 0.5; else
  if length(CR)~=1; err(1,length(err)+1)=10; end; end;
if nargin<11, strategy = 7; else
  if length(strategy)~=1; err(1,length(err)+1)=11; end; end;
if nargin<12, refresh = 10; else
  if length(refresh)~=1; err(1,length(err)+1)=12; end; end;
if length(err)>0
  fprintf(stdout,'error in parameter %d\n', err);
usage('devec3
(string,scalar,scalar,vector,vector,any,integer,integer,scalar,scalar,integer,integer)');
end

if (NP < 5)
    NP=5;
    fprintf(1,' NP increased to minimal value 5\n');
end
if ((CR < 0) | (CR > 1))
    CR=0.5;
    fprintf(1,'CR should be from interval [0,1]; set to default value 0.5\n');
end
if (itermax <= 0)
    itermax = 200;
    fprintf(1,'itermax should be > 0; set to default value 200\n');
end
refresh = floor(refresh);

%-----Initialize population and some arrays-----------------------------

pop = zeros(NP,D); %initialize pop to gain speed

%----pop is a matrix of size NPxD. It will be initialized--------------
%----with random values between the min and max values of the---------
%----parameters--------------------------------------------------------

for i=1:NP
    pop(i,:) = XVmin + rand(1,D).*(XVmax - XVmin);
end

popold    = zeros(size(pop));     % toggle population
val       = zeros(1,NP);          % create and reset the "cost array"
bestmem   = zeros(1,D);           % best population member ever
bestmemit = zeros(1,D);           % best population member in iteration
nfeval    = 0;                    % number of function evaluations

%------Evaluate the best member after initialization-------------------

ibest   = 1;                      % start with first population member
val(1)  = feval(fname,pop(ibest,:),y);
%val(1)  = feval(fname,pop(ibest,:));
bestval = val(1);                 % best objective function value so
far
nfeval  = nfeval + 1;
for i=2:NP
    % check the remaining members
    val(i)  = feval(fname,pop(i,:),y);
%val(i)  = feval(fname,pop(i,:));
    nfeval  = nfeval + 1;
    if (val(i) < bestval)        % if member is better
        ibest = i;                % save its location
        bestval = val(i);
end
end
bestmemit = pop(ibest,:); % best member of current iteration
bestvalit = bestval; % best value of current iteration
bestmem = bestmemit; % best member ever

%------DE-Minimization---------------------------------------------
%------popold is the population which has to compete. It is------
%------static through one iteration. pop is the newly--------------
%------emerging population.----------------------------------------

pm1 = zeros(NP,D); % initialize population matrix 1
pm2 = zeros(NP,D); % initialize population matrix 2
pm3 = zeros(NP,D); % initialize population matrix 3
pm4 = zeros(NP,D); % initialize population matrix 4
pm5 = zeros(NP,D); % initialize population matrix 5
bm  = zeros(NP,D); % initialize bestmember matrix
ui  = zeros(NP,D); % intermediate population of perturbed vectors
mui = zeros(NP,D); % mask for intermediate population
mpo = zeros(NP,D); % mask for old population
rot = (0:1:NP-1); % rotating index array (size NP)
rotd= (0:1:D-1); % rotating index array (size D)
rt  = zeros(NP); % another rotating index array
crossover
rtd = zeros(D); % rotating index array for exponential crossover

a1  = zeros(NP); % index array
a2  = zeros(NP); % index array
a3  = zeros(NP); % index array
a4  = zeros(NP); % index array
a5  = zeros(NP); % index array
ind = zeros(4);
iter = 1;
while ((iter < itermax) & (bestval > VTR))
    popold = pop; % save the old population
    ind = randperm(4); % index pointer array
    a1  = randperm(NP); % shuffle locations of vectors
    rt  = rem(rot+ind(1),NP); % rotate indices by ind(1) positions
    a2  = a1(rt+1);
    rt  = rem(rot+ind(2),NP);
    a3  = a2(rt+1);
    rt  = rem(rot+ind(3),NP);
    a4  = a3(rt+1);
    rt  = rem(rot+ind(4),NP);
    a5  = a4(rt+1);

    pm1 = popold(a1,:); % shuffled population 1
    pm2 = popold(a2,:); % shuffled population 2
    pm3 = popold(a3,:); % shuffled population 3
    pm4 = popold(a4,:); % shuffled population 4
    pm5 = popold(a5,:); % shuffled population 5
end
for i=1:NP  % population filled with the best member
    bm(i,:) = bestmemit;  % of the last iteration
end

mui = rand(NP,D) < CR;  % all random numbers < CR are 1, 0

if (strategy > 5)
    st = strategy-5;  % binomial crossover
else
    st = strategy;  % exponential crossover
    mui=sort(mui');  % transpose, collect 1's in each column
    for i=1:NP
        n=floor(rand*D);
        if n > 0
            rtd = rem(rod+1,n);
            mui(:,i) = mui(rtd+1,i);  %rotate column i by n
        end
    end
    mui = mui';  % transpose back
end

mpo = mui < 0.5;  % inverse mask to mui

if (st == 1)  % DE/best/1
    ui = bm + F*(pm1 - pm2);  % differential variation
    ui = popold.*mpo + ui.*mui;  % crossover
elseif (st == 2)  % DE/rand/1
    ui = pm3 + F*(pm1 - pm2);  % differential variation
    ui = popold.*mpo + ui.*mui;  % crossover
elseif (st == 3)  % DE/rand-to-best/1
    ui = popold + F*(bm-popold) + F*(pm1 - pm2);  % differential variation
    ui = popold.*mpo + ui.*mui;  % crossover
elseif (st == 4)  % DE/best/2
    ui = bm + F*(pm1 - pm2 + pm3 - pm4);  % differential variation
    ui = popold.*mpo + ui.*mui;  % crossover
elseif (st == 5)  % DE/rand/2
    ui = pm5 + F*(pm1 - pm2 + pm3 - pm4);  % differential variation
    ui = popold.*mpo + ui.*mui;  % crossover
end

%------Select which vectors are allowed to enter the new population------

for i=1:NP  % check cost of competitor
    tempval = feval(fname,ui(i,:),y);  %
    %tempval = feval(fname,ui(i,:));  % check cost of competitor
    nfeval = nfeval + 1;
    if (tempval <= val(i))  % if competitor is better than value in "cost array"
        pop(i,:) = ui(i,:);  % replace old vector with new one (for new iteration)
        val(i) = tempval;  % save value in "cost array"
%----we update bestval only in case of success to save time-----

if (tempval < bestval) % if competitor better than the best one ever
    bestval = tempval; % new best value
    bestmem = ui(i,:); % new best parameter vector ever
end
end %---end for imember=1:NP

bestmemit = bestmem; % freeze the best member of this iteration for the coming
% iteration. This is needed for some of the strategies.

%---Output section-----------------------------------------------

if (refresh > 0)
    if (rem(iter,refresh) == 0)
        fprintf(1,'%d:  %f
',iter,bestval);
    end
end

%% statistics
minc(iter)=bestval;
meanc(iter)=mean(val);
stdc(iter)=std(val);
iter = iter + 1;
end %---end while ((iter < itermax) ...

draw convergence plot;
iters=0:length(minc)-1;
plot(iters,minc,_iters,meanc,'--',iters,stdc,'-.');
xlabel('generation'); ylabel('cost');
text(0,minc(1),'best')
text(1,meanc(2),'population average')

%write results to Excel file for future plot;
xlswrite('C:\Program Files\MATLAB704\work\genetic algorithm\GAresult.xls',minc','de','A1')
xlswrite('C:\Program Files\MATLAB704\work\genetic algorithm\GAresult.xls',meanc','de','B1')

% this is to run CMAES for 24-digit real number sensor fusion;
% 5/2006

for repeat=1:10 % ten independent runs;
    opts.StopFitness=0;
end
opts.LBounds = 0; opts.UBounds = 1;

lambda=26; % set population size;
opts.PopSize=lambda;

fitfun='fupnn3'; % fupnn3 is dynamic fusion for validation data;
xstart=rand(24,1); % set xstart from range -1 to 1;
% it generates 24 digits with 8 digits for enose 32 sensors; and 16
digits
% for znose 64 windows; each digit represents 4 sensors;

insigma=0.5;

[xmin,fmin,counteval,stopflag]=cmaes07(fitfun,xstart,insigma,opts)
disp(['this is the repeat: ' num2str(repeat)])
end % end repeat;
Appendix C

% This program is to develop a Bayesian fusion network
% This program is contributed from Dr. Rick Sherry.
% June, 2006

clear all;

Truth(1:57,1)=1;
Truth(58:114,1) = 2;
FileID = 'Bayesian results_mod.txt';
FileAddress = ['C:\Program Files\MATLAB\R2006a\work\' FileID];
Data_Nose = load(FileAddress);
Enose_Data = Data_Nose(:,1:2);
Znose_Data = Data_Nose(:,3:4);
clear Data_Nose;
length_data = 114; % change to realistic value;

%Find ANN prediction for class
[dummy,Znose_max] = max(Znose_Data,[],2);
[dummy,Enose_max] = max(Enose_Data,[],2);

%Find runs where Enose and Znose are in error
Diff_Truth_Enose = Enose_max - Truth;
Diff_Truth_Znose = Znose_max - Truth;
Error_Enose = find(Diff_Truth_Enose);
Error_Znose = find(Diff_Truth_Znose);

%Find runs where both are in error
Both_in_error = [];
for i = 1:length(Error_Enose)
    match = find(Both_in_error == Error_Enose(i));
    if ~isempty(match)
        Both_in_error = [Both_in_error Error_Enose(i)];
    end
end

%Remove indices of runs where both are in error from Znose and Enose lists
Error_Enose_Only = [];
for i = 1:length(Error_Enose)
    match = find(Both_in_error == Error_Enose(i));
    if isempty(match)
        Error_Enose_Only = [Error_Enose_Only Error_Enose(i)];
    end
end

Error_Znose_Only = [];
for i = 1:length(Error_Znose)
    match = find(Both_in_error == Error_Znose(i));
    if isempty(match)
        Error_Znose_Only = [Error_Znose_Only Error_Znose(i)];
    end
end

Error_One_Only = sort([Error_Enose_Only Error_Znose_Only]);
Nose_Error_Flag = zeros(length(Error_One_Only),2);
for i = 1:length(Error_One_Only)
    match = find(Error_Enose_Only == Error_One_Only(i));
    if ~isempty(match)
        Nose_Error_Flag(i,1) = 1; % enose error;
    else
        Nose_Error_Flag(i,2) = 1; % znose error
    end
end

%adjust liklihoods to positive values and and make them sum to one
Znose_Data_Mod = Znose_Data;
Enose_Data_Mod = Enose_Data;
for i = 1:length(Error_One_Only)
    if Znose_Data_Mod(Error_One_Only(i),1) < 0
        Znose_Data_Mod(Error_One_Only(i),2) =
        Znose_Data_Mod(Error_One_Only(i),2) +
        Znose_Data_Mod(Error_One_Only(i),1);
        Znose_Data_Mod(Error_One_Only(i),1) = 0;
    end
    if Znose_Data_Mod(Error_One_Only(i),2) < 0
        Znose_Data_Mod(Error_One_Only(i),1) =
        Znose_Data_Mod(Error_One_Only(i),1) +
        Znose_Data_Mod(Error_One_Only(i),2);
        Znose_Data_Mod(Error_One_Only(i),2) = 0;
    end
    if Enose_Data_Mod(Error_One_Only(i),1) < 0
        Enose_Data_Mod(Error_One_Only(i),2) =
        Enose_Data_Mod(Error_One_Only(i),2) +
        Enose_Data_Mod(Error_One_Only(i),1);
        Enose_Data_Mod(Error_One_Only(i),1) = 0;
    end
    if Enose_Data_Mod(Error_One_Only(i),2) < 0
        Enose_Data_Mod(Error_One_Only(i),1) =
        Enose_Data_Mod(Error_One_Only(i),1) +
        Enose_Data_Mod(Error_One_Only(i),2);
        Enose_Data_Mod(Error_One_Only(i),2) = 0;
    end
end
for i = 1:length(Error_One_Only)
    Sum_Like = Znose_Data_Mod(Error_One_Only(i),1) +
    Znose_Data_Mod(Error_One_Only(i),2);
    Znose_Data_Mod(Error_One_Only(i),1) =
    Znose_Data_Mod(Error_One_Only(i),1)/Sum_Like;
    Znose_Data_Mod(Error_One_Only(i),2) =
    Znose_Data_Mod(Error_One_Only(i),2)/Sum_Like;

289
\[
\text{Sum}_\text{Like} = \text{Enose}_\text{Data}_\text{Mod}(\text{Error}_\text{One}_\text{Only}(i),1) + \text{Enose}_\text{Data}_\text{Mod}(\text{Error}_\text{One}_\text{Only}(i),2); \\
\text{Enose}_\text{Data}_\text{Mod}(\text{Error}_\text{One}_\text{Only}(i),1) = \frac{\text{Enose}_\text{Data}_\text{Mod}(\text{Error}_\text{One}_\text{Only}(i),1)}{\text{Sum}_\text{Like}}; \\
\text{Enose}_\text{Data}_\text{Mod}(\text{Error}_\text{One}_\text{Only}(i),2) = \frac{\text{Enose}_\text{Data}_\text{Mod}(\text{Error}_\text{One}_\text{Only}(i),2)}{\text{Sum}_\text{Like}}; \\
\text{end}
\]

```matlab
fid = fopen('C:\Program Files\MATLAB\R2006a\work\Bayes_Net_In.txt','w');
for i = 1:length(Error_One_Only)
    fprintf(fid, ' %4d %4d %4d %6.4f %6.4f %6.4f %6.4f
', Error_One_Only(i), ...
    Nose_Error_Flag(i,1),Nose_Error_Flag(i,2), ...
    Enose_Data_Mod(Error_One_Only(i),1), ...
    Enose_Data_Mod(Error_One_Only(i),2), ...
    Znose_Data_Mod(Error_One_Only(i),1), ...
    Znose_Data_Mod(Error_One_Only(i),2));
end 
```

clear all;
fprintf(1,'\n Three Node Model with Soft Evidence Model\n')

`FileID = 'Bayes_Net_In.txt';
FileAddress = ['C:\Program Files\MATLAB\R2006a\work' FileID];
Data_Nose = load(FileAddress);
Node_Index = Data_Nose(:,1);
Nose_Error_Flag = Data_Nose(:,2:3);
Enose_Data = Data_Nose(:,4:5);
Znose_Data = Data_Nose(:,6:7);
%clear Data_Nose;
length_data = length(Node_Index);

Apple = 1;
Enose = 2;
zNose = 3;

n = 3;
dag = zeros(n);
dag(Apple, Enose) = 1;`
dag(Apple, zNose) = 1;

NodeNames = {'Apple' 'Enose' 'zNose'};
ns = 2*ones(1,n);
dnodes = 1:n;

use_prior_performance = 1; % 1 = (use) , 2 = don't use
if use_prior_performance == 1
    fprintf(1,'Using Prior Nose Performance\n')
else
    fprintf(1,'Not Using Prior Nose Performance\n')
end

type_evidence = 1; % 1 = hard evidence, 2 = soft evidence
if type_evidence == 1
    fprintf(1,'Hard Evidence\n')
else
    fprintf(1,'Soft Evidence\n')
end

Count_Right = 0;
Count_Wrong = 0;
for i = 1:length_data

    bnet = mk_bnet(dag, ns, 'names', {'Apple', 'Enose', 'zNose'},
                  'discrete', dnodes);

    % true = good is 2, false = bad is 1
    bnet.CPD{Apple} = tabular_CPD(bnet, Apple, [0.36   0.64]);

    if use_prior_performance == 1
        bnet.CPD{Enose} = tabular_CPD(bnet, Enose, [0.85 0.15
                                                  0.15 .85]);
        bnet.CPD{zNose} = tabular_CPD(bnet, zNose, [0.57 0.09   0.43
                                                   0.91]);
    else
        bnet.CPD{Enose} = tabular_CPD(bnet, Enose, [1 0 0 1]);
        bnet.CPD{zNose} = tabular_CPD(bnet, zNose, [1 0 0 1]);
    end

    engine = jtree_inf_engine(bnet);

    if type_evidence == 1
        %Hard Evidence
evidence = cell(1,n);
if Enose_Data(i,1) > Enose_Data(i,2)
    evidence{Enose} = 2;
else
    evidence{Enose} = 1;
end
if Znose_Data(i,1) > Znose_Data(i,2)
    evidence{zNose} = 2;
else
    evidence{zNose} = 1;
end
[engine, loglik] = enter_evidence(engine, evidence);
else
    % Soft Evidence
    evidence = cell(1,n);
    soft_evidence = cell(1,n);
    evidence{Enose} = [];
    evidence{zNose} = [];
    soft_evidence{Enose} = [Enose_Data(i,2) Enose_Data(i,1)];
    soft_evidence{zNose} = [Znose_Data(i,2) Znose_Data(i,1)];
    [engine, loglik] = enter_evidence(engine, evidence, 'soft',
    soft_evidence);
end
obs_nodes = find(~isemptycell(evidence));
for iVar = 1:n
    MargProb(iVar) = marginal_nodes(engine, iVar);
end
Belief(i) = 1;
if MargProb(1).T(1) < MargProb(1).T(2)
    Belief(i) = 2;
end
% if Node_Index(i) <= 35 % first 35 are good;
if Node_Index(i) <= 57 % first 35 are good;
    if Belief(i) == 2;
        Count_Right = Count_Right + 1;
    else
        Count_Wrong = Count_Wrong + 1;
    end
else
    if Belief(i) == 1;
        Count_Right = Count_Right + 1;
    else
        Count_Wrong = Count_Wrong + 1;
    end
end
fprintf(1,'Run %4d %4d %4d
',Node_Index(i),Nose_Error_Flag(i,1),Nose_Error_Flag(i,2))
fprintf(1,'Marginal Posterior Probabilities \n')
fprintf(1,
       ' %-14s  Bad    Good\n')
fprintf(1,'%14s %6.4f %6.4f
',NodeNames{1},MargProb(1).T(1),MargProb(1).T(2))
for iVar = 2:n
    if isempty(find(obs_nodes == iVar))
        fprintf(1,'%-14s %6.4f %6.4f
',NodeNames{iVar},MargProb(iVar).T(1),MargProb(iVar).T(2))
    else
        T = 0;
        F = 1;
        if evidence{iVar} == 2;
            T = 1;
            F = 0;
        end
        fprintf(1,'%-14s %6.4f %6.4f\n',NodeNames{iVar},F,T)
    end
end
fprintf('
');
clear bnet;
clear MargProb;
clear obs_nodes;
clear evidence;
clear soft_evidence;
clear engine;
end
fprintf(1,'%4d %4d %4d
',Count_Right, Count_Wrong, Count_Right+Count_Wrong)
Appendix D

*zNose*™ data preprocessing algorithms

The *zNose*™ raw data needs to be preprocessed before further processing. The preprocessing algorithms were developed by former members in this lab (Korach, 2004; Veraverbeke et al., 2004) and they were reiterated here.

B.1 Vertical correction

In order to correct the vertical variation, the first derivative of the frequency data was taken using Equation (B.1).

\[
b_n = \frac{f_{n+2} - f_{n-2}}{t_{n+2} - t_{n-2}}
\]

where \(f\) is frequency, \(t\) is time, and \(b\) is the baseline corrected frequency at the \(n^{th}\) time step.

B.2 Horizontal shifts correction

Horizontal variation may be caused by fluctuations in injection time, temperature changes. If full spectra are used for further processing, this horizontal variation leads to misinterpretation. The horizontal shifts correction algorithm is based on Equation (B.2):

\[
t_{\text{new},i} = a + b t_{\text{old},i} + c t^2_{\text{old},i}
\]
where $t_{\text{new},i}$ is the corrected time which is assigned to the $i^{th}$ frequency reading, $t_{\text{old},i}$ is the original time for the $i^{th}$ frequency reading, $a$, $b$, and $c$ are the regression coefficients applied to transform the old time value into a new one. In the following scenarios:

1. $a=0$, $b=1$, and $c=0$; no horizontal shift correction is carried out;
2. $a \neq 0$, $b=1$ and $c=0$, the spectrum shifts over a constant value $a$.
3. $a \neq 0$, $b \neq 1$ and $c=0$, the new time value is a linear function of the old time value.
4. $a \neq 0$, $b \neq 1$ and $c \neq 0$, the original spectrum is stretched non-linearly over time, with the largest shifts for the frequency points with the largest original time values.

Reference:

Appendix E

% this is SAS MANOVA analysis program
options ls=80;
title "MANOVA - apple Data";
data apple1;
  infile "V:\My Documents\Matlab\myfile.txt";
  input sensor1 sensor2 sensor3 sensor4 sensor5 sensor6 sensor7 sensor8
  sensor9 sensor10 sensor11 sensor12 sensor13 sensor14 sensor15 sensor16
  sensor17 sensor18 sensor19 sensor20 sensor21 sensor22 sensor23 sensor24
  sensor25 sensor26 sensor27 sensor28 type $;
  run;
proc print;
  run;
proc glm;
  class type;
  model sensor1 sensor2 sensor3 sensor4 sensor5 sensor6 sensor7 sensor8
  sensor9 sensor10 sensor11 sensor12 sensor13 sensor14 sensor15 sensor16
  sensor17 sensor18 sensor19 sensor20 sensor21 sensor22 sensor23 sensor24
  sensor25 sensor26 sensor27 sensor28 = type;
  lsmeans type / stderr;
  manova h=type / printe printh;
  run;
# This is a sample PBS script. It will request 1 processor on 1 node
# for 48 hours.
#
# Request 1 processors on 1 node
#
#PBS -l nodes=1:ppn=1
#
# Request 48 hours of walltime
#
#PBS -l walltime=48:00:00
#
# Request that regular output and terminal output go to the same file
#
#PBS -j oe
#PBS -q lionxo-test
#
# The following is the body of the script. By default,
# PBS scripts execute in your home directory, not the
# directory from which they were submitted. The following
# line places you in the directory from which the job
# was submitted.
#
# cd ~
#
# Now we want to run the program "hello". "hello" is in
# the directory that this script is being submitted from,
# $PBS_O_WORKDIR.
#
echo " "
echo " "
echo "Job started on `hostname` at `date`"
unset DISPLAY
/usr/global/matlab/bin/matlab > runcase0.out 2>&1 << EOF
derun2
exit
EOF
echo " "
echo "Job Ended at `date`"
echo " "

Appendix G

Note: The following sensors and wavelengths windows were used in Table 9.2 and 9.3.

a Previously selected sensors and windows:

The 17 selected Enose sensors include: 2, 4, 5, 6, 8, 13, 14, 18, 20, 23, 24, 25, 27, 28, 30, 32;
The 37 selected zNose™ windows include: 3, 5, 6, 8, 9, 10, 11, 12, 14, 19, 20, 23, 26, 28, 30, 33, 34, 40, 41, 42, 44, 45, 46, 47, 48, 49, 50, 52, 54, 55, 57, 58, 59, 60, 62, 63, 64;

b Adaptively selected sensors and wavelength windows for calibration models for Enose and zNose™ individually:
Enose sensors include: 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30, 32;
zNose™ windows include: 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 19, 20, 21, 22, 25, 26, 34, 36, 38, 39, 41, 49, 50, 53, 56, 58, 59, 60, 62, 63, 64;

c Previously selected sensors for feature level (ANN) fusion model:
Enose sensors: 2, 4, 5, 6, 7, 8, 10, 11, 16, 18, 19, 20, 21, 23, 24, 25, 27, 29, 31, 32;
zNose™ windows: 3, 4, 5, 6, 7, 8, 17, 18, 20, 23, 34, 40, 45, 47, 48, 49, 50, 55, 57, 59, 61, 63;

d Adaptively selected sensors and wavelength windows for feature level (ANN) fusion model:
Enose sensors: 1, 4, 6, 7, 9, 12, 14, 15, 16, 17, 19, 23, 24, 25, 26, 29, 30, 31;
zNose™ windows: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 19, 21, 25, 27, 34, 40, 46, 47, 49, 50, 52, 53, 56, 60, 61, 62, 63, 64;

e Soft evidence with prior knowledge (prior knowledge was obtained in Chapter 8)

f Soft evidence without prior knowledge

g Hard evidence with prior knowledge
Vita

Changying Li was born in February 14, 1978 in Balikun, a small town in northwest China, where the winter temperature usually goes to \(-40^\circ\text{C}\). He was raised there until 14 years old and moved to Hami, a more populous and warmer city, with his parents. He completed his primary and secondary education in Xinjiang Province until 1996 when he went to college in Beijing, which is roughly 2,500 miles away from his hometown.

He studied Agricultural Architecture and Environmental Engineering at China Agricultural University (CAU) for four years before he started a master’s degree at the same university in 2000. When he was a junior in college, Changying was picked as one of four undergraduates of CAU to visit Norway, a rare opportunity for college students in China back in the 1990s. The three-month visit to this Scandinavia country, which is located at the other end of the Asian-Europe Continent, greatly broadened his view about the world in which he lived and a thought of pursuing a high degree abroad was embedded in his mind. In 2003, he finally made this dream come true by pursuing his doctoral degree at The Pennsylvania State University in the United States, which has been Changying’s longest journey both geographically and intellectually. The three-year experiences at Penn State were exciting, productive, and memorable and Changying expects to obtain his Ph.D. degree in December, 2006. He has accepted a Postdoctoral Fellowship at University of Illinois at Urbana-Champaign before his graduation.

Changying has interests and has been actively involved in various public services at different stages of his education. During his three-year Ph.D. study at Penn State, he had served the President of Chinese Friendship Association, which is the largest international student organization at Penn State, and the Chair of Student Activity Committee of American Overseas Chinese Scholars in Agricultural and Biological Engineering (AOC). He is interested in oration, debating, and singing, and has tried a hand in numerous competitions of these kinds. Changying practiced winter swimming for three years in Beijing and had kept jogging for more than 15 years. He is still single and eligible as of this writing, but does not want to keep this status too long.