



**ESTABLISHING A CLASSIFIER AND ESTIMATION
SYSTEM OF NITRIC OXIDE AND HYDROGEN
PEROXIDE IN LACUNAR STROKE BY
SUPPORT VECTOR
MACHINES**

BY

MR. YUTTHANA PHIMTHONG-NGAM

**A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR
OF PHILOSOPHY (MEDICAL ENGINEERING)
FACULTY OF ENGINEERING
THAMMASAT UNIVERSITY
ACADEMIC YEAR 2015
COPYRIGHT OF THAMMASAT UNIVERSITY**

**ESTABLISHING A CLASSIFIER AND ESTIMATION
SYSTEM OF NITRIC OXIDE AND HYDROGEN
PEROXIDE IN LACUNAR STROKE BY
SUPPORT VECTOR
MACHINES**

BY

MR. YUTTHANA PHIMTHONG-NGAM

**A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR
OF PHILOSOPHY (MEDICAL ENGINEERING)
FACULTY OF ENGINEERING
THAMMASAT UNIVERSITY
ACADEMIC YEAR 2015
COPYRIGHT OF THAMMASAT UNIVERSITY**



THAMMASAT UNIVERSITY
FACULTY OF ENGINEERING

DISSERTATION

BY

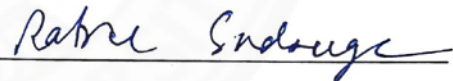
MR. YUTTHANA PHIMTHONG-NGAM

ENTITLED

ESTABLISHING A CLASSIFIER AND ESTIMATION SYSTEM OF NITRIC
OXIDE AND HYDROGEN PEROXIDE IN LACUNAR STROKE BY
SUPPORT VECTOR MACHINES

was approved as partial fulfillment of the requirements for
the degree of Doctor of Philosophy (Medical Engineering)
on July 27, 2015

Chairman



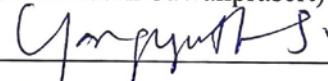
(Prof. Dr. Ratre Sudsuang)

Member and Advisor



(Assoc. Prof. Dr. Kesorn Suwanprasert)

Member



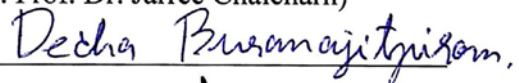
(Assoc. Prof. Yongyut Siripakarn, M.D.)

Member



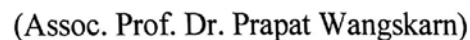
(Assoc. Prof. Dr. Jarree Chaicharn)

Member



(Dr. Decha Buranajitpirom)

Dean



(Assoc. Prof. Dr. Prapat Wangskarn)

| | |
|--------------------------------|--|
| Dissertation Title | ESTABLISHING A CLASSIFIER AND ESTIMATION SYSTEM OF NITRIC OXIDE AND HYDROGEN PEROXIDE IN LACUNAR STROKE BY SUPPORT VECTOR MACHINES |
| Author | Mr.Yutthana Phimthong-Ngam |
| Degree | Doctor of Philosophy (Medical Engineering) |
| Major Field/Faculty/University | Faculty of Engineering Thammasat University |
| Dissertation Advisor | Associate Professor Kesorn Suwanprasert, Ph.D. |
| Academic Years | 2015 |

ABSTRACT

In this study, we approached the performance of classification models by the support vector machines (SVMs) algorithms of NO and H₂O₂ concentration for assessment of a mechanism of lacunar stroke. Both NO and H₂O₂ are free radicals that play an important role as redox signaling, neuromodulator and oxidant in the system. We use cerebrovascular reactivity (CVR) test for assessment of cerebral reserve function and autoregulation in control vs LS. It is well known that the heart of SVMs classifier design is kernel function selection and adjustment of its parameters. We improve the most effective kernel function based on the best advantage of each single kernel model for training and testing these real-world medical data and then construct a novel integration of hybrid kernel function.

For NO, the classifier performances by each single kernel function and hybrid kernel function are following: it showed 90.00% from polynomial kernel function in experiment phase and is followed by 88.00% and 84.00% from polynomial in recovery and basal phase, respectively. Considering the best parameters of most experimental results, they have shown classifier performance of NO by polynomial is high when the parameter β is odd numbers. The classification results of RBF in

experiment phase is the best (94.00%), followed by results in basal (88.00%) and recovery (84.00%) phase, respectively. The classifier performance of RBF kernel is high when the parameter σ is an even number. For sigmoid kernel function have shown low performance of the most kernel function with highest classified accuracy is 78.00% in recovery phase and is followed by 74.00% and 58.00% in basal and experiment phase, respectively. The best parameters most experimental results have shown that classifier performance of sigmoid is high when the sum of a parameter γ and α is odd numbers. An improvement of classifier performance for NO assessment, when we vary the value of a parameter (η and ϕ) to test the data sets in each phrase, the results of a hybrid kernel in experiment phase is the best (96.00%), followed by results in basal (94.00%) and recovery (92.00%) phase, respectively. The best values of both parameters resulting in the highest performance for classification of NO by hybrid kernel function are 0.5. Consider the values of parameters compared with the highest performance, low value of parameters is detected.

In the case of H₂O₂, for polynomial, the highest classified accuracy is 80.00% in experiment phase and is followed by 72.00% and 50.00% in basal and recovery phases, respectively. The results of RBF in experiment phase is the best (90.00%), followed by results in basal (86.00%) and recovery (76.00%) phase, respectively. For sigmoid kernel function, it demonstrated the highest classified accuracy is 72.00% in basal phase and is followed by 58.00% and 54.00% in experiment and recovery phase, respectively. Contrast to NO, classifier performance of H₂O₂ by sigmoid kernel function is lower than those NO of all phases. For improvement of classifier performance for H₂O₂ assessment, the results of a hybrid kernel in experiment phase is the best (92.00%), followed by results in basal (86.00%) and recovery (82.00%) phase, respectively.

In conclusion, the findings suggest that low value of right parameters η and ϕ are able to improve performance of NO and H₂O₂ classification in lacunar stroke. Hybrid kernel function is the best giving the greatest classification accuracy for NO and H₂O₂ that play a novel role as neuromodulator and neurovascular oxidative stress.

Keywords: Classifier, Nitric Oxide, Hydrogen Peroxide, Lacunar Stroke, Support Vector Machines



ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who gave me the possibility to complete this dissertation. I am deeply indebted to my supervisor and advisor, Assoc. Prof. Dr. Kesorn Suwanprasert for stimulating suggestions and encouragement helped me in all the time of research and writing of this dissertation. In addition, special thanks for her guiding in a physiological background and giving me permission to commence this dissertation in the first instance.

I would like to thanks chairman, Prof. Dr. Ratre Sudsuang and also committee members, Asst. Prof. Yongyut Siripakarn, M.D., Assoc. Prof. Dr. Jarree Chaicharn and Dr. Decha Buranajitpirom for all advice and commendation.

I also would like to acknowledge grant from under the National Research University Project of Thailand and Office of Higher Education Commission and faculty of medicine, Thammasat University.

Finally, I would like to give my special thanks to my family and everyone, who have been proud of me, and who have been giving their endless love and encouragement to me.

Mr. Yutthana Phimthong-Ngam

TABLE OF CONTENTS

| | Page |
|--|------|
| ABSTRACT | (1) |
| ACKNOWLEDGEMENTS | (4) |
| LIST OF TABLES | (8) |
| LIST OF FIGURES | (11) |
| LIST OF ABBREVIATIONS | (14) |
| CHAPTER 1 INTRODUCTION | 1 |
| 1.1 Motivation | 1 |
| 1.2 The objectives | 3 |
| 1.3 Outline of studies | 3 |
| CHAPTER 2 REVIEW OF LITERATURE | 4 |
| 2.1 Lacunar stroke | 4 |
| 2.2 Epidemiology of lacunar stroke | 4 |
| 2.3 Endothelial barrier dysfunction in lacunar stroke | 6 |
| 2.4 Oxidative stress and endothelial barrier dysfunction | 9 |
| 2.5 Statistical learning theory | 10 |
| 2.6 Medical data classification problem | 12 |
| 2.7 Support vector machines | 12 |
| 2.7.1 Kernel functions and hyperplane classifiers | 14 |
| 2.7.1 Kernel functions and hyperplane classifiers | 15 |
| 2.7.1 Kernel functions and hyperplane classifiers | 18 |
| 2.8 Related tasks | 19 |

| | |
|--|----|
| CHAPTER 3 RESEARCH METHODOLOGY | 23 |
| 3.1 Subjects | 23 |
| 3.2 The Assessment of endothelial dysfunction | 23 |
| 3.2.1 Interpretation of nitric oxide (NO) level | 23 |
| 3.2.1.1 Method for the measurement of NO | 23 |
| 3.2.2 Interpretation of hydrogen peroxide (H ₂ O ₂) level | 24 |
| 3.2.2.1 Method for the measurement of H ₂ O ₂ | 24 |
| 3.3 Data standardization and normalization | 24 |
| 3.4 Experiment procedure | 27 |
| CHAPTER 4 RESULTS AND DISCUSSION | 32 |
| 4.1 Finding | 32 |
| 4.1.1 To determine Classifier performance of nitric oxide (NO) by each single kernel function | 32 |
| 4.1.1.1 The results of polynomial kernel function for NO assessment | 33 |
| 4.1.1.2 The results of Gaussian radial basis function (RBF) for NO assessment | 37 |
| 4.1.1.3 The results of sigmoid kernel function for NO assessment | 41 |
| 4.1.2 An improvement of classifier performance of NO assessment | 45 |
| 4.1.3 Comparison of best performance of hybrid kernel function and single kernel function for NO assessment | 49 |
| 4.1.4 To determine classifier performance of hydrogen peroxide (H ₂ O ₂) by each single kernel function | 52 |
| 4.1.4.1 The results of polynomial kernel function for H ₂ O ₂ assessment | 53 |
| 4.1.4.2 The results of Gaussian radial basis function (RBF) kernel function for H ₂ O ₂ assessment | 57 |
| 4.1.4.3 The results of sigmoid kernel function for H ₂ O ₂ assessment | 61 |

| | |
|---|------------|
| 4.1.5 An improvement of classifier performance for H ₂ O ₂ assessment | 65 |
| 4.1.6 Comparison of best performance hybrid kernel function and single kernel function for H ₂ O ₂ assessment | 69 |
| 4.1.7 The results of the best classification modeling for classification of NO and H ₂ O ₂ | 72 |
| 4.2 Discussion | 74 |
| CHAPTER 5 CONCLUSIONS AND RECOMMENDATIONS | 76 |
| 5.1 Conclusions | 76 |
| REFERENCES | 78 |
| APPENDICES | 76 |
| APPENDIX A | 88 |
| Source code of experiments for classification modeling (For the Scilab programming language) | 88 |
| APPENDIX B | 163 |
| Academic publications | 163 |
| BIOGRAPHY | 166 |

LIST OF TABLES

| Tables | Page |
|--|------|
| 4.1 Comparative data of NO concentration in control and patient subjects | 32 |
| 4.2 Classifier performance of NO by polynomial kernel function in basal phase | 33 |
| 4.3 Classifier performance of NO by polynomial kernel function in experiment phase | 34 |
| 4.4 Classifier performance of NO by polynomial kernel function in recovery phase | 35 |
| 4.5 Classifier performance of NO by RBF kernel function in basal phase | 37 |
| 4.6 Classifier performance of NO by RBF kernel function in experiment phase | 38 |
| 4.7 Classifier performance of NO by RBF kernel function in recovery phase | 39 |
| 4.8 Classifier performance of NO by sigmoid kernel function in basal phase | 41 |
| 4.9 Classifier performance of NO by sigmoid kernel function in experiment phase | 42 |
| 4.10 Classifier performance of NO by sigmoid kernel function in recovery phase | 43 |
| 4.11 Classifier performance of NO by hybrid kernel function in basal phase | 45 |
| 4.12 Classifier performance of NO by hybrid kernel function in experiment phase | 46 |
| 4.13 Classifier performance of NO by hybrid kernel function in recovery phase | 47 |
| 4.14 Comparison of best performance of hybrid kernel function and single kernel function in basal phase for NO assessment | 49 |
| 4.15 Comparison of best performance of hybrid kernel function and single kernel function in experiment phase for NO assessment | 50 |

| | | |
|------|--|----|
| 4.16 | Comparison of best performance hybrid kernel function and single kernel function in recovery phase for NO assessment | 51 |
| 4.17 | Comparative data of H ₂ O ₂ concentration in control and patient subjects | 52 |
| 4.18 | Classifier performance of H ₂ O ₂ by polynomial kernel function in basal phase | 53 |
| 4.19 | Classifier performance of H ₂ O ₂ by polynomial kernel function in experiment phase | 54 |
| 4.20 | Classifier performance of H ₂ O ₂ by polynomial kernel function in recovery phase | 55 |
| 4.21 | Classifier performance of H ₂ O ₂ by RBF kernel function in basal phase | 57 |
| 4.22 | Classifier performance of H ₂ O ₂ by RBF kernel function in experiment phase | 58 |
| 4.23 | Classifier performance of H ₂ O ₂ by RBF kernel function in recovery phase | 59 |
| 4.24 | Classifier performance of H ₂ O ₂ by sigmoid kernel function in basal phase | 61 |
| 4.25 | Classifier performance of H ₂ O ₂ by sigmoid kernel function in experiment phase | 62 |
| 4.26 | Classifier performance of H ₂ O ₂ by sigmoid kernel function in recovery phase | 63 |
| 4.27 | Classifier performance of H ₂ O ₂ by hybrid kernel function in basal phase | 65 |
| 4.28 | Classifier performance of H ₂ O ₂ by hybrid kernel function in experiment phase | 66 |
| 4.29 | Classifier performance of H ₂ O ₂ by hybrid kernel function in recovery phase | 67 |
| 4.30 | Comparison of best performance hybrid kernel function and single kernel function in basal phase for H ₂ O ₂ assessment | 69 |
| 4.31 | Comparison of best performance hybrid kernel function and single kernel unction in experiment phase for H ₂ O ₂ assessment | 70 |

| | | |
|------|---|----|
| 4.32 | Comparison of best performance hybrid kernel function and single kernel function in recovery phase for H ₂ O ₂ assessment | 71 |
| 4.33 | Summarize the results of the best modeling parameter in basal phase | 72 |
| 4.34 | Summarize the results of the best modeling parameter in experiment phase | 73 |
| 4.35 | Summarize the results of the best modeling parameter in recovery phase | 74 |



LIST OF FIGURES

| Figures | Page |
|--|------|
| 1.1 The conception framework | 3 |
| 2.1 Survival after lacunar infarcts compared with general population controls | 5 |
| 2.2 Scheme illustrating physiological and pathophysiological reactions of nitric oxide (NO) and hydrogen peroxide (H ₂ O ₂) | 10 |
| 2.3 Illustration of map data in higher-dimensional space and separate it there with a hyperplane | 15 |
| 2.4 Showing a) the distance from the hyperplane to the origin b) the margin | 17 |
| 3.1 Illustration of normal distribution | 27 |
| 3.2 The experimental procedure | 31 |
| 4.1 Separation of NO data assessed by polynomial kernel function in basal phase | 33 |
| 4.2 Separation of NO data assessed by polynomial kernel function in experiment phase | 34 |
| 4.3 Separation of NO data assessed by polynomial kernel function in recovery phase | 35 |
| 4.4 Comparison of best classifier performance of NO by polynomial kernel function in each phase of CVR test | 36 |
| 4.5 Separation of NO data assessed by RBF kernel function in basal phase | 37 |
| 4.6 Separation of NO data assessed by RBF kernel function in experiment phase | 38 |
| 4.7 Separation of NO data assessed by polynomial kernel function in recovery phase | 39 |
| 4.8 Comparison of best classifier performance for NO of RBF kernel function in each phase of CVR test | 40 |

| | | |
|------|---|----|
| 4.9 | Separation of NO data assessed by sigmoid kernel function in basal phase | 41 |
| 4.10 | Separation of NO data assessed by polynomial kernel function in experiment phase | 42 |
| 4.11 | Separation of NO data assessed by polynomial kernel function in recovery phase | 43 |
| 4.12 | Comparison of best classifier performance of NO by sigmoid kernel function in each phase of CVR test | 44 |
| 4.13 | Separation of NO data assessed by hybrid kernel function in basal phase | 45 |
| 4.14 | Separation of NO data assessed by hybrid kernel function in experiment phase | 46 |
| 4.15 | Separation of NO data assessed by hybrid kernel function in recovery phase | 47 |
| 4.16 | Comparison of best classifier performance of NO by hybrid kernel function in each phase of CVR test | 48 |
| 4.17 | Comparison chart of best performance of hybrid kernel function and single kernel function in basal phase for NO assessment | 49 |
| 4.18 | Comparison chart of best performance hybrid kernel function and single kernel function in experiment phase for NO assessment | 50 |
| 4.19 | Comparison chart of best performance hybrid kernel function and single kernel function in recovery phase for NO assessment | 51 |
| 4.20 | Separation of H ₂ O ₂ data assessed by polynomial kernel function in basal phase | 53 |
| 4.21 | Separation of H ₂ O ₂ data assessed by polynomial kernel function in experiment phase | 54 |
| 4.22 | Separation of H ₂ O ₂ data assessed by polynomial kernel function in recovery phase | 55 |
| 4.23 | Comparison of best classifier performance for H ₂ O ₂ of polynomial kernel function in each phase of CVR test | 56 |

| | | |
|------|---|----|
| 4.24 | Separation of H ₂ O ₂ data assessed by RBF kernel function in basal phase | 57 |
| 4.25 | Separation of H ₂ O ₂ data assessed by RBF kernel function in experiment phase | 58 |
| 4.26 | Separation of H ₂ O ₂ data assessed by RBF kernel function in recovery phase | 59 |
| 4.27 | Comparison of best classifier performance of H ₂ O ₂ by RBF kernel function in each phase of CVR test | 60 |
| 4.28 | Separation of H ₂ O ₂ data assessed by sigmoid kernel function in basal phase | 61 |
| 4.29 | Separation of H ₂ O ₂ data assessed by sigmoid kernel function in experiment phase | 62 |
| 4.30 | Separation of H ₂ O ₂ data assessed by sigmoid kernel function in recovery phase | 63 |
| 4.31 | Comparison of best classifier performance for H ₂ O ₂ of sigmoid kernel function in each phase of CVR test | 64 |
| 4.32 | Separation of H ₂ O ₂ data assessed by hybrid kernel function in basal phase | 65 |
| 4.33 | Separation of H ₂ O ₂ data assessed by hybrid kernel function in experiment phase | 66 |
| 4.34 | Separation of H ₂ O ₂ data assessed by hybrid kernel function in recovery phase | 67 |
| 4.35 | Comparison of best classifier performance of H ₂ O ₂ by hybrid kernel function in each phase of CVR test | 68 |
| 4.36 | Comparison chart of best performance hybrid kernel function and single kernel function in basal phase for H ₂ O ₂ assessment | 69 |
| 4.37 | Comparison chart of best performance hybrid kernel function and single kernel function in recovery phase for H ₂ O ₂ assessment | 70 |
| 4.38 | Comparison chart of best performance hybrid kernel function and single kernel function in recovery phase for H ₂ O ₂ assessment | 71 |

LIST OF ABBREVIATIONS

| Symbols/Abbreviations | Terms |
|-------------------------------|--------------------------------|
| % | percentage |
| et al. | et alibi, and others |
| etc. | et cetera, and other things |
| e.g. | for example |
| [ml]LS | lacunar stroke |
| NO | nitric oxide |
| H ₂ O ₂ | hydrogen peroxide |
| nNOS | neuronal nitric oxide synthase |
| eNOS | endothelial NO synthase |
| SVMs | Support Vector Machines |
| η | eta |
| α | alpha |
| β | beta |
| γ | gamma |
| χ | chi |
| ϕ | phi |
| ξ | xi |
| μ | mu |
| λ | lambda |
| CVR | cerebrovascular reactivity |
| ROS | reactive oxygen species |
| O ₂ •- | superoxide anion |
| ONOO- | peroxynitrite |
| ADMA | asymmetric dimethyl arginine |
| FMD | flow-mediated vasodilatation |
| ROS | reactive oxygen species |
| pA | Pico Ampere |

CHAPTER1

INTRODUCTION

1.1 Motivation

Lacunar stroke (LS) or lacunar infarction is small deep artery occlusion caused by prolonged cerebral endothelial dysfunction overwhelmed free radicals indeed vascular inflammation. Several lines of evidence have reported that LS is closely related to recurrent stroke and death (1).

Normally, the endothelium forms a relatively impermeable barrier between elements of circulating blood and the vessel wall, secretes vasoactive substances to maintain an antithrombotic surface, regulates vessel tone, modulates inflammatory responses, and inhibits proliferation of underlying vascular smooth muscle cells. The endothelium modulates vascular tone by synthesizing and releasing vasoactive that promote vasodilation. Nitric oxide (NO) plays a vital role as mediator of vasodilation in blood vessels (2). In cerebral artery, it is produced by eNOS resulting phosphorylation of multiple proteins that cause smooth muscle relaxation. Moreover, NO is produced by nNOS activation of L-arginine in brain causing glutamate release at synaptic junction. Thus, NO is an important neuromodulator for excitation of neuron through glutamate junction. Once it is synthesized, it is able to diffuse permeable cell membrane of vascular wall to circulation at same time that it diffuses and acts at smooth muscle cell or vessel. Thus, measurement of circulating NO is very useful for endothelial assessment. Interestingly, activation of cerebral autoregulation by cerebrovascular reactivity (CVR) within 30 seconds produces high level of circulating NO associated with increase in cerebral blood flow from cerebral vasodilator response (3).

Generally, physiologic level of reactive oxygen species (ROS): superoxide anion ($O_2^{\bullet-}$), hydrogen peroxide (H_2O_2) and hydroxyl radical ($\cdot OH$) is controlled by antioxidant scavengers and these ROS play important role as redox signaling. The redox signaling is implicated in many different pathological processes and physiological in the vasculature. Overwhelmed ROS damage cellular function and cause pathophysiological process. Unstable O_2 by NADH/NADPH oxidase in vascular wall

results free radical superoxide ($O_2 \bullet^-$) which it is further dismutation by superoxide dismutase (SOD) to be hydrogen peroxide (H_2O_2). If cellular metabolism is completely, H_2O_2 is converted to be H_2O and free radicals are removed from the system. In another word, if cellular metabolism is not completely, H_2O_2 is accumulated and damages cell membrane and cellular function by lipid peroxidation and protein oxidation. If the system has large amount of NO, $O_2 \bullet^-$ will react with NO rapidly and produces peroxynitrite ($ONOO^-$), one of reactive nitrogen species (RNS). Eventually, cell or organ will be further damaged by both of ROS and RNS causing pathophysiological disease. For H_2O_2 , in normal level, it involves physiological role in defend mechanism and cellular signaling. H_2O_2 has biphasic fashion: cerebral vasodilation and peripheral vasoconstriction. During LS, neuronal death followed by glia activation is commonly found so that, low level of NO with high level of H_2O_2 may be neurovascular oxidative stress marker. Thus, H_2O_2 measurement is a valuable tool for oxidative stress study and reflects to $O_2 \bullet^-$ in the system (4).

Development of artificial intelligent systems for medical applications is an important and interesting area. Ordinarily, a physician typically accumulates their knowledge based on patients' symptoms and confirmed diagnoses. In other words, prognostic relevance of symptoms towards certain diseases and diagnostic accuracy of a patient are highly dependent on a physician's experience. As such, the development of medical decision support systems becomes a viable approach be assistant tool for physicians to swiftly and accurately diagnose patients (5). Support Vector Machines (SVMs) are high-performance supervised machine learning techniques used in regression, classification, and other learning tasks. In classification, SVMs are a powerful binary classifier for solving binary problems in nonlinear classification. In physiological response, all data are mapped to a high-dimensional feature space using a mapping function that separates the data into two classes within the maximum margin. The mapping function is called kernel function, which it is a key part of SVMs (6).

The goal of our work is the effective procedure for NO and H_2O_2 classification and to distinguish their level in young healthy and in LS. Both biomarker concentration was real time monitored by using electrochemistry method. Collected data are characterized and used as training data for classification model study. Effective

model will be used for prediction of high risk, recurrent and follow up stroke as well as recovery from any intervention treatment.

1.2 The objectives

1. To utilize a Support Vector Machines (SVMs) methodology to assess pathophysiological mechanisms during complexes correlation in lacunar stroke by analyzing classify nitric oxide (NO) and hydrogen peroxide (H_2O_2) in healthy control and lacunar stroke patient.

2. To evaluate and enhancements the performance of our algorithm for predicting lacunar stroke.

1.3 Outline of studies

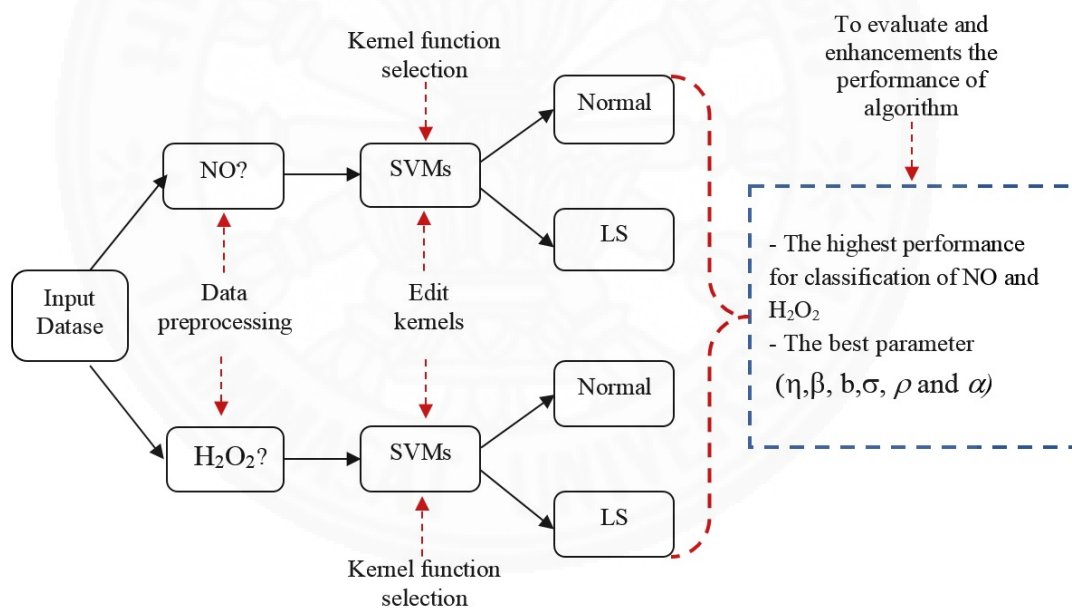


Figure 1.1 The conceptual framework.

CHAPTER 2

REVIEW OF LITERATURE

2.1 Lacunar stroke

Lacunar stroke is small deep infarcts as a result of the occlusion of penetrating arteries in the deep brain tissue. Normally it located in the brainstem, corona radiata, internal capsule, thalamus and the brainstem as a result of occlusion of a single perforating artery(7). Many studies showed that most infarcts were caused by two types of arterial pathology: intracranial atherosclerosis (in situ atheroma either at the mouth or along the length of the penetrating vessel), and segmental arterial disorganization or lipohyalinosis secondary to the effects of hypertension (8). These pathologies are distinct from the concentric hyaline wall thickening (hyaline atherosclerosis), which is the main determinant of widespread incomplete infarction of white matter that causes vascular dementia (9). However, lacunar infarcts and more diffuse cerebral small-artery disease affecting the white matter commonly coincide and share many of the same risk factors. The total number of lacunar infarcts studied pathologically is small, and most studies were done in an era when hypertension was poorly detected and managed. Most lacunae are clinically undetectable and part of a broader category of cerebral small-vessel disease. Cerebral-arteriole changes may even be part of a more generalized small-vessel disease, as suggested by a recent report that showed pathological changes in the retinal arteries parallel to findings of lacunae (10).

2.2 Epidemiology of lacunar stroke

LS is account for approximately one-fourth of all ischemic strokes. Several studies of lacunar stroke find a low risk of death, which should come as no surprise because the primary lesion is small, the rate of recovery is normally rapid (which decreases the risk of death due to secondary complications), and cardiac comorbidities are less common than in most other stroke types(11). The mean case fatality was 2.5% (range 0-10%) at 30 days, and 2.8% (range 2-15%) at 1 year. Differences between

studies may arise from limited accuracy owing to small numbers of patients and low numbers of outcome events. However, after 5 years of follow-up about a quarter of all patients had died (mean value 27.4%; range 17-38%). The average death rate per year was 2.8% (range 3–15%) in all studies, but notably higher mean 5.1 % (range 3–7 %) in studies with follow-up times of 4 years or more (12).

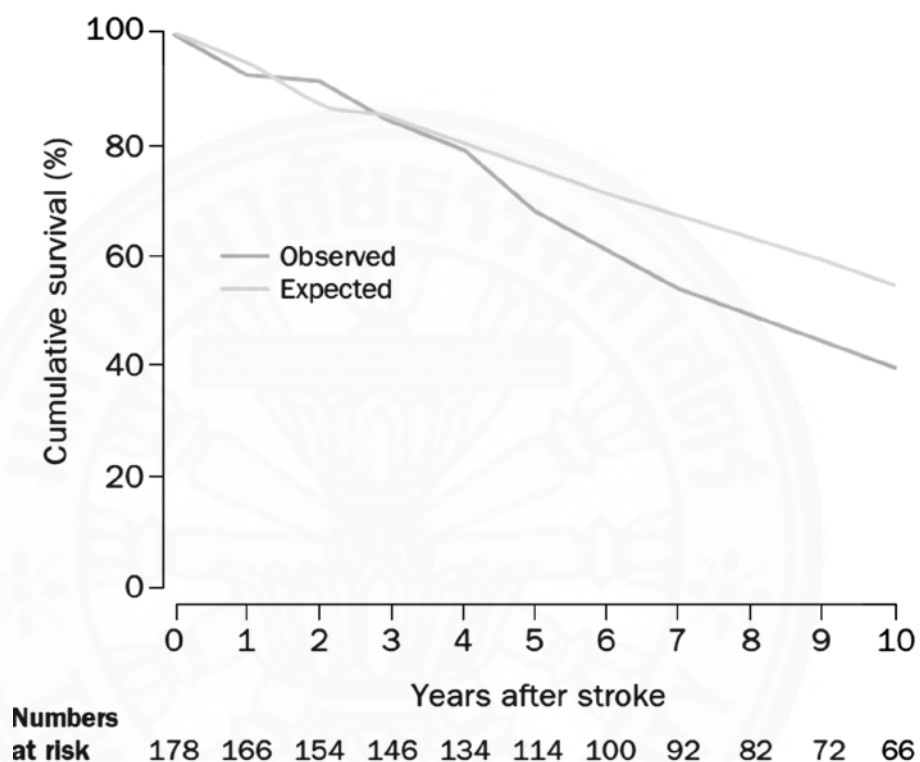


Figure 2.1 Survival after lacunar infarcts compared with general population controls (2).

By searching literature is a population based study from 1960 to 1984, which found the age and sex adjusted average annual incidence rate of lacunar cerebral infarction was 13.4/100,000 persons, accounting for 12% of all first cerebral infarcts. Temporal trends in incidence rates, stroke recurrence rates, prevalence of diabetes mellitus, and causes of death (given survival for 30 days) for cases of lacunar infarction were not significantly different from those for cases of nonlacunar infarction. Hypertension was found in 81% of patients who had a lacunar infarct and in 70% of patients who had a nonlacunar infarct ($p=0.05$). A potential cardiac source of embolism

was found in 12% of patients who had a lacunar infarct and in 28% of patients who had a nonlacunar infarct ($p=0.002$). Survival was significantly better after a lacunar infarct than after a nonlacunar infarct (13).

2.3 Endothelial barrier dysfunction in lacunar stroke

The endothelium impairs production of homeostatic mediators of vascular conditions, (e.g. NO), Occurring in an intima that is identified by improved dysregulated vascular smooth muscle cell growth, thrombus formation and, aberrant vessel tone. One of the signs of an endothelium dysfunction is decreased levels of bioavailable NO (14). This could be a result of an increase in NO inactivation or decrease in NO synthesis due to the locally increased production of reactive oxygen species (ROS) (15). The endothelial dysfunction acts as an initial marker of atherosclerosis as shown by the cognizance that fatty streak progression is related to more impaired vascular relaxation (16). The several coronary artery disease risk factors recognized with endothelial dysfunction, together with diabetes mellitus, tobacco use, hypercholesterolemia, and hyperhomocysteinemia are associated with diminished bioavailable NO as proofed by an abnormal coronary vasodilator reaction to acetylcholine challenge (17).

The appearance of endogenous NOS inhibitors in vivo is confirmed as one mechanism relates to decrease NO production. A naturally occurring of asymmetric dimethyl arginine (ADMA), L-arginine, is confirmed recently a competitive inhibitor of NOS (18). The levels of ADMA are increased d in patients with hypercholesterolemia and atherosclerotic vascular disease and, in fact, plasma ADMA levels as been in a relationship with the severity of endothelial dysfunction (19). To interrupt production of ADMA, NO itself maybe a new risk factor for the improvement of atherothrombotic vascular disease. L-arginine supplementation increases the L-arginine/ADMA ratio and thereby improves endothelial function (20). In fact, physiological studies demonstrated that chronic oral administration of L-arginine to repair eNOS activity in hypercholesterolemia rabbits is associated with a delay in progression of the atherosclerosis and a reduction in the lesion surface area and

thickness (21). Similarly, in patients with hypercholesterolemia, intravenous L-arginine improves NO mediated brachial artery reactivity. Unless reduced production of NO, endothelial dysfunction is related to an increase in vascular oxidant stress from the production of especially superoxide anion, endogenous ROS, in excess of antioxidant capacity. Once formed, superoxide reacts with NO to form peroxynitrite. This reaction, which consumes NO, is diffusion-limited and three times faster than the catabolism of superoxide by superoxide dismutase (22).

The physiologic condition, NO is seemingly formed at a concentration in the pico-molar to nano-molar range, and the presence of antioxidant defenses minimizes its utilization by superoxide (23). At very low concentrations, peroxynitrite has the same bioactive as NO; it is only at higher levels that it exerts its toxic effects by forming the cytotoxic peroxynitrous acid, as well as resulting in protein modification by nitration of amino acids (24).

There are several key sources of ROS production in the vasculature that could be activated directly or indirectly as in the setting of endothelial dysfunction. Free radical generation by endothelial, vascular smooth muscle, or adventitial cells in the vessel wall may occur via activation of enzyme systems such as NAD (P) H oxidase(s), which has been identified in vascular cells and shown to produce abundant ROS in response to angiotensin II, thrombin, tumor necrosis factor- α , or elevated mechanical forces (22-24). A second important source of free radical production is the enzyme xanthine oxidase. In hypertensive rats with elevated levels of vascular ROS, oral administration of oxypurinol, a xanthine oxidase inhibitor, lowered blood pressure, suggesting enhanced bioavailability of NO (25). Similarly, hypercholesterolemic animals with established atherosclerotic vascular disease demonstrate increased levels of superoxide anion production that may be reversed by treatment with oxypurinol. In hypercholesterolemic patients, intravenous administration of oxypurinol similarly improved the blunted vasodilator response to acetylcholine (26). Finally, NOS itself is capable of producing ROS. In the absence of substrate, L-arginine, or cofactors such as BH₄, NOS has been shown to synthesize superoxide in preference to NO (enzyme ‘‘uncoupling’’). Thus, coronary artery disease risk factors that deplete levels of L-arginine or BH₄ may promote NOS-mediated ROS formation, and, in turn, increase peroxynitrite generation (27, 28). Increased levels of ROS additionally influence the

oxidation of lipids to initiate plaque development. Small, dense LDL particles are more susceptible to oxidative modification, and many traditional coronary artery disease risk factors appear to accelerate this process (29). In hypercholesterolemia, LDL metabolism by endothelial cells and macrophages in the vessel wall deplete local antioxidant defenses such as α -tocopherol and coenzyme Q (30). Once these antioxidant defenses have been overwhelmed, LDL particles undergo oxidative modification. The mechanism of oxidation starts by the formation of a radical species from a nonradical fatty acid precursor (31). Oxidized LDL is also able to regenerate itself by autocatalytic lipid peroxidation. Once formed, oxidized LDL, in turn, reduces NO levels by decreasing the expression of eNOS via destabilization of posttranscriptional mRNA (32).

Oxidized LDL may also deplete NO directly by forming lipid peroxynitrites. Decreased NO bioavailability disrupts the nonthrombogenic intimal surface and promotes platelet adhesion and aggregation, as well as deposition of platelets on the abnormal endothelial surface. In the setting of established atherosclerosis, there are additional abnormalities in the coagulation system that favors thrombus formation, including an increase in circulating von Willebrand factor and a decrease in heparan sulfated glycosaminoglycans (33). Furthermore, oxidized LDL or homocysteine increases the activity of tissue factor, and thereby activation of the extrinsic clotting cascade (34). Cellular signals that initiate and amplify the inflammatory response may be activated in the absence of NO. For example, protein kinase C is activated to initiate the production of cytokines, interleukins, and tumor necrosis factor- α , and the expression of membrane adhesion molecules and chemotactic proteins, such as monocyte chemoattractant protein-1 (MCP-1). The expression of these inflammatory mediators and membrane receptors enhances monocyte and lymphocyte adhesion and migration into the vessel wall (35). The loss of NO production and the presence of ROS stimulate vascular smooth muscle cell proliferation. Activated smooth muscle cells undergo phenotype modulation from a contractile to a synthetic state, which initiates cell proliferation and migration to the intima and produces extracellular matrix (36). High levels of ROS adjust hormone-mediated hypertrophy and promote neointimal hyperplasia and formation of the fibrous component of the atherosclerotic lesion (37, 38).

2.4 Oxidative stress and endothelial barrier dysfunction

Oxidative stress is a set of circumstances under which there is an imbalance between pro oxidant and antioxidant influences at a particular tissue site. The vascular endothelium in general constitutes a primary target for oxidants released during acute inflammatory events. In the context of acute lung injury the associated generation of reactive oxidant species (ROS) can precipitate endothelial dysfunction in the short term and overt injury/cytotoxicity following chronic exposure. The initial extracellular species produced in many cases is the superoxide anion, which is then converted to H_2O_2 . Since the endothelial cell membrane is freely permeable to H_2O_2 , conversion of the latter to the hydroxyl radical can occur within the cell, thereby increasing the nature and extent of cell injury. A number of studies have shown that H_2O_2 itself can directly cause endothelial barrier dysfunction in endothelial cells; for example, in bovine aortic endothelial cells, a 90 min exposure to H_2O_2 (0.1-30 mM) increased albumin transfer in a stepwise manner, apparently without conversion to the hydroxyl radical (39). In pulmonary micro vascular endothelial cells, PKC activation and redistribution of cytoskeletal actin to the cell periphery was found to be involved in H_2O_2 -induced increases in permeability (40), following a 60 min challenge with (0.075-0.5 mM) H_2O_2 . In bovine pulmonary artery endothelial monolayers exposed to 0-2 or 1 mM H_2O_2 for 60 min, increases in albumin transfer were associated with changes in plasma membrane and cell functional complexes with a consequent change in cell shape. This loss of barrier function was not reversed 48 h later, in contrast to the partially reversible effects seen with a shorter H_2O_2 exposure time of 20 min (41). Reductions in cell viability associated with overt damage to endothelium have been observed following longer exposure times to oxidant; for example, incubation of porcine aortic endothelial cells with H_2O_2 caused a decrease in cell viability that was dependent on dose.

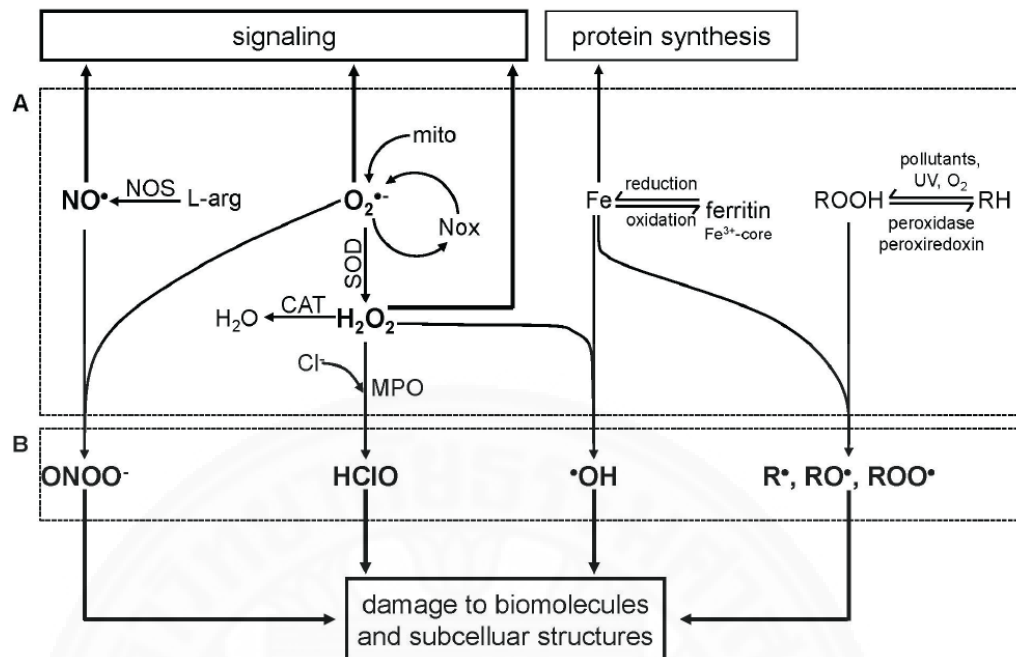


Figure 2.2 Scheme illustrating physiological and pathophysiological reactions of nitric oxide (NO) and hydrogen peroxide (H₂O₂).

2.5 Statistical learning theory

Statistical learning theory (STL) has had the most widespread and profound impact on learning theory over the last two decades, which could be supported by the over 700 references. STL has been synthesized and developed primarily by Vladimir N. Vapnik (42). In much of the work on distributed data mining (DDM), the primary goal is identical to that of single-source data mining, namely the ability of the learner to generalize across unseen data examples. To achieve this behavior of the machine-learner, we need to present to it a series of data points that represent well the underlying phenomenon that we seek to make a model for. As classification learners are typically evaluated on their ability to generalize well, so are SVMs when used in classification problem settings (43). A classification problem covers a broad range of application. One application could be to determine if an image is a tree or not. Another application could be to determine a state model for a distributed sensor network. The generalization capability comes from the fact that SVMs are large margin classifiers (44). This means the training algorithm seeks to separate the positive points from the negative points, if

the learning process is a binary classification problem. The problem of identifying a vectorial images as either a tree or not is an example of a binary classification problem (45). This is achieved by trying to separate the two classes of points with a plane that is equidistant from the closest points of each class. If there was two measurements on a line and each one belonged to different classes, then the SVM would place the discriminant function mid-way between the two points. The solution can also be restated as the plane that separates the convex hulls of each class with the largest possible margin (46). A convex hull is set of points on the boundary of some class of points. Some classification problems are either so noisy or impossible to separate completely due to the lack of information in the chosen dimensions describing properties of the class instances Noise is present when poor measurements is part of the problem domain or there are lacking key attributed to achieve separation of the data points The solution to this problem was proposed by C. Cortes and V. Vapnik (47). In their paper, the soft margin SVM is introduced, which allows the SVM be trained using slack margins (48). A slack margin is a method incorporated in this algorithm to balance the misclassification rate against generalization ability. Misclassification is related to those data instances that cannot be classified correctly during the training process of the model (49). This is balanced by introducing an upper limit to the size that the individual Lagrange multipliers can take. The Lagrange multiplier is part of a linear combination of training points in feature space. A common choice is to set this upper limit to 10, but it will be problem dependent. Given that kernel has been chosen, the choice of C , which is the name for this parameter, is optimized to the training set by use of a validation set. The value of C is varied over a range of numbers while the classification accuracy is monitored against a validation set. There are variations of this scheme using the parameter C to control the soft margins. A complementary SVM is the ν -SVM that depends on a parameter ν , to control the margins. The parameter ν is interesting in that it provides an upper bound on the number of support vectors in the solution (50).

2.6 Medical data classification problem

Many methods have been proposed for classification problems in medical data. Today not only simple execution tasks are requested but also decisions on incoming data. These tasks can be very complex, like biometric identification tasks. A human identification problem cannot be solved by classical hard-structured computer programs because the developer cannot precisely specify the way how to get a correct identification from a given input. A powerful approach to this kind of problems is through learning a learning algorithm constructs a decision function based on a part of the training data and some hypothesis. By applying the decision function on some new data, the computer can classify it as known or unknown (51).

2.7 Support vector machines

Support Vector Machines (SVMs) are a maximal margin algorithm. It seeks to place a hyperplane between classes of points such that the distances between the closest points are maximized. It is equivalent to maximum separation of the distance between the convex hulls enclosing the class member points (52). Vladimir Vapnik is respected as the researcher who primarily laid the groundwork for the support vector algorithm (53). The first breakthrough came in 1992 when Boser et al. constructed the SVM learning algorithm as we know it today (54). The algorithm worked for problems in which the two classes of points were separable by a hyperplane. In the meantime Corinna Cortes was completing her dissertation at AT&T labs, and she and Victor Vapnik worked out the soft margin approach. It involves the introduction of slack variables, or error margins that are introduced to absorb errors that are inevitable for non-separable problems (55). The SVM was primarily constructed to address binary classification problems. This has been adapted by introducing versions of the SVM that can train a multi classifier concurrently (56). Other approaches involved the use of voting schemes in which a meta-learner takes the votes from the individual binary classifiers and casts the final vote. A particularly easy voting scheme is the one-against-all voter, which for SVMs amounts to training C classifiers and finding the C_i classifier with the hyperplane furthest away from the new point to be tested. The SVM has been

extended to other learning areas as well. The areas relevant for this work are the extension toward regression and clustering (55). The regression algorithm extension has been refined by Alex Smola and Bernhard Schölkopf and pioneered by Vapnik (57). The regression case is carried out by using the slack variable approach once again. A so-called ε -tube is constructed around the regression line (58).

Classified data is an ordinary work in machine learning. Each data point is assigned to one of two floors and is aiming to make a decision on the new information. SVMs classification labels on different layers by a defined set of support vectors that are a member of the series of training inputs that schemes a hyperplane in the feature space (59). When to use SVMs, two problems should be fixed: how to select the best input feature subclass for SVMs, and how to set the greatest kernel parameters. Usually, the Two problems have been resolved separately neglect their close relationship, this often leads to a lower classification accuracy. Both problems are important because the feature subset choice that affects the appropriate attributes kernel parameters and vice versa. Consequently, to get the best feature subset and SVMs parameters must occur simultaneously (60).

Get a set of training examples, each conspicuous for belonging to one of two classes, a SVMs training algorithm builds a model to set an example in one category or the other, producing it a non-probabilistic binary linear classifier (61). A SVMs model represents a sample point in space, Mapping to provide examples of the classification that is divided by a gap as wide as possible. New samples are mapped into the same space and expected to be in categories depending on which side of the space they are in (62).

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces (63). Feature selection is used to identify a powerfully predictive subset of fields within a database and reduce the number of fields presented to the mining process (64). By extracting as much information as possible from a given data set while using the smallest number of features, we can save significant computational time and build models that generalize better for unseen data points (65). Feature subset selection is an important issue in building a SVMs-based classification model. As well as feature selection, the proper

setting of parameters for the SVMs classifier can also increase classification accuracy (66). The parameters that should be optimized include penalty parameter C and the kernel function parameters such as the gamma (γ) for the radial basis function (RBF) kernel. To design a SVMs classifier, one must choose a kernel function, set the kernel parameters and determine a soft margin constant C (penalty parameter). As a rule, the Grid algorithm is an alternative to finding the best C and gamma (γ) when using the RBF kernel function. However, this method is time consuming and does not perform well. Moreover, the Grid algorithm cannot perform the feature selection task (67).

Both feature subset selection and model parameter setting substantially influence classification accuracy. The optimal feature subset and model parameters must be determined simultaneously. Since feature subset and model parameters greatly affects the classification accuracy (68).

2.7.1 Kernel functions and hyperplane classifiers

The nature of SVMs is the clarification to a classification problem pointed out by the support vectors that determine the maximum margin hyperplane (69). SVMs can also be used to separate classes that cannot be separated with a linear classifier (70). In such cases, the coordinates of the objects are mapped into a feature space using nonlinear functions called feature functions. The feature space is a high-dimensional space in which the two classes can be separated with a linear classifier. The mapping function is called kernel function, which it is a key part of Support Vector Machines (71).

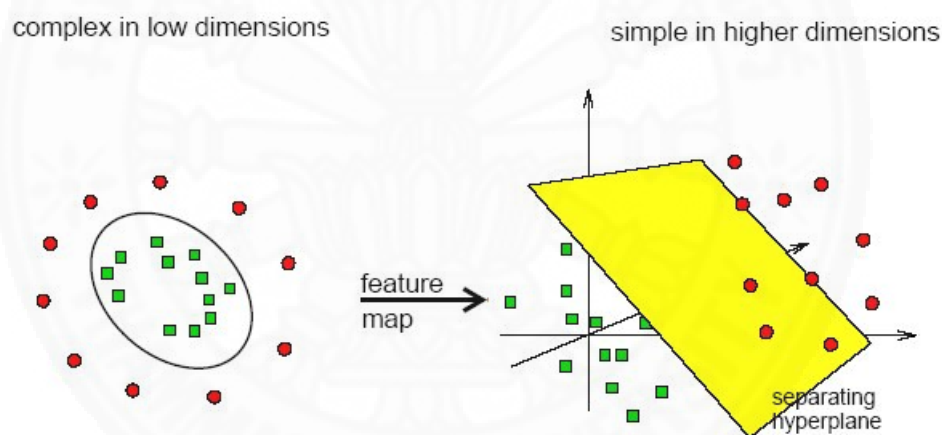
Kernel functions can be used for classification of every kind of data. On some given empirical data

$$(x_1, y_1), \dots, (x_n, y_n) \in \mathcal{X} \times \{\pm 1\} \quad (2.1)$$

a generalization to unseen data has to be done. The patterns x_i are taken from the domain \mathcal{X} , a non-empty set of data. The targets or labels $y_i = \{\pm 1\}$ define the membership of the pattern. When $x \in \mathcal{X}$ represents a test data, then the goal is to predict the corresponding label $y_i \in \{\pm 1\}$ such that (x, y) is in some sense similar to the training samples. Making decisions in the original data space is very difficult; in most

cases no trivial separation between data of two different targets can be found. To solve this problem the data is mapped with a non-linear mapping function ϕ to another high dimensional feature space H (72).

In a good feature space a separation function should be linear and general enough to provide acceptable results also for unseen data. To guarantee a good generalization, some feature selection has to be done (73). Non relevant features with low variance can be excluded from the feature space, other weighted to gain importance. Using too many features can cause an overfitting decision function, focused too much on the training data and unable to generalize to new data. The inverse problem underfitting occurs when too less features are selected (74).



Separation may be easier in higher dimensions

Figure 2.3 Illustration of map data in higher-dimensional space and separate it there with a hyperplane (75).

2.7.2 Linear support vector machines

The heart of SVMs classifier design is the notion of the margin. We start from the simple case of two linearly separable classes. Consider training data D , a set of N points of the form (76)

$$D = \{(x_i, y_i \mid x_i \in R^v, y_i \in \{-1, +1\})\}_{i=1}^N \quad (2.2)$$

where the y_i is either -1 or +1, indicating the class to which the point x_i belongs. Each x_i is a v -dimensional real vector. We want to find the maximum margin hyperplane that divides the points having $y_i = -1$ from those having $y_i = +1$. Any hyperplane can be written as the set of points X satisfying

$$w^T x + w_0 = 0 \quad (2.3)$$

The margin is the region between the two parallel hyperplanes

$$w^T x + w_0 = +1 \text{ and } w^T x + w_0 = -1 \quad (2.4)$$

It can be shown that the Euclidean distance of any point that lies on either of the two hyperplanes in Equation 2.3 from the classifier hyperplane given by Equation 2.4 is equal to $1/\|w\|$, $\|\bullet\|$ denotes the Euclidean norm.

Given a set of training points, x_i , with respective class labels, $y_i \in \{\pm 1\}$, $i = 1, 2, \dots, N$ for a 2-class classification task, compute a hyperplane Equation 2.2 so it is

$$\text{Minimize} \quad J(w, w_0, \xi) = C \sum_{i=1}^N \xi_i + \frac{1}{2} \|w\|^2 \quad (2.5)$$

$$\text{Subject to} \quad \xi_i \geq 0$$

$$w^T x_i + w_0 \geq 1 - \xi_i \quad \text{if } x_i \in w_1 \quad (2.6)$$

$$w^T x_i + w_0 \leq \xi_i - 1 \quad \text{if } x_i \in w_2 \quad (2.7)$$

where $\xi_i \geq 0$, C is a user-defined constant which in the cost function defines the compromise between misclassification error and a large margin. SVMs are based on the principle of structural risk minimization, which balances model complexity through regularization. The first set of constraints corresponds to

$y_i(w^T x_i + b) \geq 1$ while the second set imposes positive slack variables ξ_i , tolerating misclassifications. The value of ξ_i indicates the distance of x_i with respect to the decision boundary in the event of $\xi_i = 0$: x_i is correctly classified and lies outside the margin or on the margin boundary, $0 < \xi_i < 1$: x_i is correctly classified, but lies inside the margin and $\xi_i \geq 1$: It implies that the decision function and the target have a different sign, indicating that x_i is misclassified (This can be verified by a close inspection of the constraints in Equation 2.6 and 2.7.)

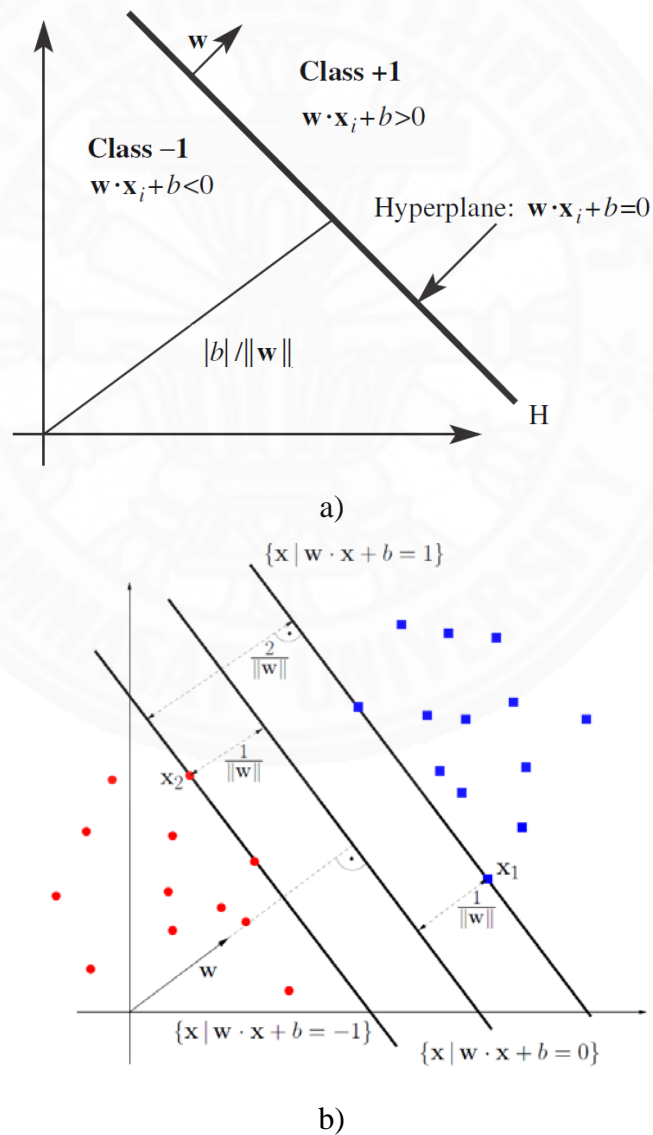


Figure 2.4 Showing a) the distance from the hyperplane to the origin b) the margin.

The margin width is equal to $2/\|w\|$. The solution is given as a weighted average of the training points:

$$w = \sum_{i=1}^N \lambda_i y_i x_i \quad (2.8)$$

λ_i are the Lagrange multipliers of the optimization task and they are zero for all points outside the margin and on the correct side of the classifier. The rest of the points, with nonzero λ_i , which contribute to the buildup of w , are called support vectors (77).

2.7.3 Nonlinear support vector machines

To set up an experiment, start by use the SVMs for solving a nonlinear classification task, we adopt the principles of mapping the feature vectors in a higher dimensional space, where we expect, with high probability, the classes to be linearly separable (78). The mapping is as follows ($x_i \in R^l$):

$$x_i \rightarrow \phi(x_i) \in H \quad (2.9)$$

where H is higher dimension than R^l . If the mapping function is carefully chosen from a known family of functions that have specific desirable properties, the inner product between $\phi(x_1)$ and $\phi(x_2)$ of two points x_1, x_2

$$\langle \phi(x_1), \phi(x_2) \rangle = k(x_1, x_2) \quad (2.10)$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product operation in H and $k(\cdot, \cdot)$ is a function known as kernel function (79). Typical examples of kernel functions are

1. Linear Kernel

$$k(x, y) = x^T y + c \quad (2.11)$$

2. Polynomial Kernel

$$k(x, y) = (x^T y + \beta)^d \quad (2.12)$$

where β and d are user-defined parameters.

3. Gaussian Radial Basis Function (RBF) Kernel

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \quad (2.13)$$

where σ is a user-defined parameter that specifies the rate of decay of $k(x, y)$ toward zero, as y moves away from x .

4. Sigmoid kernel function

$$k(x, y) = \tanh(\gamma(x^T \cdot y) + \alpha) \quad (2.14)$$

where γ and α are user-defined parameters (80).

2.8 Related tasks

Approaches for feature selection can be categorized into two models, namely a filter model and a wrapper model (81). Statistical techniques, such as principal component analysis, factor analysis, independent component analysis and discriminate analysis can be adopted in filter based feature selection approaches to investigate other

indirect performance measures, most of which are based on distance and information. Chen and Hsieh (82) presented latent semantic analysis and web page feature selection, which are combined with the SVMs technique to extract features. Gold (83) presented a Bayesian viewpoint of SVMs classifiers to tune hyper-parameter values in order to determine useful criteria for pruning irrelevant features.

The wrapper model (84) applies the classifier accuracy rate as the performance measure. Some researchers have concluded that if the purpose of the model is to minimize the classifier error rate, and the measurement cost for all the features is equal, then the classifier's predictive accuracy is the most important factor. Restated, the classifier should be constructed to achieve the highest classification accuracy. The features adopted by the classifier are then chosen as the optimal features. In the wrapper model, meta-heuristic approaches are commonly employed to help in looking for the best feature subset. Although meta-heuristic approaches are slow, they obtain the (near) best feature subset. Shon (85) employed GA to screen the features of a dataset. The selected subset of features is then fed into the SVMs for classification testing. Zhang (86) developed a GA-based approach to discover a beneficial subset of features for SVMs in machine condition monitoring. Samanta (87) proposed a GA approach to modify the RBF width parameter of SVMs with feature selection. Nevertheless, since these approaches only consider the RBF width parameter for the SVMs, they may miss the optimal parameter setting. Huang and Wang (88) presented a GA-based feature selection and parameters optimization for SVMs. Moreover, Huang et al. (89) utilized the GA-based feature selection and parameter optimization for credit scoring.

Several kernel functions help the SVMs obtain the optimal solution. The most frequently used such kernel functions are the polynomial, sigmoid and radial basis kernel function (RBF). The RBF is generally applied most frequently, because it can classify high-dimensional data, unlike a linear kernel function. Additionally, the RBF has fewer parameters to set than a polynomial kernel. RBF and other kernel functions have similar overall performance. Consequently, RBF is an effective option for kernel function. Therefore, this study applies an RBF kernel function in the SVMs to obtain optimal solution. Two major RBF parameters applied in SVMs, C and γ , must be set appropriately. Parameter C represents the cost of the penalty. The choice of value for C

influences on the classification outcome. If C is too large, then the classification accuracy rate is very high in the training phase, but very low in the testing phase. If C is too small, then the classification accuracy rate is unsatisfactory, making the model useless. Parameter γ has a much greater influence on classification outcomes than C , because its value affects the partitioning outcome in the feature space. An excessively large value for parameter γ results in over-fitting, while a disproportionately small value leads to under-fitting. Grid search (90) is the most common method to determine appropriate values for C and γ . Values for parameters C and γ that lead to the highest classification accuracy rate in this interval can be found by setting appropriate values for the upper and lower bounds (the search interval) and the jumping interval in the search. Nevertheless, this approach is a local search method, and vulnerable to local optima. Additionally, setting the search interval is a problem. Too large a search interval wastes computational resource, while too small a search interval might render a satisfactory outcome impossible.

In addition to the commonly used grid search approach, other techniques are employed in SVMs to improve the possibility of a correct choice of parameter values. Pai and Hong (91) proposed an SA-based approach to obtain parameter values for SVMs, and applied it in real data; however, this approach does not address feature selection, and therefore may exclude the optimal result. As well as the two parameters C and γ , other factors, such as the quality of the feature's dataset, may influence the classification accuracy rate. For instance, the correlations between features influence the classification result. Accidental removal of important features might lower the classification accuracy rate. Additionally, some dataset features may have no influence at all, or may contain a high level of noise. Removing such features can improve the searching speed and accuracy rate.

It is worth underlining that the kernel-based implementation of SVMs involves the problem of the selection of multiple parameters, including the kernel parameters (e.g., the γ and p parameters for the Gaussian and polynomial kernels, respectively) and the regularization parameters C . Studies have also illustrated that a radial basis kernel yields the best results in remote sensing applications (92, 93). We chose to use the radial basis kernel for SVMs in this study. The verification of the

applicability of other specialized kernel functions for the classification of remote sensing data may be used in future studies.



CHAPTER 3

RESEARCH METHODOLOGY

3.1 Subjects

The training and testing data sets of NO and H₂O₂ concentration were obtained from fifteen lacunar stroke patients (age 65.6 ± 13.61 years old) and sixteen young healthy controls (age 27.33 ± 3.85 years old). In this study, breath-holding for 30 seconds without a drop of oxygen saturation (SaO₂) were used for CVR test composing of basal, experiment and recovery phases. All subject gave inform sign the consent to participate, which approved by the local ethical committee, faculty of medicine, Thammasat University, Thailand (MTU-EC-PH-6-076155) (94).

3.2 The Assessment of endothelial dysfunction

3.2.1 Interpretation of nitric oxide (NO) level

3.2.1.1 Method for the measurement of NO

Endothelial dysfunction was evaluated by NO sensor. Electrochemistry method is well suited to measuring NO production released into a vascular lumen which based on the direct oxidation of NO to nitrite. NO reaction is oxidized on carbon-fiber microelectrode as shown in the following reaction.



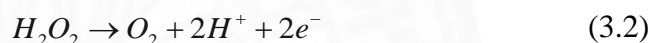
This electrochemistry method is depended on a nitrite level which NO is electrochemical reactive species that are oxidized on the surface of carbon-fiber microelectrode. Quantitation of NO concentration was assessed by an amino-700 probe (inNO nitric oxide measuring system, model inNO-T S/N 3782-G, Tampa, FL, USA). The electric current (pA) was measured and showed linear proportion to NO concentration. Usually, sensitivity of measurement was approximately about 20 nM/L and microelectrode probe was calibrated by calibration solution containing 2 ml of 1 M

H₂SO₄, 20 mg (potassium iodide, KI), and 18 ml of distill water and nitrite standard. The standard of calibration set by the microelectrode probe soaked into calibration solution and set background of sensor to zero. Nitrite standard was added into calibration solution at 50, 100 and 200 nM, respectively. The reaction of nitric oxide measuring system showed the proportion of nitrite and NO was a ratio of one to one (95).

3.2.2 Interpretation of hydrogen peroxide (H₂O₂) level

3.2.2.1 Method for the measurement of H₂O₂

H₂O₂ in the plasma diffuse through probe membrane and is oxidized given electrical signal that generates as proportion to free radicals concentration in the sample. H₂O₂ oxidation on membrane probe is following:



Quantitation of H₂O₂ level concentration was assessed by HP-250 hydrogen peroxide sensor probe (H₂O₂ measuring system, model inNO-T S/N 3782-G, Tampa, FL, USA). The electrical current (pA) was measured and showed Linear proportion to H₂O₂ concentration. Usual y, sensitivity of measurement was approximately 150-1,000 pA and microelectrode probe was calibrated by calibration solution containing 20 ml of 0.1M Phosphate Buffer Solution (PBS) solution, pH 7.4 and then added each of 10 μM/L of 500 μM of H₂O₂ standard.

3.3 Data standardization and normalization

When approaching data for modeling, some standard procedures should be used to prepare the data for modeling:

1. First the data should be filtered, and any outliers removed from the data
2. The data should be normalized or standardized to bring all of the variables into proportion with one another. For example, if one variable is 100 times larger than another (on average), then your model may be better behaved if you normalize/standardize the two variables to be approximately equivalent. Technically

though, whether normalized/standardized, the coefficients associated with each variable will scale appropriately to adjust for the disparity in the variable sizes. However, if normalized/standardized, then the coefficients will reflect meaningful relative activity between each variable (i.e., a positive coefficient will mean that the variable acts positively towards the objective function, and vice versa, plus a large coefficient versus a small coefficient will reflect the degree to which that variable influences the objective function. Whereas the coefficients from un-normalized/un-standardized data will reflect the positive/negative contribution towards the objective function, but will be much more difficult to interpret in terms of their relative impact on the objective function.

3. Non-numeric qualitative data should be converted to numeric quantitative data, and normalized/standardized. For example, if a survey question asked an interviewee to select where the economy will be for the next six months (i.e., deep recession, moderate recession, mild recession, neutral, mild recovery, moderate recovery, or strong recovery), these can be converted to numerical values of 1 through 7, and thus quantified for the model.

To normalize data, traditionally this means to fit the data within unity, so all data values will take on a value of 0 to 1. Since some models collapse at the value of zero, sometimes an arbitrary range of say 0.1 to 0.9 is chosen instead, but for this post I will assume a unity-based normalization. The following equation is what should be used to implement a unity-based normalization:

$$x_{i,0,1} = \frac{\delta_i - \delta_{\min}}{\delta_{\max} - \delta_{\min}} \quad (3.3)$$

where:

δ_i = each data point i

δ_{\max} = the maxima among all the data points

δ_{\min} = the minima among all the data points

$x_{i,0,1}$ = the data point i normalized between 0 and +1

If you desire to have a more centralized set of normalized data, with zero being the central point, then the following equation can be used instead to normalize your data:

$$x_{i,-1+1} = \frac{\delta_i - \left(\frac{\delta_{\max} + \delta_{\min}}{2}\right)}{\left(\frac{\delta_{\max} - \delta_{\min}}{2}\right)} \quad (3.4)$$

where:

δ_i = each data point i

δ_{\max} = the maxima among all the data points

δ_{\min} = the minima among all the data points

$x_{i,-1+1}$ = the data point i normalized between -1 and +1

Finally, to standardize your data, you will want the data to reflect how many standard deviations from the average that that data lies, with the following normal distribution curve representing the probability of each standard deviation for a normal distribution (this graphic is borrowed from Wikipedia). The Z-Score is what will be calculated to standardize the data, and it reflects how many standard deviations from the average that the data point falls (96-99).

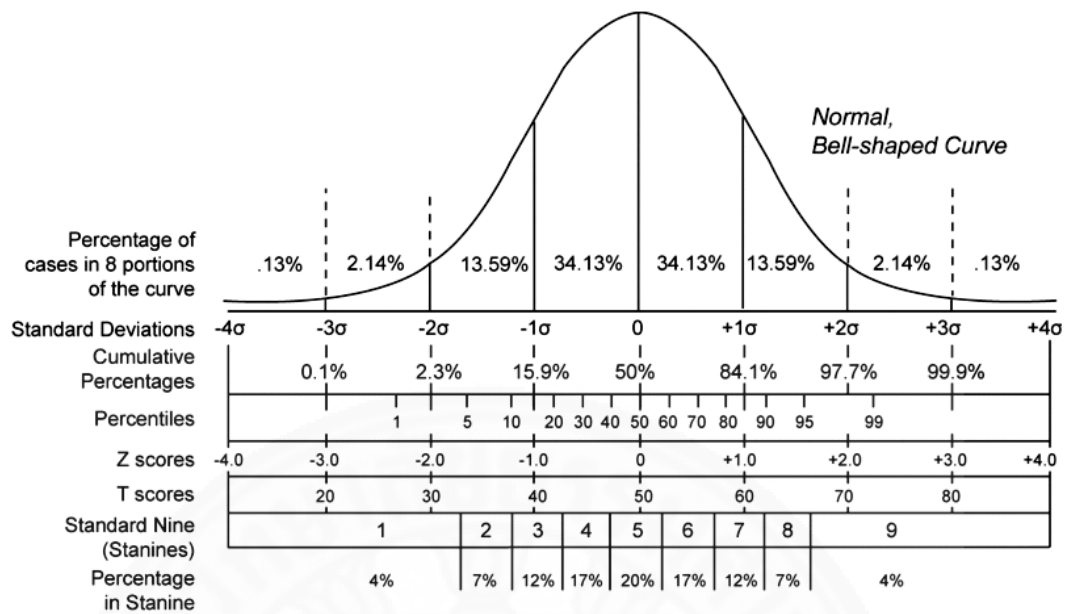


Figure 3.1 Illustration of normal distribution.

To determine the Z-Score of each data point, the following equation should be used:

$$x_{i,1\sigma} = \frac{x_i - \bar{x}_s}{\sigma_{x,s}} \quad (3.5)$$

where:

$x_{i,1\sigma}$ = the data point i standardized to 1σ , also known as Z-Score

x_i = Each data point i

\bar{x}_s = The average of all the sample data points

$\sigma_{x,s}$ = The sample standard deviation of all sample data point

3.4 Experiment procedure

The performance of SVMs depends on the different type of kernel functions, because each single kernel functions that demonstrates the different characteristics and limitations in application. Regarding physiological processes of NO and H₂O₂ synthesis, release to the circulation and reach the equilibrium state of each cycle in the circulation, the best fit explaining dynamic process should be from

integration function model. Look at the model, if we can integrate two or more single kernel functions to construct hybrid kernel function and give the superiority among the single ones, it should be fitted with NO. On the other hand, the performance of the hybrid kernel function will be worse than the single ones because of the excessive parameters that we combine them. Therefore, the most important step is search for the right proportions of each parameter of single kernel functions in order to construct the best performance for hybrid kernel functions.

For kernel function selection, we were matching kernel function for specified tasks are extremely for SVMs classifier performance. Previously, there have been reported that RBF kernel function is best for classification problems. It can be well adapted under many conditions, low-dimension, high-dimension, small sample or even large sample, etc. Since RBF has a wider convergence domain, thus it is satisfactory basis for the classification of the function. Regarding polynomial kernel, it has been devised as being alternative to a usual nonlinear SVMs training algorithms and it also is better suitable for prediction as well. The sigmoid kernel is quite popular for support vector machines due to its origin that derived from neural network.

A novel integration of hybrid kernel function, we use all functions, including single linear kernel, polynomial kernel, RBF kernel and sigmoid kernel for training these sample data and then evaluate the performance of each kernel function. The next step, we combine single polynomial, RBF and sigmoid kernels to construct a novel integration of hybrid kernel function. The general form of hybrid kernel function can be expressed as follows:

$$k_{hkf}(x, y) = \eta_1(x^T y + \beta)^d + \eta_2 \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) + \eta_3 \tanh(\gamma(x^T \cdot y) + \alpha) \quad (3.6)$$

Consider the general form of Equation 4.7 is more complex. Therefore, in accordance with each part of the kernel functions, it is also separated as a feature, the parameter selection, and valuation. The optimal parameters are divided into two steps to complete as follows.

1) First, we prepared preliminary data sets for two-class classification. In order to get better accuracy, data sets are normalized to the range $[-1, +1]$.

The training data x_i is classified to either of the two classes depending on the outcome of the following operation:

$$w^T x_i + b \geq 1, \text{ for all } y_i = +1 \quad (3.7)$$

$$w^T x_i + b \leq -1, \text{ for all } y_i = -1 \quad (3.8)$$

where $y_i=+1$ or -1 represent positive class and negative class respectively, w is the weight vector and b is the bias (or $-b$ is the threshold). With the decision rule given by

$$f_{w,b}(x) = \text{sgn}(w^T x + b) \quad (3.9)$$

the function $f(\cdot)$ in its simplest form is the sign or step function. Next, to initialize range and value of the parameters of the single kernel function (β, d, σ, γ and α) and assume that $\eta_1=\eta_2=\eta_3=1$ then evaluate the performance of each parameters to find the best performance of hybrid kernel function.

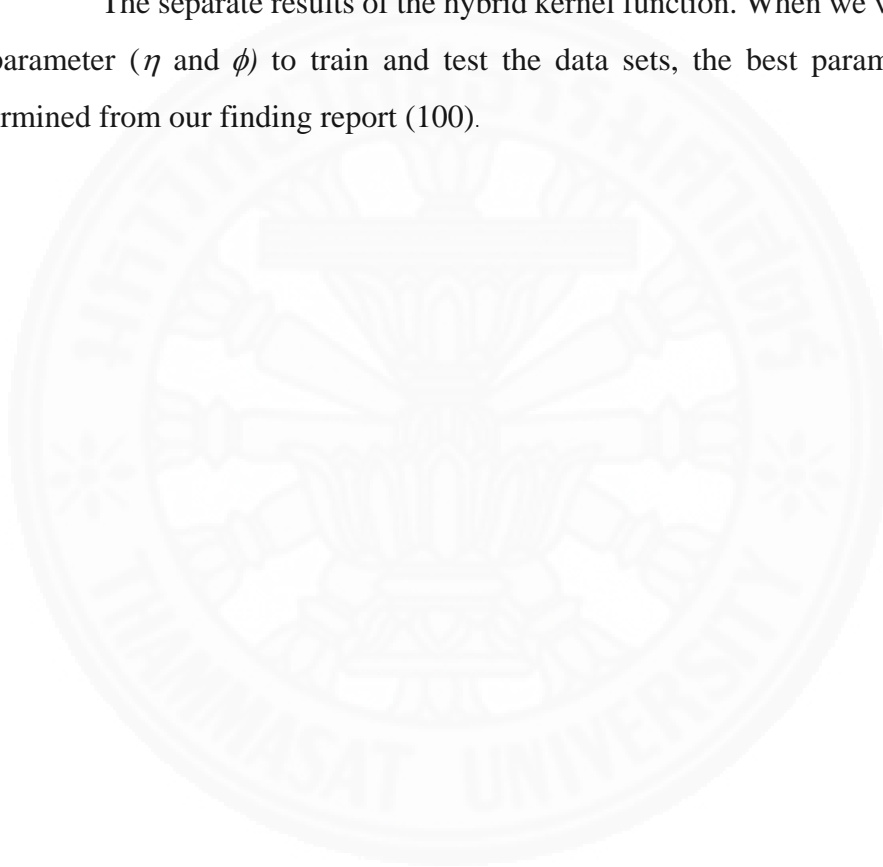
2) Finally, to assume that $\eta_1=1, \eta_2=\eta_3=0$ or in other words such as a single polynomial kernel function, parameter, β and d are selected for further running. Then, by simulation, experiments could be the performance of SVMs with the trend of changes, thus, we can choose the best parameter β and d of the polynomial kernel. Similarly, we can choose the best parameter σ of the RBF and the best parameter γ and α of the sigmoid kernel function by suppose that $\eta_1=\eta_3=0, \eta_2=1$ and $\eta_1=\eta_2=0, \eta_3=1$. Then, a good best parameters has been identified (β, b, σ, ρ and α) into the hybrid kernel function, and then regard which single kernel function in the hybrid kernel function is more important, which is demonstrated in the parameters (η_1, η_2 and η_3) the proportion of the above. The experiments have been repeated many times, for the best results of hybrid kernel function.

From findings results, use the best parameters to construct the hybrid kernel function for training the samples. The hybrid kernel function is:

$$k_{hkf}(x, y) = \eta(x^T \cdot y + 1)^3 + \phi \left(\frac{e^{2(x^T \cdot y + 3)} + 1}{e^{2(x^T \cdot y + 3)} - 1} \right) \quad (3.10)$$

where η and ϕ are user-defined parameters.

The separate results of the hybrid kernel function. When we vary the value of parameter (η and ϕ) to train and test the data sets, the best parameters can be determined from our finding report (100).



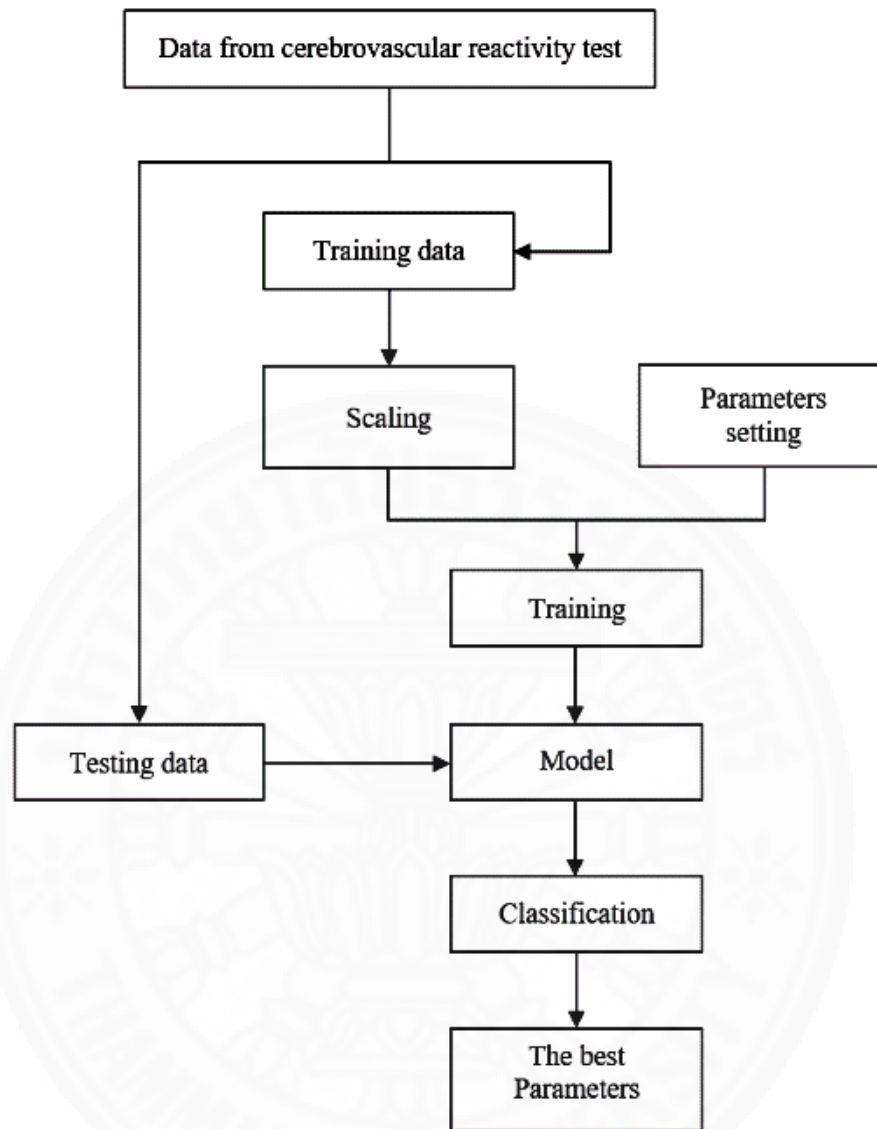


Figure 3.2 The experimental procedure.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Finding

In this study, we investigate the performance of classification models by the support vector machines algorithms of NO and H₂O₂ obtained from healthy control and lacunar stroke. NO and H₂O₂ were measured by electrochemistry method and gain high sensitivity with high accuracy above 90%. We used changes of plasma NO and H₂O₂ level for assessment of high-risk lacunar stroke.

4.1.1 To determine Classifier performance of nitric oxide (NO) by each single kernel function

As shown in table 4.1, the comparative data of NO level in control and patient subjects in each phase were tested for comparison of classifier performance of each kernel function during experiment.

Table 4.1 Comparative data of NO concentration in control and patient subjects

| Phase | | NO concentration ($\mu\text{M/L}$) | |
|------------|---------|--------------------------------------|---------------------|
| | | control | patient |
| Basal | Max | 82.20 | 67.40 |
| | Min | 23.30 | 21.60 |
| | Average | <u>58.06</u> | <u>39.00</u> |
| Experiment | Max | 105.30 | 63.60 |
| | Min | 78.00 | 18.20 |
| | Average | <u>73.11</u> | <u>40.21</u> |
| Recovery | Max | 98.00 | 68.8 |
| | Min | 28.40 | 24.6 |
| | Average | <u>63.63</u> | <u>38.7</u> |

4.1.1.1 The results of polynomial kernel function for NO assessment

Table 4.2 Classifier performance of NO by polynomial kernel function in basal phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Polynomial | $\beta=1, d=2$ | 88.00* | 7 |
| | $\beta=1, d=5$ | 76.00 | 18 |
| | $\beta=5, d=2$ | 78.00 | 36 |
| | $\beta=1, d=3$ | 86.00 | 9 |
| | $\beta=1, d=4$ | 74.00 | 28 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 3.9.

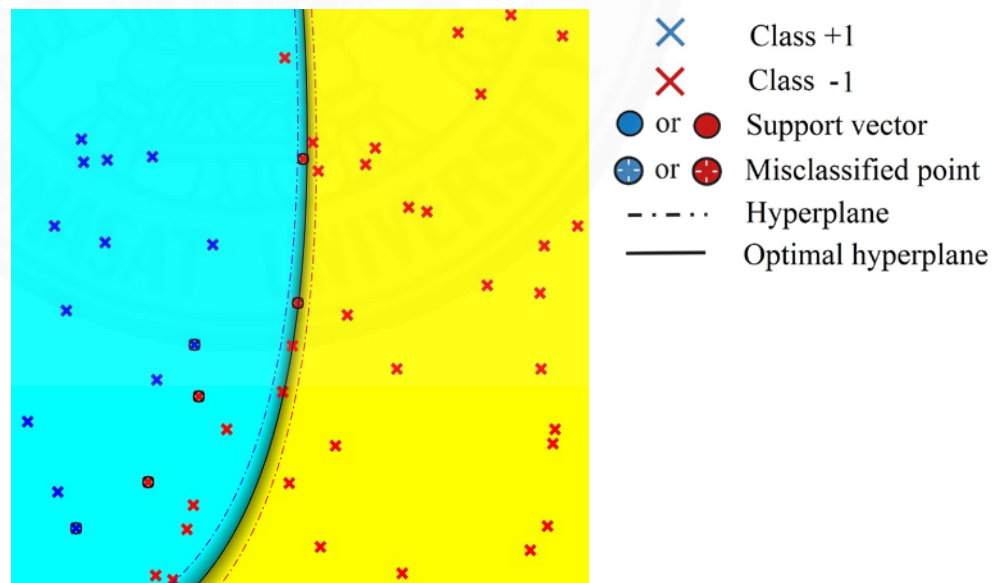


Figure 4.1 Separation of NO data assessed by polynomial kernel function in basal phase.

Table 4.3 Classifier performance of NO by polynomial kernel function in experiment phase

| Kernel Function | The experimental results | | |
|-----------------|-----------------------------|-------------------------|---------------------|
| | Parameter [†] □ | Classified accuracy (%) | No. support vectors |
| Polynomial | $\beta=1, d=5$ | 80 | 8 |
| | $\beta=5, d=2$ | 78 | 27 |
| | $\beta=3, d=3$ | 90* | 9 |
| | $\beta=3, d=4$ | 78 | 7 |
| | $\beta=4, d=4$ | 86 | 33 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.12.

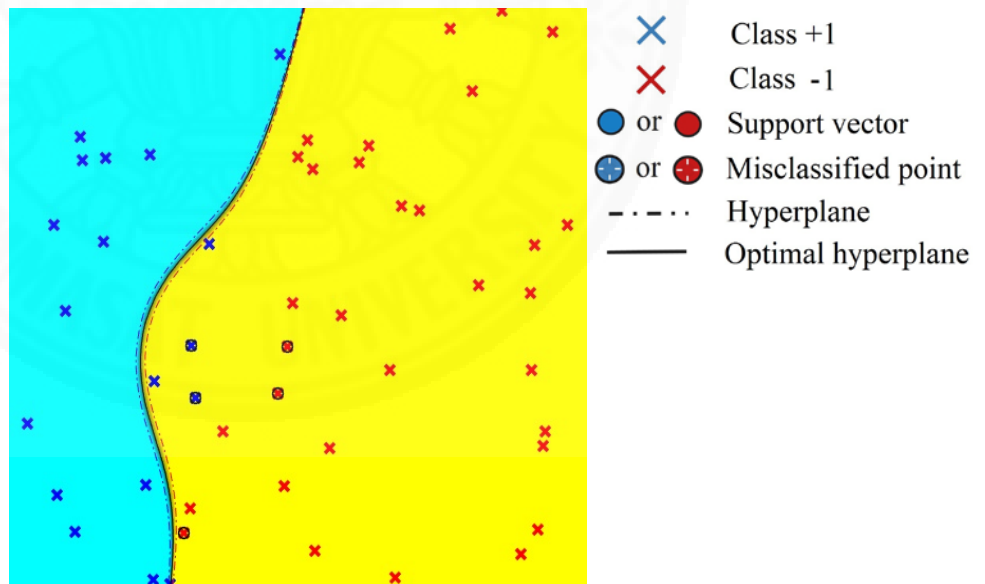


Figure 4.2 Separation of NO data assessed by polynomial kernel function in experiment phase.

Table 4.4 Classifier performance of NO by polynomial kernel function in recovery phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Polynomial | $\beta=1, d=5$ | 84.00* | 8 |
| | $\beta=5, d=2$ | 78.00 | 35 |
| | $\beta=2, d=3$ | 82.00 | 16 |
| | $\beta=5, d=3$ | 72.00 | 14 |
| | $\beta=3, d=4$ | 76.00 | 33 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.12.

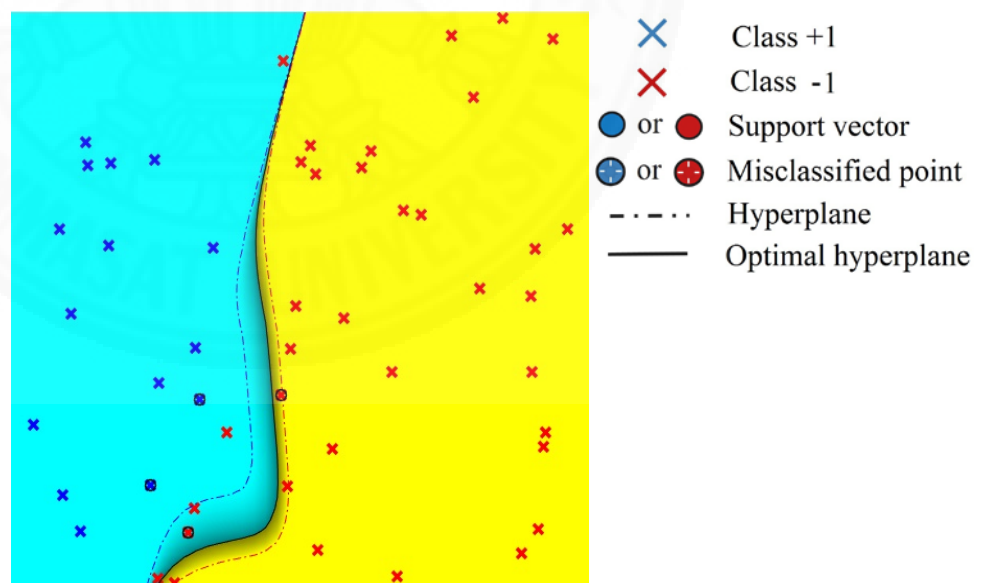


Figure 4.3 Separation of NO data assessed by polynomial kernel function in recovery phase.

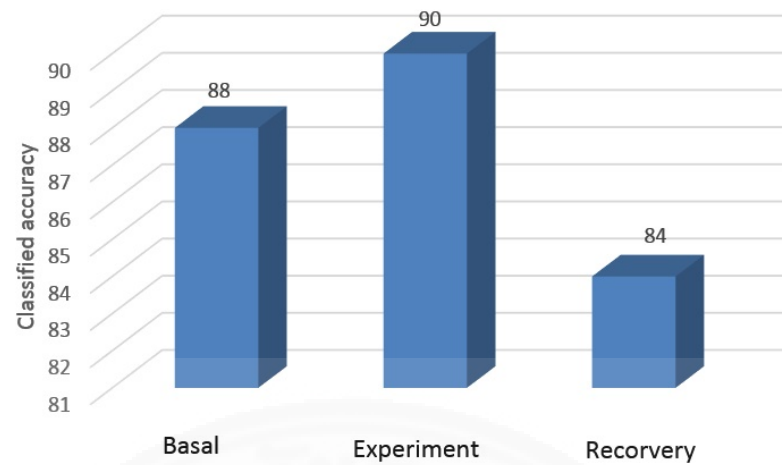


Figure 4.4 Comparison of best classifier performance of NO by polynomial kernel function in each phase of CVR test.

From above experiment data, a highest classified accuracy is 90.00 from polynomial in experiment phase and is followed by 88.00 and 84.00 from polynomial kernel function in recovery and basal phase, respectively. According to, the best parameters of polynomial kernel functions in each phase, it is able to construct the new high performance kernel function for training the samples data as well. Considering the best parameters in table 4.2- 4.4 and most experimental results, they have shown classifier performance of NO by polynomial kernel function is high when parameter β is odd numbers.

4.1.1.2 The results of Gaussian radial basis function (RBF) kernel function for NO assessment

Table 4.5 Classifier performance of NO by RBF kernel function in basal phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| RBF | $\sigma=2$ | 66.00 | 50 |
| | $\sigma=3$ | 76.00 | 50 |
| | $\sigma=4$ | 78.00 | 44 |
| | $\sigma=5$ | 84.00 | 39 |
| | $\sigma=6$ | 88.00* | 32 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.13.

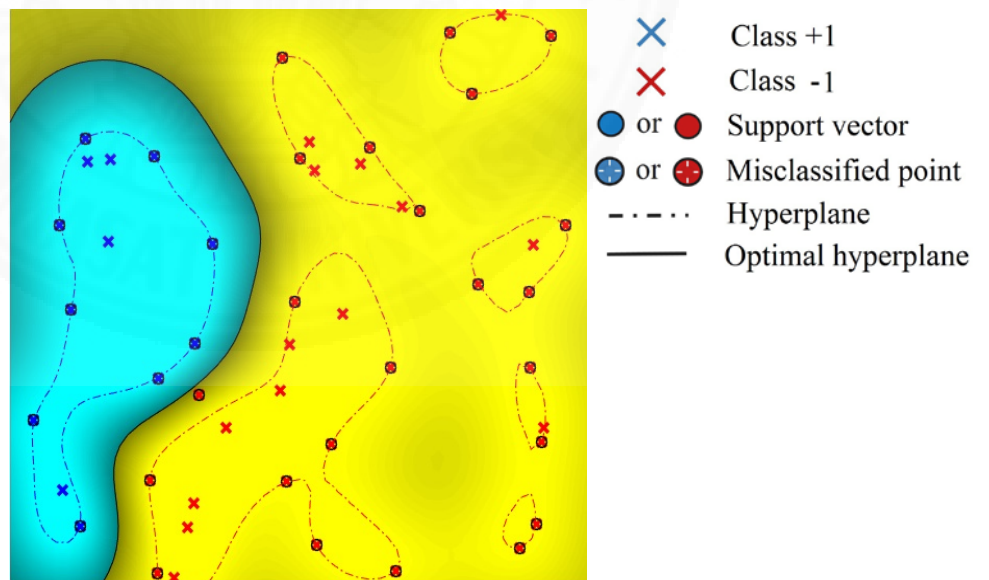


Figure 4.5 Separation of NO data assessed by RBF kernel function in basal phase.

Table 4.6 Classifier performance of NO by RBF kernel function in experiment phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| RBF | $\sigma=1$ | 92.00 | 50 |
| | $\sigma=4$ | 94.00* | 42 |
| | $\sigma=7$ | 90.00 | 34 |
| | $\sigma=9$ | 90.00 | 26 |
| | $\sigma=15$ | 88.00 | 23 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.13.

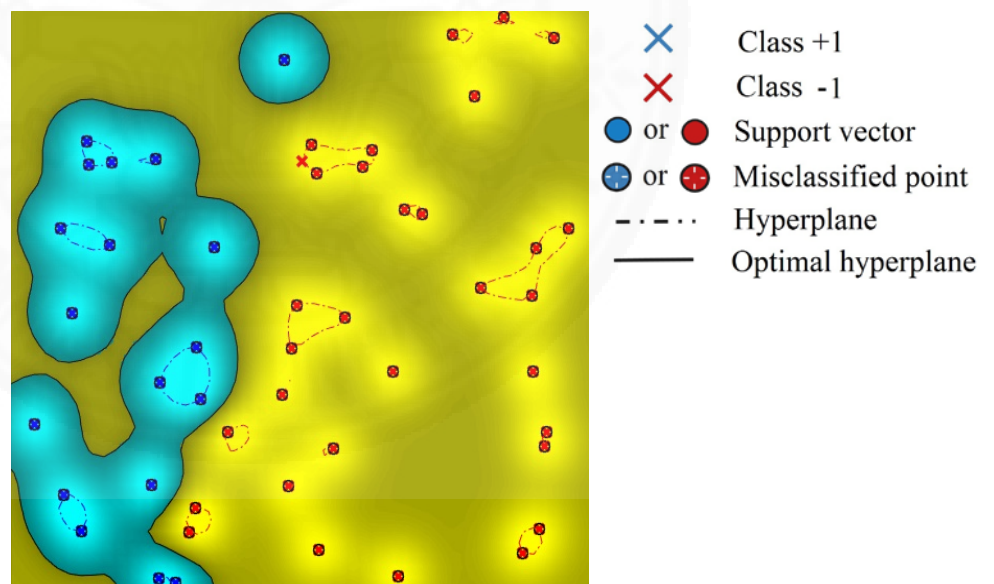


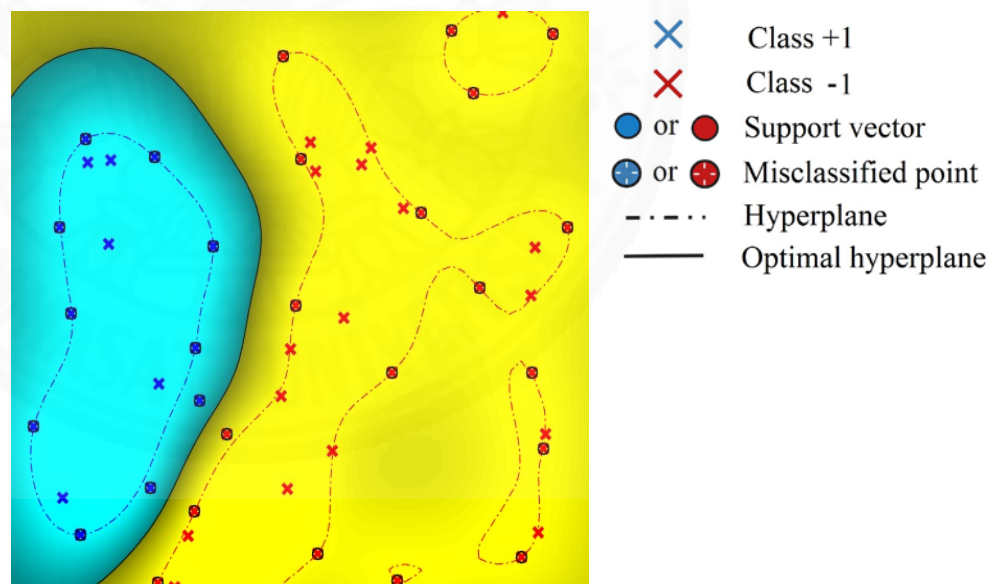
Figure 4.6 Separation of NO data assessed by RBF kernel function in experiment phase.

Table 4.7 Classifier performance of NO by RBF kernel function in recovery phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| RBF | $\sigma=1$ | 72.00 | 50 |
| | $\sigma=2$ | 76.00 | 50 |
| | $\sigma=3$ | 80.00 | 50 |
| | $\sigma=4$ | 82.00 | 50 |
| | $\sigma=8$ | 84.00* | 31 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.13.



Phase 4.7 Separation of NO data assessed by polynomial kernel function in recovery phase.

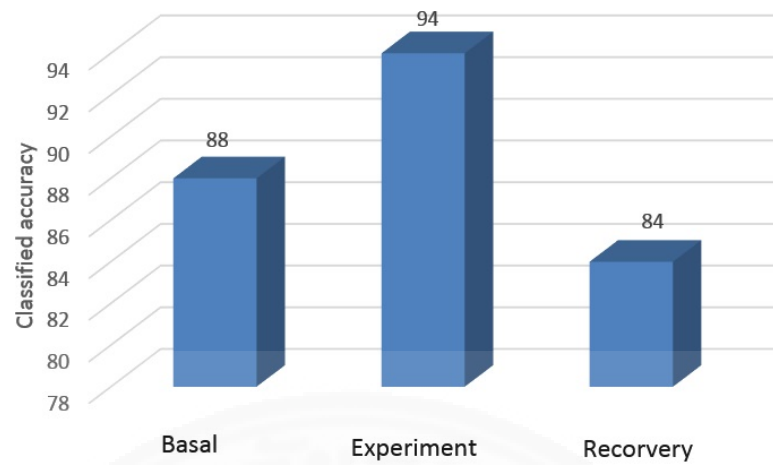


Figure 4.8 Comparison of best classifier performance for NO of RBF kernel function in each phase of CVR test.

All findings studied by RBF kernel function, the classification results of RBF in experiment phase is the best, followed by the RBF kernel function in basal and recovery phase, respectively. Then we choose the parameter σ to test the sample, the optimal parameters have been present in Table 4.5-4.7 and Figure 4.5-4.7 showing the separate results of the kernel function. The classifier performance of NO by RBF kernel is high when parameter σ is even number.

4.1.1.3 The results of sigmoid kernel function for NO assessment

Table 4.8 Classifier performance of NO by sigmoid kernel function in basal phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Sigmoid | $\gamma=1, \alpha=1$ | 74.00* | 26 |
| | $\gamma=1, \alpha=2$ | 66.00 | 29 |
| | $\gamma=1, \alpha=4$ | 68.00 | 24 |
| | $\gamma=2, \alpha=2$ | 72.00 | 32 |
| | $\gamma=2, \alpha=3$ | 68.00 | 30 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.14.

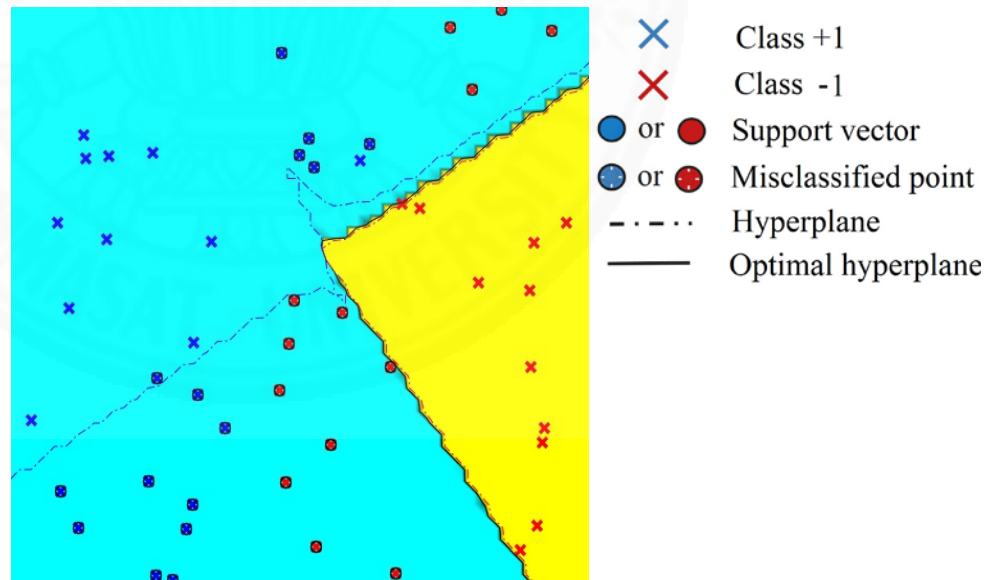


Figure 4.9 Separation of NO data assessed by sigmoid kernel function in basal phase.

Table 4.9 Classifier performance of NO by sigmoid kernel function in experiment phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Sigmoid | $\gamma=1, \alpha=2$ | 54.00 | 8 |
| | $\gamma=1, \alpha=3$ | 58.00* | 12 |
| | $\gamma=1, \alpha=5$ | 56.00 | 45 |
| | $\gamma=3, \alpha=3$ | 54.00 | 27 |
| | $\gamma=3, \alpha=5$ | 52.00 | 16 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.14.

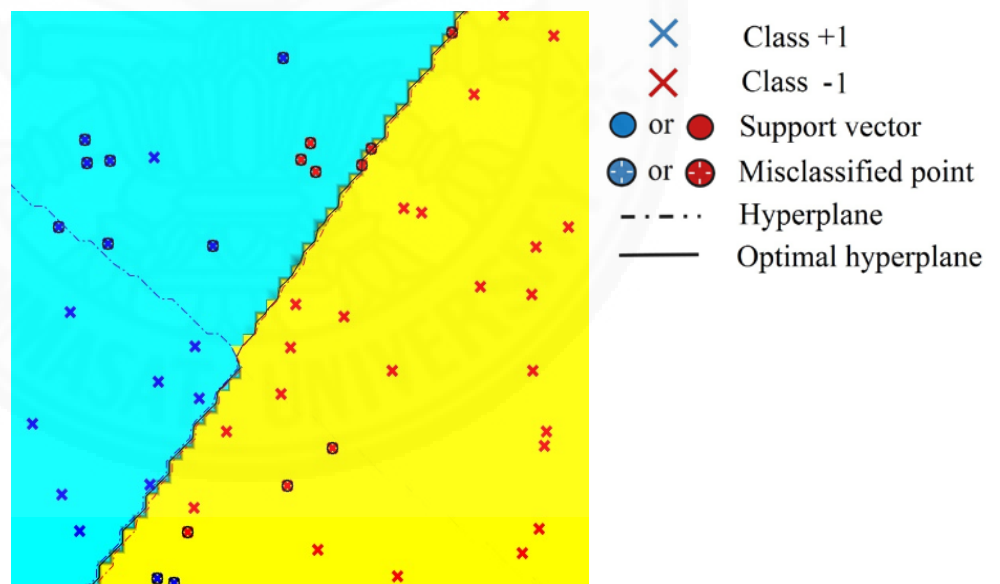


Figure 4.10 Separation of NO data assessed by polynomial kernel function in experiment phase.

Table 4.10 Classifier performance of NO by sigmoid kernel function in recovery phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Sigmoid | $\gamma=1, \alpha=2$ | 78.00* | 26 |
| | $\gamma=1, \alpha=4$ | 58.00 | 26 |
| | $\gamma=2, \alpha=1$ | 46.00 | 27 |
| | $\gamma=2, \alpha=3$ | 52.00 | 26 |
| | $\gamma=2, \alpha=4$ | 48.00 | 27 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.14.

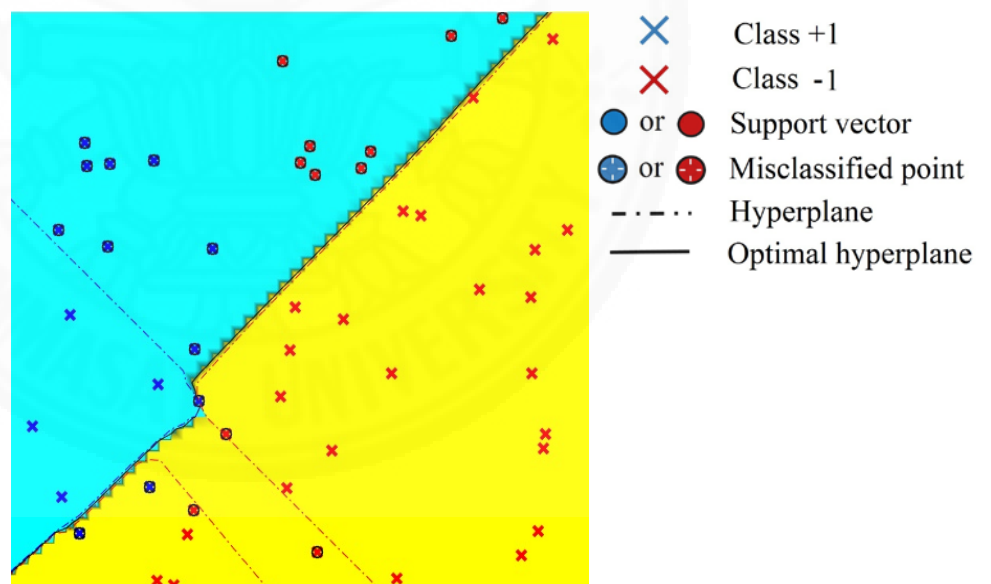


Figure 4.11 Separation of NO data assessed by polynomial kernel function in recovery phase.

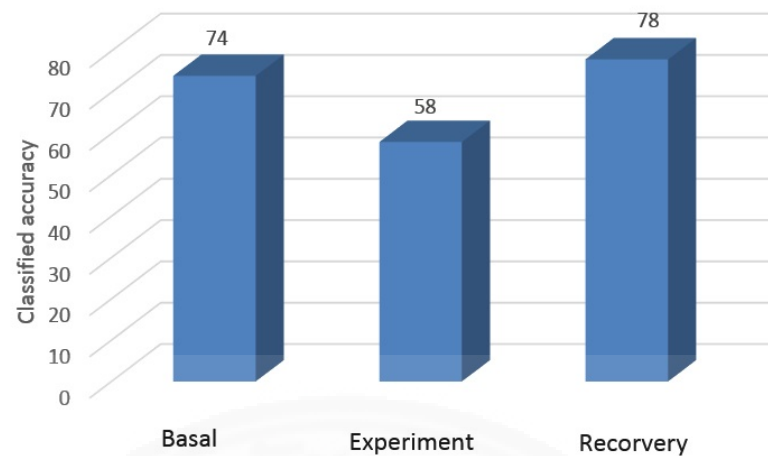


Figure 4.12 Comparison of best classifier performance of NO by sigmoid kernel function in each phase of CVR test.

From experiment results, they have shown low performance of the most sigmoid kernel function with highest classified accuracy is 78.00 in recovery phase and is followed by 74.00 and 58.00 in basal and experiment phase, respectively. Considering the best parameters in table 4.8-4.10 and most experimental results have shown that classifier performance of NO by sigmoid kernel function is high when the sum of parameter γ and α is odd numbers.

4.1.2 An improvement of classifier performance of NO assessment

According to the previous experiments, the best parameters of each kernel functions are used to construct the hybrid kernel function for improvement classifier performance for NO assessment.

Table 4.11 Classifier performance of NO by hybrid kernel function in basal phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Hybrid | $\eta=1, \phi=3$ | 82.00 | 19 |
| | $\eta=1, \phi=4$ | 86.00 | 18 |
| | $\eta=1, \phi=5$ | 88.00 | 11 |
| | $\eta=0.1, \phi=0.4$ | 90.00 | 6 |
| | $\eta=0.5, \phi=0.5$ | 94.00* | 6 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 3.10.

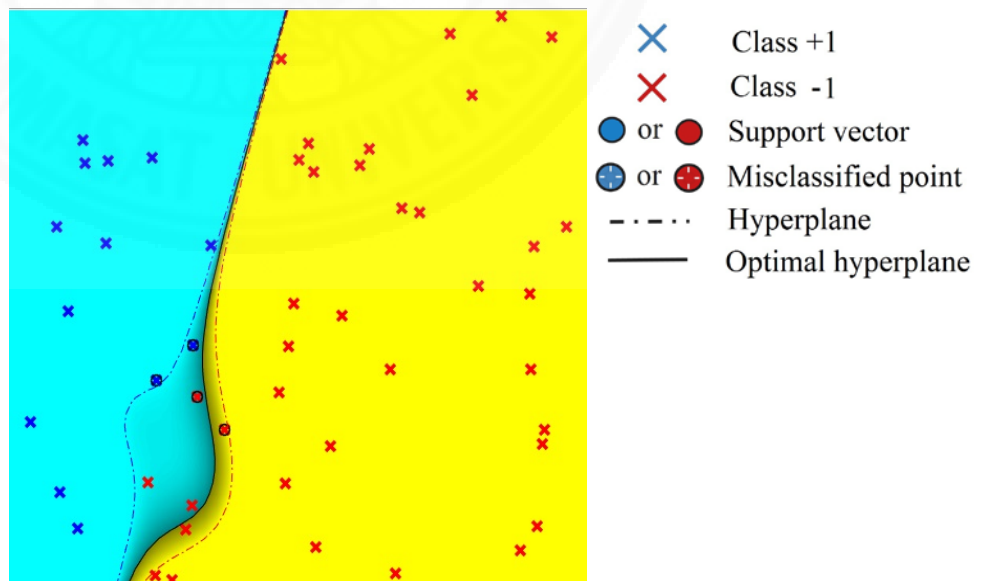


Figure 4.13 Separation of NO data assessed by hybrid kernel function in basal phase.

Table 4.12 Classifier performance of NO by hybrid kernel function in experiment phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Hybrid | $\eta=1, \phi=1$ | 92.00 | 16 |
| | $\eta=0.1, \phi=0.1$ | 92.00 | 14 |
| | $\eta=0.1, \phi=0.3$ | 92.00 | 15 |
| | $\eta=0.3, \phi=0.5$ | 90.00 | 12 |
| | $\eta=0.4, \phi=0.5$ | 96.00* | 17 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 3.10.

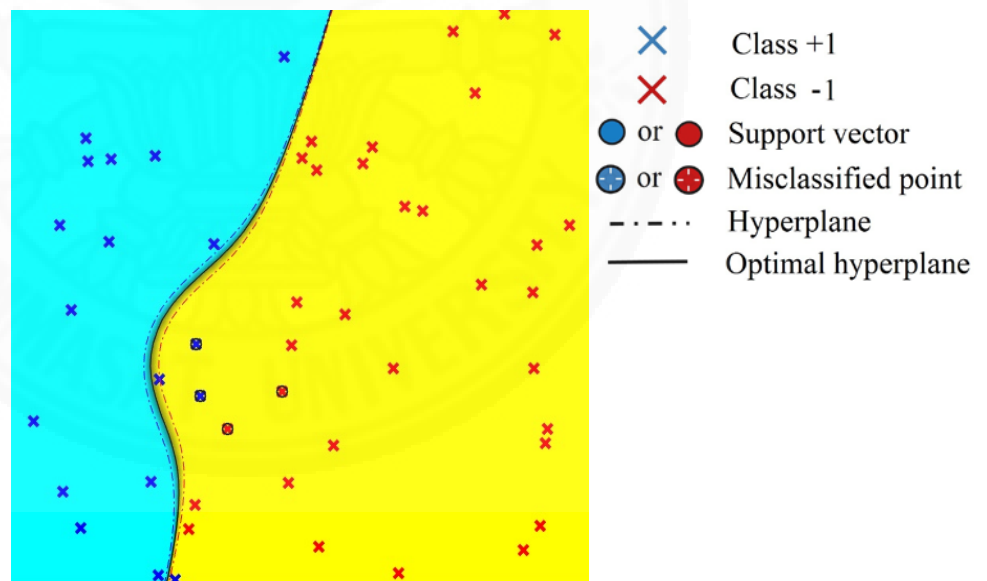


Figure 4.14 Separation of NO data assessed by hybrid kernel function in experiment phase.

Table 4.13 Classifier performance of NO by hybrid kernel function in recovery phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Hybrid | $\eta=1, \phi=4$ | 78.00 | 15 |
| | $\eta=0.1, \phi=0.2$ | 76.00 | 14 |
| | $\eta=0.1, \phi=0.5$ | 80.00 | 11 |
| | $\eta=0.3, \phi=0.5$ | 92.00 | 7 |
| | $\eta=0.5, \phi=0.5$ | 92.00* | 4 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 3.10.

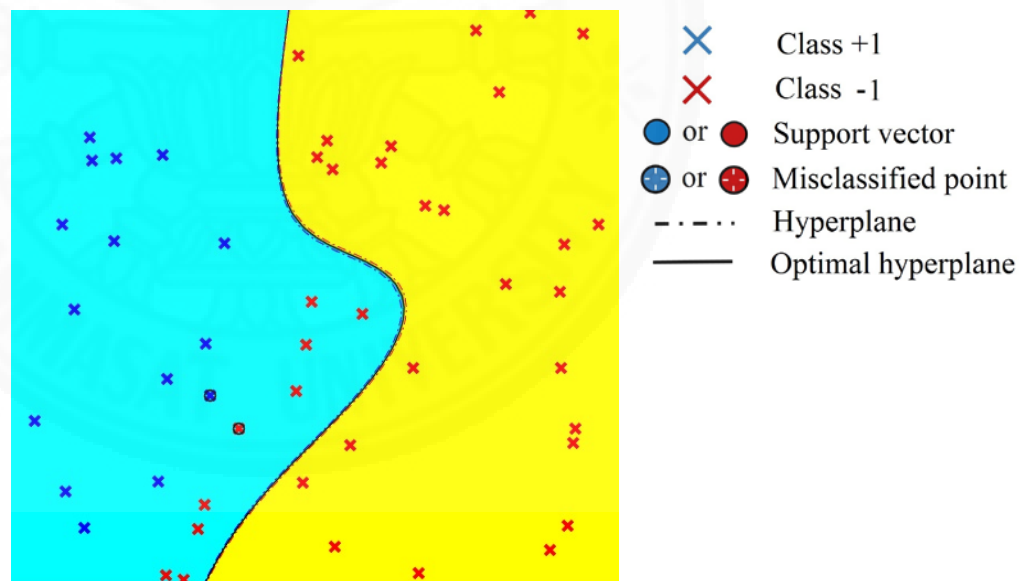


Figure 4.15 Separation of NO data assessed by hybrid kernel function in recovery phase.

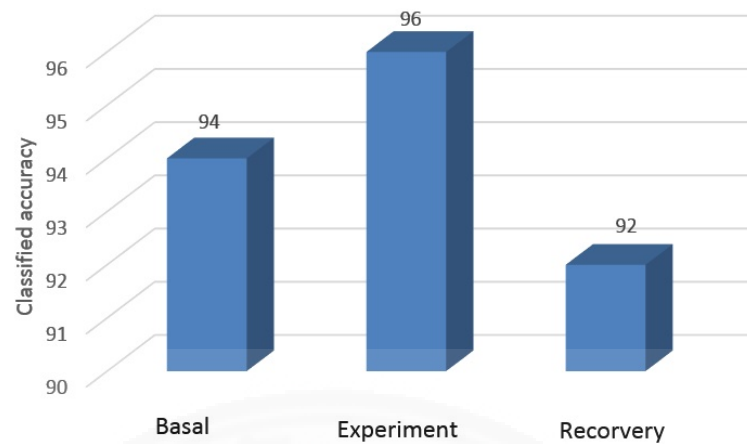


Figure 4.16 Comparison of best classifier performance of NO by hybrid kernel function in each phase of CVR test.

Figure 4.13-4.15 show the separate results of the hybrid kernel function. When we vary the value of parameter (η and ϕ) to test the data sets in each phrase, the best parameters determined are present in Table 4.11-4.13. The best values of both parameters (η and ϕ) resulting in the highest performance for classification of nitric oxide for hybrid kernel function are 0.5. Consider the values of parameters compared with highest performance, low value of parameters is detected.

4.1.3 Comparison of best performance of hybrid kernel function and single kernel function for NO assessment

Table 4.14 Comparison of best performance of hybrid kernel function and single kernel function in basal phase for NO assessment

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Polynomial | $\beta=1, d=2$ | 88.00 | 7 |
| RBF | $\sigma=6$ | 88.00 | 32 |
| Sigmoid | $\gamma=1, \beta=1$ | 74.00 | 26 |
| Hybrid | $\eta=0.5, \phi=0.5$ | 94.00* | 6 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.12, 2.13, 2.14 and 3.10.

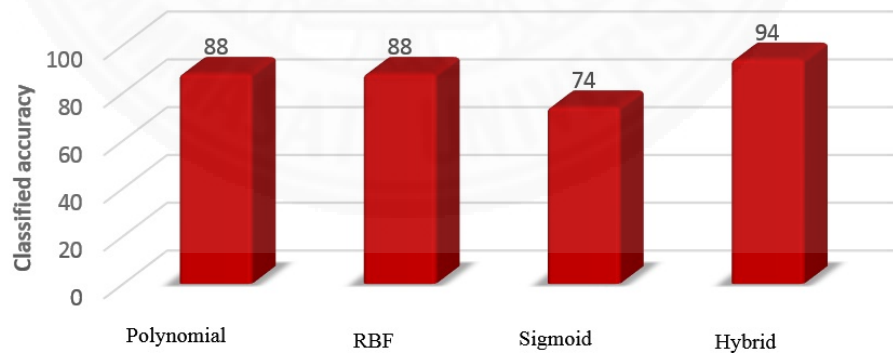


Figure 4.17 Comparison chart of best performance of hybrid kernel function and single kernel function in basal phase for NO assessment.

Table 4.15 Comparison of best performance of hybrid kernel function and single kernel unction in experiment phase for NO assessment

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Polynomial | $\beta=3, d=3$ | 90.00 | 9 |
| RBF | $\sigma=4$ | 94.00 | 42 |
| Sigmoid | $\gamma=1, \beta=3$ | 58.00 | 12 |
| Hybrid | $\eta=0.4, \phi=0.5$ | 94.00* | 17 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.12, 2.13, 2.14 and 3.10.

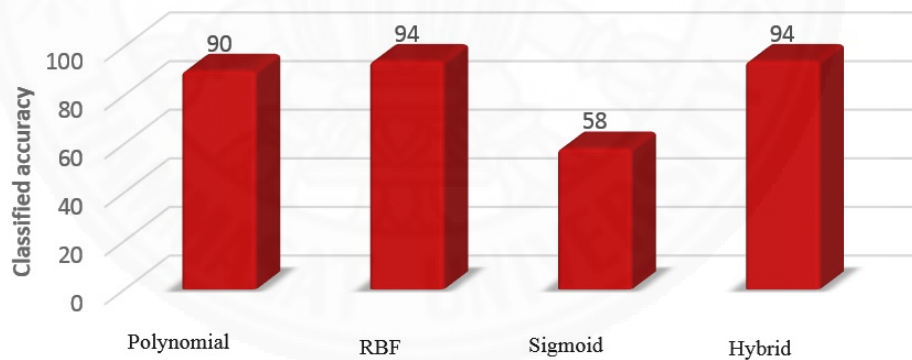


Figure 4.18 Comparison chart of best performance hybrid kernel function and single kernel function in experiment phase for NO assessment.

Table 4.16 Comparison of best performance hybrid kernel function and single kernel function in recovery phase for NO assessment

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Polynomial | $\beta=1, d=5$ | 84.00 | 8 |
| RBF | $\sigma=8$ | 84.00 | 31 |
| Sigmoid | $\gamma=1, \beta=2$ | 78.00 | 26 |
| Hybrid | $\eta=0.5, \phi=0.5$ | 92.00* | 4 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.12, 2.13, 2.14 and 3.10.

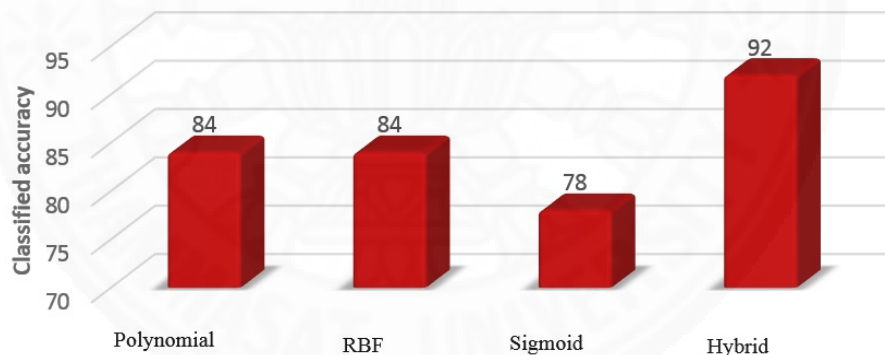


Figure 4.19 Comparison chart of best performance hybrid kernel function and single kernel function in recovery phase for NO assessment.

From all findings result in Table 4.14-4.16 Figure 4.17-4.19, they suggest that parameters η and ϕ are able to improve performance of NO classification in lacunar stroke.

4.1.4 To determine classifier performance of hydrogen peroxide (H₂O₂) by each single kernel function

As shown in table 4.17, the data set of H₂O₂ level in control and patient subjects in each phase were tested for classifier performance of each kernel function during experiment.

Table 4.17 Comparative data of H₂O₂ concentration in control and patient subjects

| phase | | H ₂ O ₂ concentration (μM/L) | |
|------------|---------|--|---------------------|
| | | control | patient |
| Basal | Max | 15.70 | 12.02 |
| | Min | 6.00 | 9.68 |
| | Average | <u>10.00</u> | <u>10.91</u> |
| Experiment | Max | 13.40 | 12.35 |
| | Min | 5.90 | 7.92 |
| | Average | <u>9.30</u> | <u>9.57</u> |
| Recovery | Max | 14.20 | 8.90 |
| | Min | 5.80 | 7.41 |
| | Average | <u>9.40</u> | <u>8.52</u> |

4.1.4.1 The results of polynomial kernel function for H₂O₂ assessment

Table 4.18 Classifier performance of H₂O₂ by polynomial kernel function in basal phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Polynomial | $\beta=1, d=3$ | 56.00 | 5 |
| | $\beta=1, d=4$ | 68.00 | 7 |
| | $\beta=3, d=3$ | 64.00 | 5 |
| | $\beta=4, d=2$ | 72.00* | 7 |
| | $\beta=3, d=4$ | 66.00 | 4 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.12.

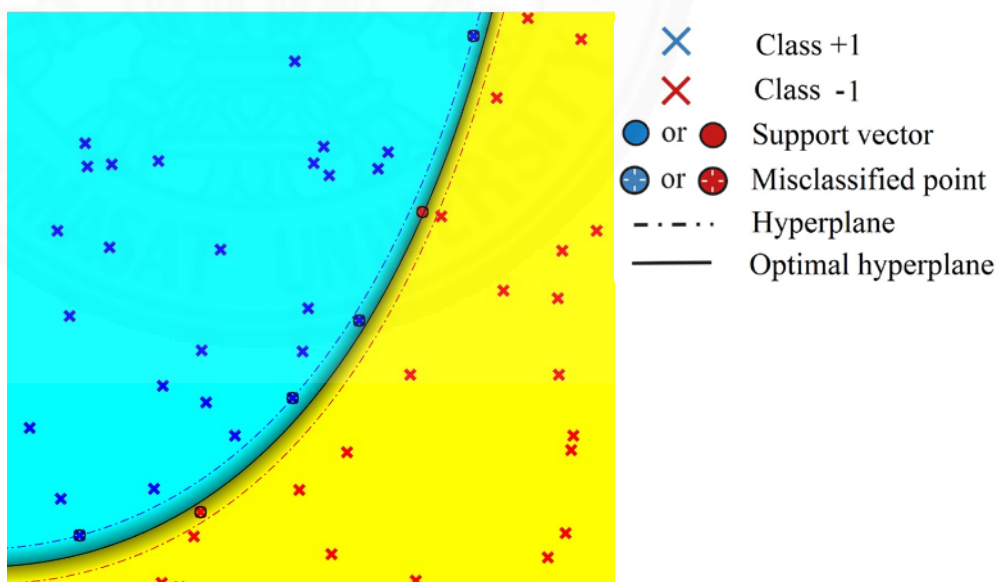


Figure 4.20 Separation of H₂O₂ data assessed by polynomial kernel function in basal phase.

Table 4.19 Classifier performance of H₂O₂ by polynomial kernel function in experiment phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Polynomial | $\beta=2, d=2$ | 76.00 | 6 |
| | $\beta=4, d=2$ | 80.00* | 5 |
| | $\beta=5, d=2$ | 78.00 | 5 |
| | $\beta=3, d=4$ | 66.00 | 4 |
| | $\beta=4, d=4$ | 76.00 | 14 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.12.

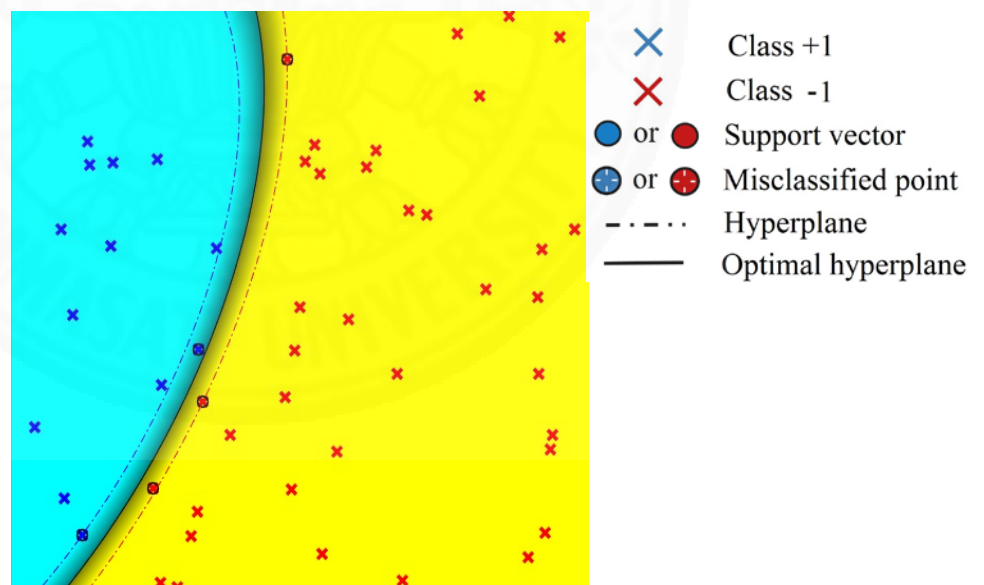


Figure 4.21 Separation of H₂O₂ data assessed by polynomial kernel function in experiment phase.

Table 4.20 Classifier performance of H₂O₂ by polynomial kernel function in recovery phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Polynomial | $\beta=1, d=3$ | 36 | 19 |
| | $\beta=1, d=4$ | 46 | 16 |
| | $\beta=3, d=2$ | 38 | 4 |
| | $\beta=2, d=3$ | 44 | 5 |
| | $\beta=4, d=3$ | 50* | 5 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.12.

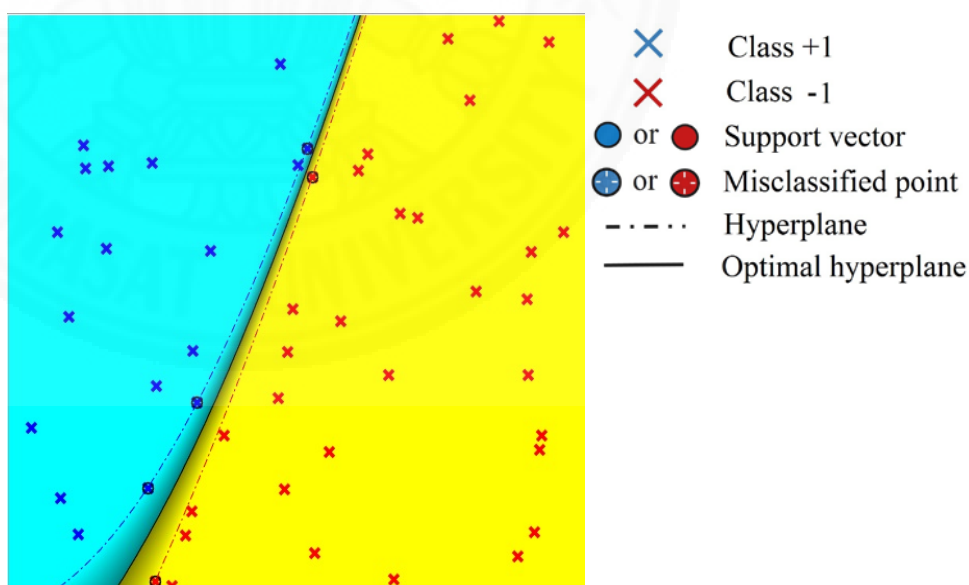


Figure 4.22 Separation of H₂O₂ data assessed by polynomial kernel function in recovery phase.

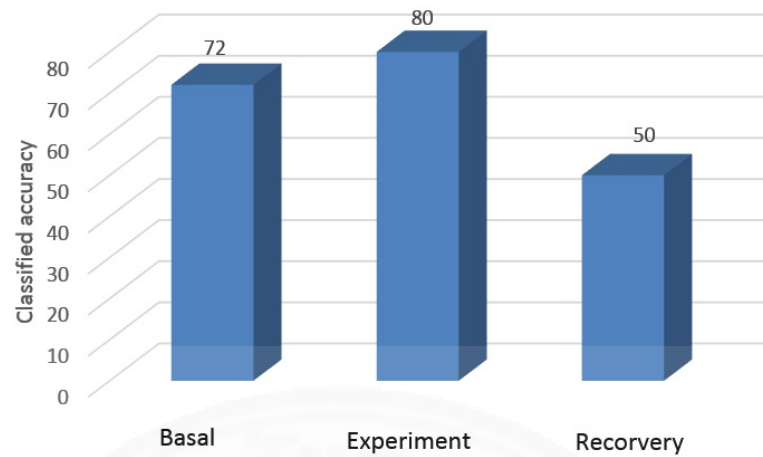


Figure 4.23 Comparison of best classifier performance for H₂O₂ of polynomial kernel function in each phase of CVR test.

All findings results are present in Table 4.14-4.16 and Figure 4.20-4.22. Figure 4.23 shows percentage of classified accuracy of each phase in CVR test. It demonstrates a highest classified accuracy is 80.00% in experiment phase and is followed by 72.00% and 50.00% in basal and recovery phase, respectively. Compare to NO, classifier performance of H₂O₂ by polynomial kernel function lower than those NO of all phase.

4.1.4.2 The results of Gaussian radial basis function (RBF) kernel function for H₂O₂ assessment

Table 4.21 Classifier performance of H₂O₂ by RBF kernel function in basal phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| RBF | $\sigma=1$ | 56 | 23 |
| | $\sigma=2$ | 72 | 13 |
| | $\sigma=3$ | 86* | 13 |
| | $\sigma=4$ | 68 | 17 |
| | $\sigma=6$ | 74 | 6 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.13.

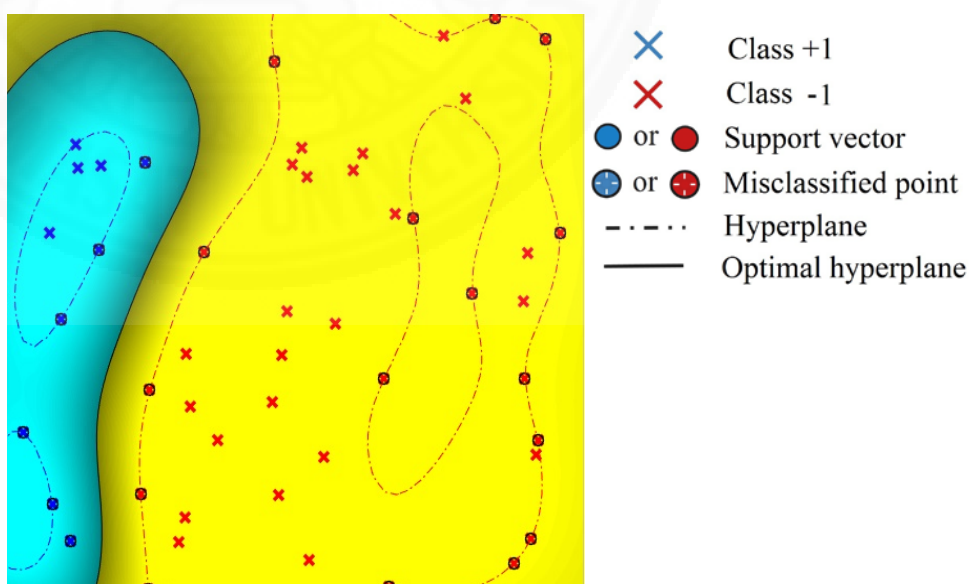


Figure 4.24 Separation of H₂O₂ data assessed by RBF kernel function in basal phase.

Table 4.22 Classifier performance of H₂O₂ by RBF kernel function in experiment phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| RBF | $\sigma=2$ | 78 | 13 |
| | $\sigma=3$ | 82 | 16 |
| | $\sigma=4$ | 90* | 10 |
| | $\sigma=5$ | 84 | 10 |
| | $\sigma=10$ | 80 | 6 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.13.

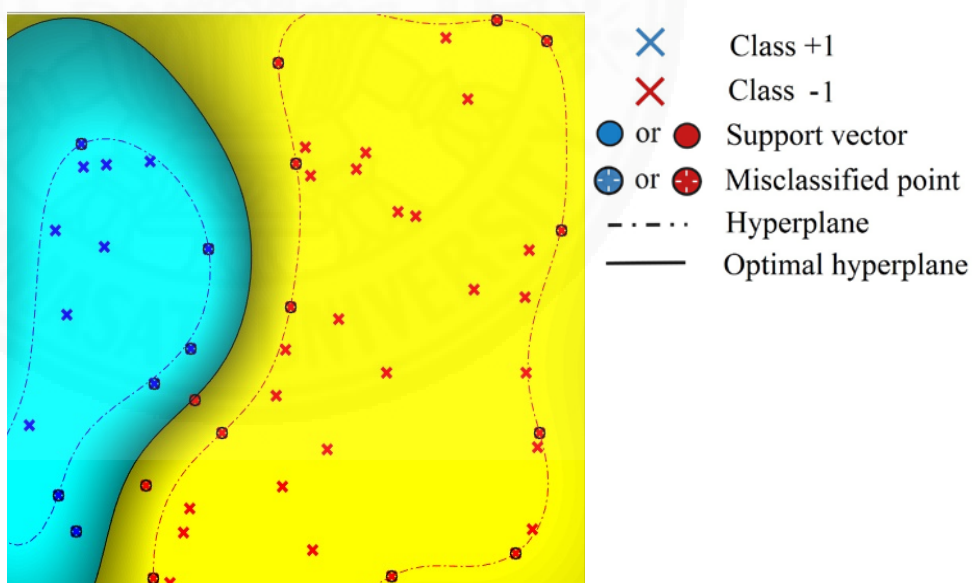


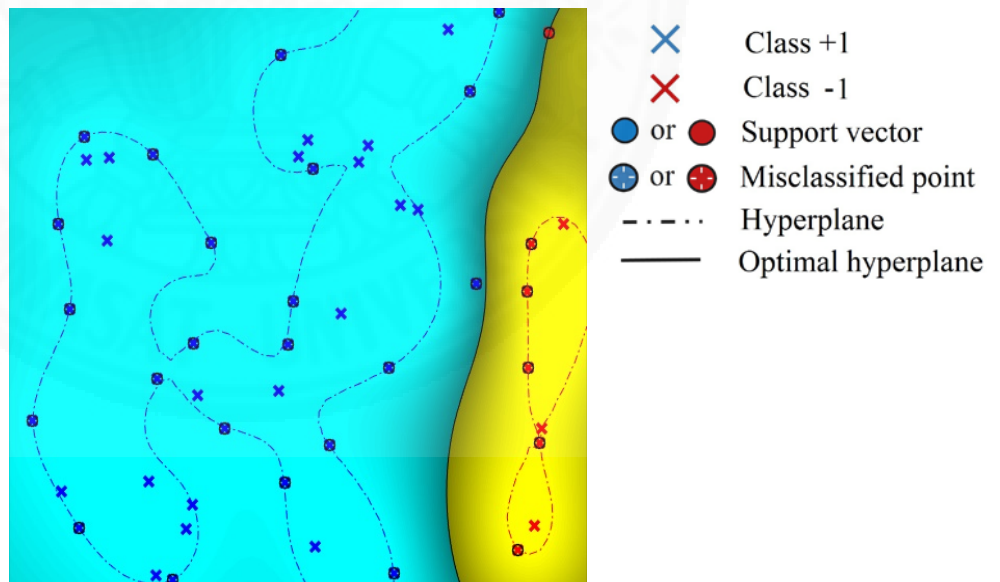
Figure 4.25 Separation of H₂O₂ data assessed by RBF kernel function in experiment phase.

Table 4.23 Classifier performance of H₂O₂ by RBF kernel function in recovery phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| RBF | $\sigma=1$ | 56 | 50 |
| | $\sigma=4$ | 64 | 50 |
| | $\sigma=11$ | 64 | 36 |
| | $\sigma=12$ | 76* | 31 |
| | $\sigma=14$ | 62 | 30 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.13.

Figure 4.26 Separation of H₂O₂ data assessed by RBF kernel function in recovery phase.

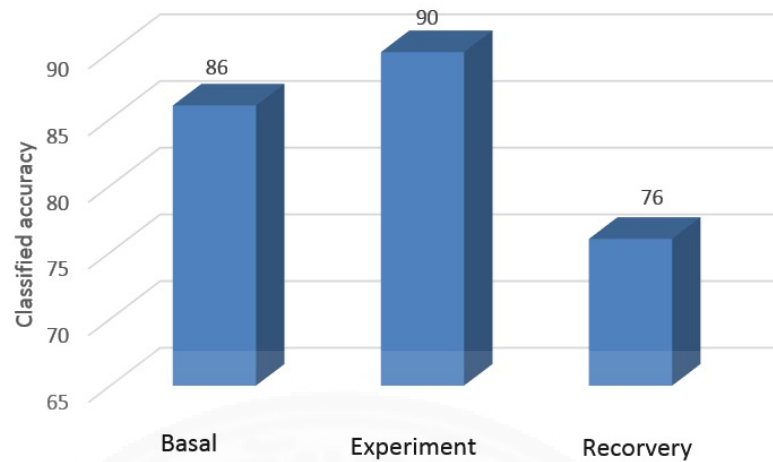


Figure 4.27 Comparison of best classifier performance of H₂O₂ by RBF kernel function in each phase of CVR test.

All findings results are present in Table 4.21-4.23 and Figure 4.24-4.26. Figure 4.23 shows percentage of classified accuracy of each phase in CVR test. It demonstrates a highest classified accuracy is 90.00% in experiment phase and is followed by 86.00% and 76.00% in basal and recovery phase, respectively. Compare to NO, classifier performance of H₂O₂ by RBF kernel function lower than those NO of all phase.

4.1.4.3 The results of sigmoid kernel function for H₂O₂ assessment

Table 4.24 Classifier performance of H₂O₂ by sigmoid kernel function in basal phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Sigmoid | $\gamma=1, \alpha=3$ | 64 | 28 |
| | $\gamma=1, \alpha=5$ | 62 | 28 |
| | $\gamma=2, \alpha=1$ | 66 | 32 |
| | $\gamma=2, \alpha=2$ | 72* | 32 |
| | $\gamma=3, \alpha=2$ | 64 | 30 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.14.

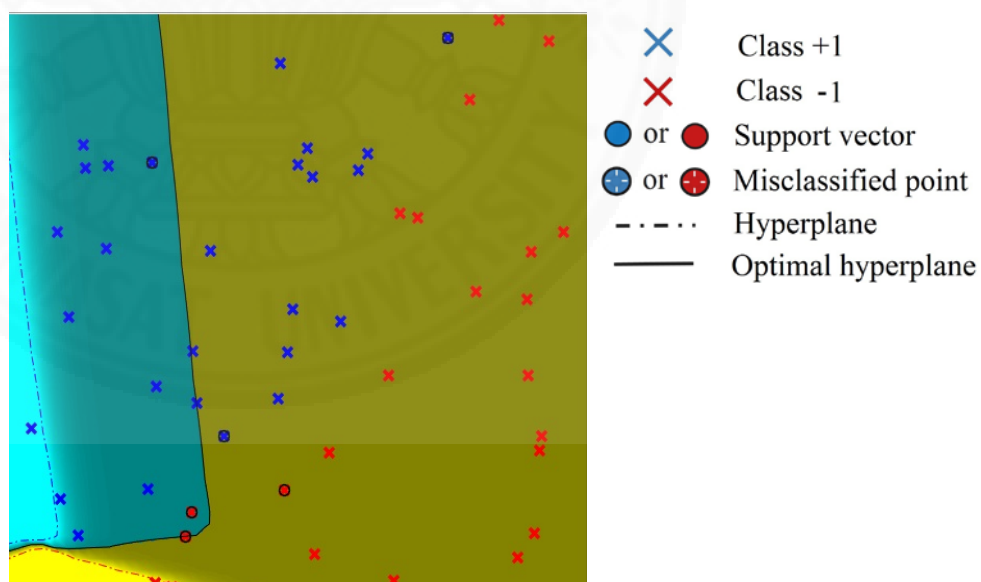


Figure 4.28 Separation of H₂O₂ data assessed by sigmoid kernel function in basal phase.

Table 4.25 Classifier performance of H₂O₂ by sigmoid kernel function in experiment phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Sigmoid | $\gamma=1, \alpha=3$ | 58.00* | 29 |
| | $\gamma=1, \alpha=5$ | 56.00 | 29 |
| | $\gamma=2, \alpha=3$ | 54.00 | 29 |
| | $\gamma=2, \alpha=5$ | 54.00 | 28 |
| | $\gamma=3, \alpha=1$ | 52.00 | 28 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.14.

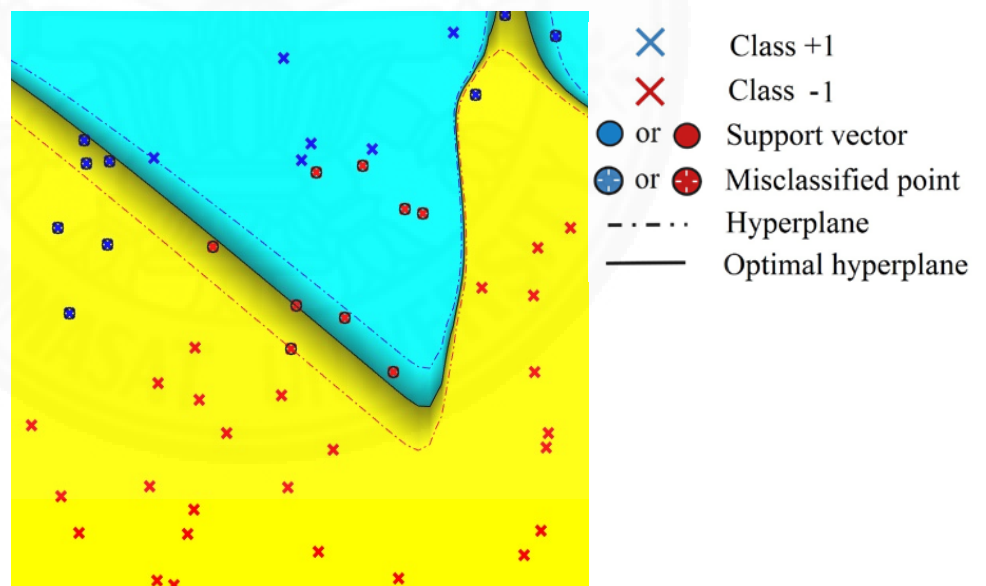


Figure 4.29 Separation of H₂O₂ data assessed by sigmoid kernel function in experiment phase.

Table 4.26 Classifier performance of H₂O₂ by sigmoid kernel function in recovery phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Sigmoid | $\gamma=1, \alpha=3$ | 52 | 26 |
| | $\gamma=2, \alpha=1$ | 46 | 27 |
| | $\gamma=2, \alpha=5$ | 54* | 26 |
| | $\gamma=3, \alpha=1$ | 48 | 27 |
| | $\gamma=3, \alpha=2$ | 46 | 27 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.14.

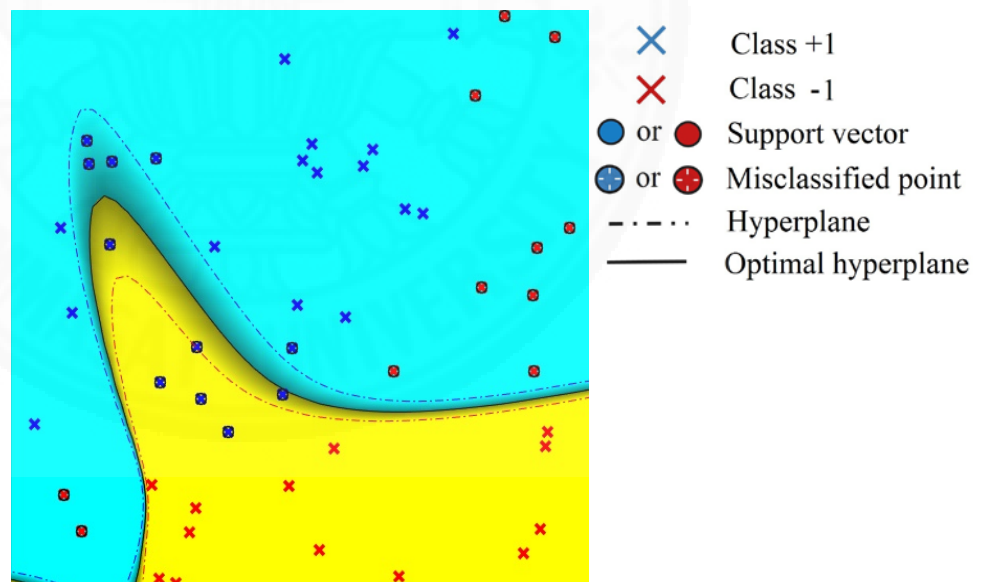


Figure 4.30 Separation of H₂O₂ data assessed by sigmoid kernel function in recovery phase.

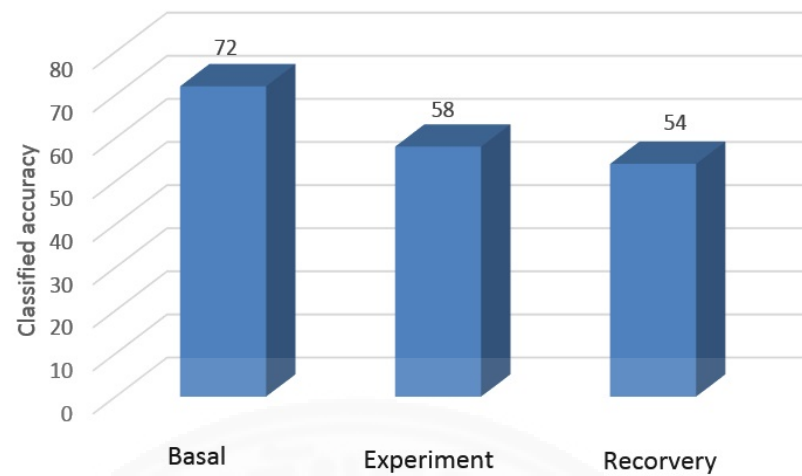


Figure 4.31 Comparison of best classifier performance for H₂O₂ of sigmoid kernel function in each phase of CVR test.

All findings results are present in Table 4.24-4.26 and Figure 4.28-4.30. Figure 4.31 shows percentage of classified accuracy of each phase in CVR test. It demonstrates a highest classified accuracy is 72.00% in basal phase and is followed by 58.00% and 54.00% in experiment and recovery phase, respectively. Compare to NO, classifier performance of H₂O₂ by sigmoid kernel function lower than those NO of all phase.

4.1.5 An improvement of classifier performance for H₂O₂ assessment

Table 4.27 Classifier performance of H₂O₂ by hybrid kernel function in basal phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Hybrid | $\eta=1, \phi=4$ | 80 | 7 |
| | $\eta=1, \phi=5$ | 88 | 9 |
| | $\eta=0.3, \phi=0.5$ | 76 | 11 |
| | $\eta=0.4, \phi=0.5$ | 86* | 9 |
| | $\eta=0.5, \phi=0.5$ | 84 | 11 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 3.10.

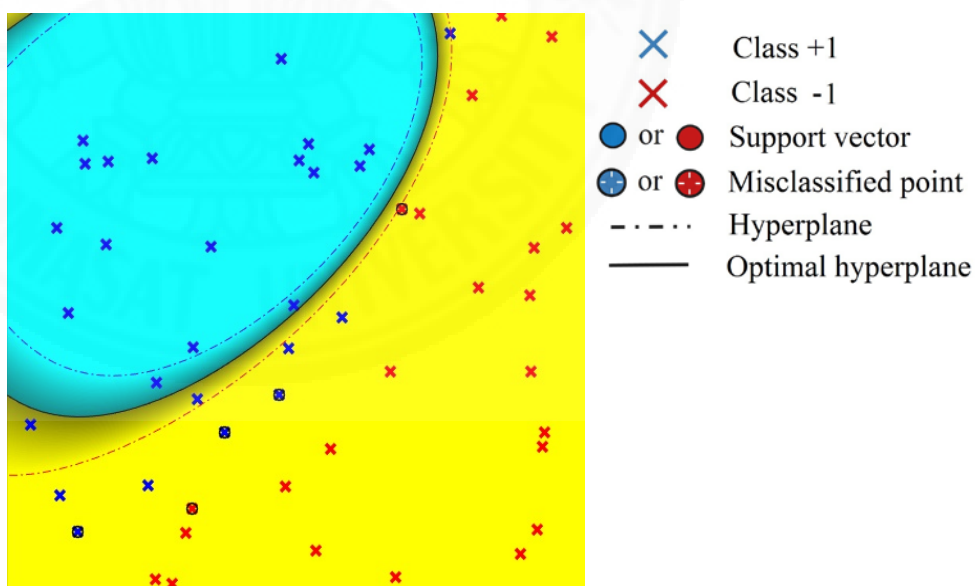


Figure 4.32 Separation of H₂O₂ data assessed by hybrid kernel function in basal phase.

Table 4.28 Classifier performance of H₂O₂ by hybrid kernel function in experiment phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Hybrid | $\eta=1, \phi=5$ | 80 | 7 |
| | $\eta=0.1, \phi=0.1$ | 92* | 11 |
| | $\eta=0.1, \phi=0.3$ | 88 | 11 |
| | $\eta=0.1, \phi=0.5$ | 82 | 11 |
| | $\eta=0.4, \phi=0.5$ | 84 | 9 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 3.10.

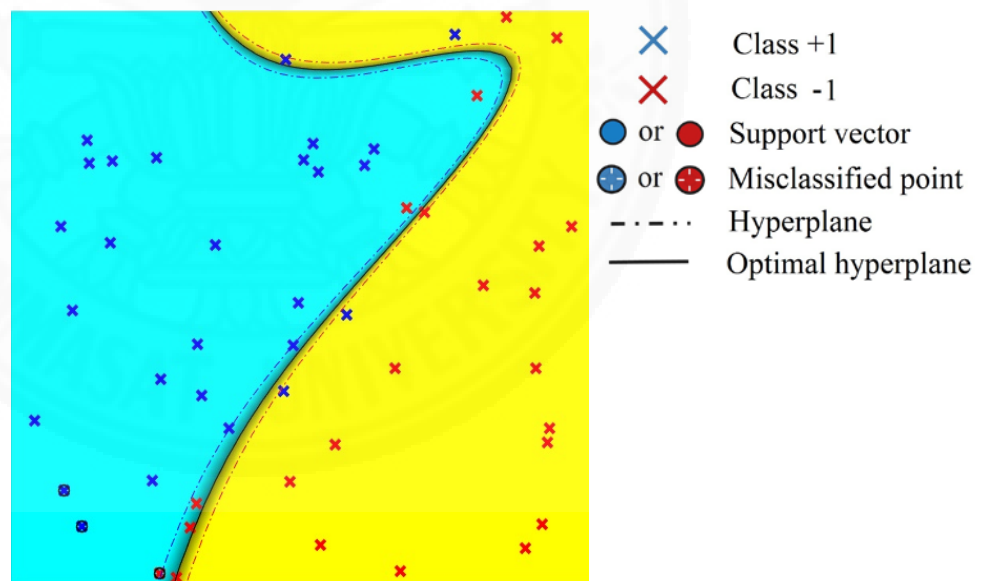


Figure 4.33 Separation of H₂O₂ data assessed by hybrid kernel function in experiment phase.

Table 4.29 Classifier performance of H₂O₂ by hybrid kernel function in recovery phase

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Hybrid | $\eta=1, \phi=2$ | 74 | 7 |
| | $\eta=1, \phi=3$ | 78 | 6 |
| | $\eta=0.1, \phi=0.1$ | 76 | 10 |
| | $\eta=0.1, \phi=0.2$ | 80 | 10 |
| | $\eta=0.1, \phi=0.3$ | 82* | 8 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 3.10.

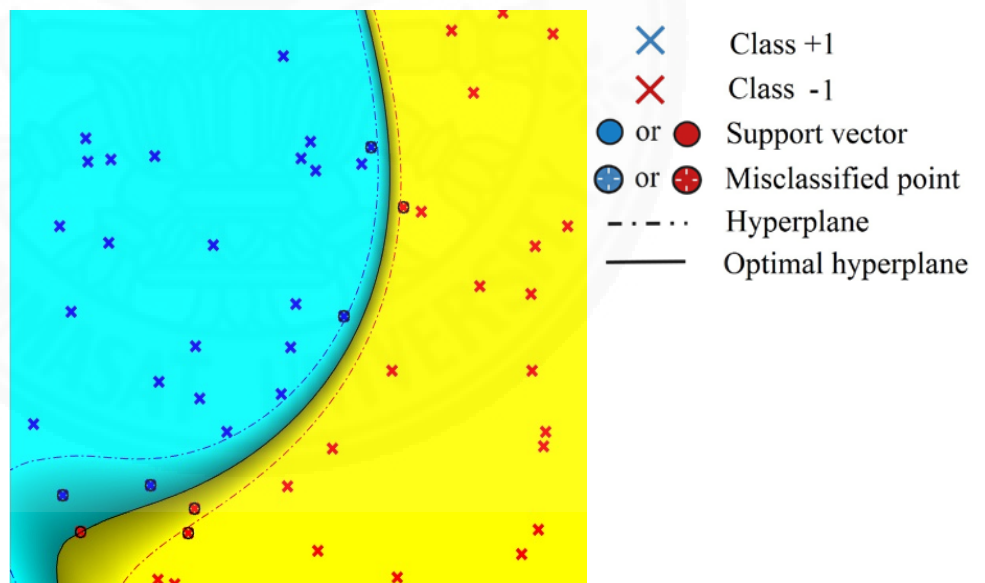


Figure 4.34 Separation of H₂O₂ data assessed by hybrid kernel function in recovery phase.

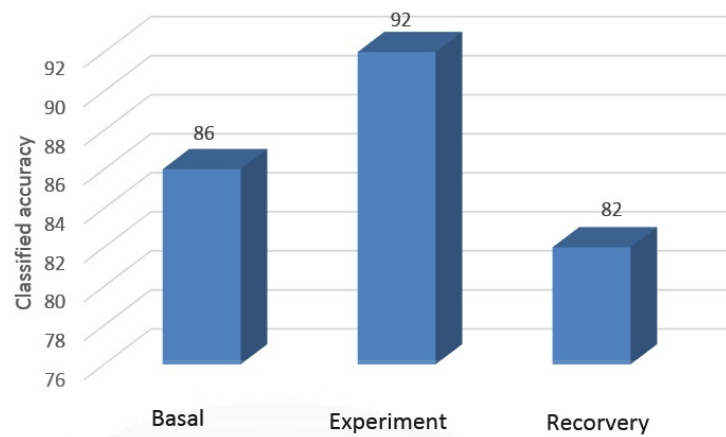


Figure 4.35 Comparison of best classifier performance of H₂O₂ by hybrid kernel function in each phase of CVR test.

The best parameters of H₂O₂ by hybrid kernel function determined are present in Table 4.27-4.29. Figure 4.32-4.34 show the separate results of the hybrid kernel function. When we vary the value of parameter (η and ϕ) to test the data sets in each phrase, the best values of both parameters resulting in the highest performance for low value is detected.

4.1.6 Comparison of best performance hybrid kernel function and single kernel function for H₂O₂ assessment

Table 4.30 Comparison of best performance hybrid kernel function and single kernel function in basal phase for H₂O₂ assessment

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Polynomial | $\beta=2, d=4$ | 72.00 | 7 |
| RBF | $\sigma=6$ | 74.00 | 6 |
| Sigmoid | $\gamma=2, \beta=2$ | 72.00 | 32 |
| Hybrid | $\eta=0.4, \phi=0.5$ | 86.00* | 6 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.12, 2.13, 2.14 and 3.10.

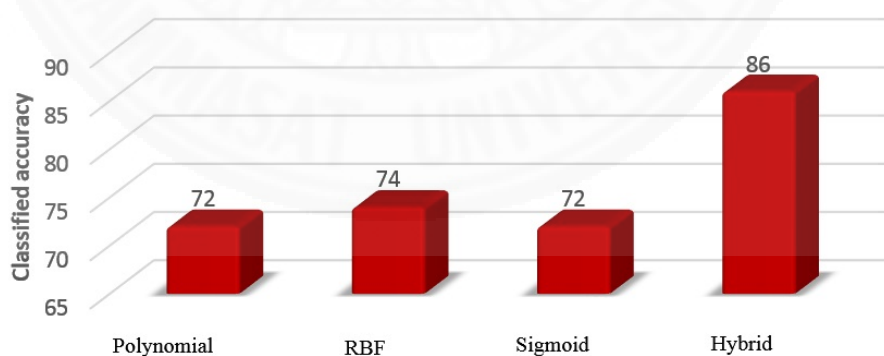


Figure 4.36 Comparison chart of best performance hybrid kernel function and single kernel function in basal phase for H₂O₂ assessment.

Table 4.31 Comparison of best performance hybrid kernel function and single kernel unction in experiment phase for H₂O₂ assessment

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Polynomial | $\beta=4, d=2$ | 80.00 | 5 |
| RBF | $\sigma=4$ | 90.00 | 10 |
| Sigmoid | $\gamma=1, \beta=3$ | 58.00 | 29 |
| Hybrid | $\eta=0.1, \phi=0.1$ | 92.00* | 11 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.12, 2.13, 2.14 and 3.10.

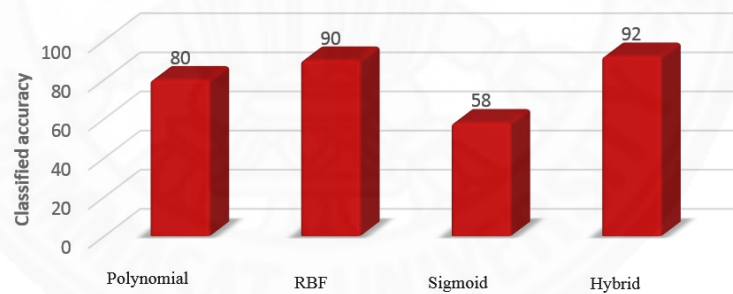


Figure 4.37 Comparison chart of best performance hybrid kernel function and single kernel function in experiment phase for H₂O₂ assessment.

Table 4.32 Comparison of best performance hybrid kernel function and single kernel function in recovery phase for H₂O₂ assessment

| Kernel Function | The experimental results | | |
|-----------------|--------------------------|-------------------------|---------------------|
| | Parameter [†] | Classified accuracy (%) | No. support vectors |
| Polynomial | $\beta=4, d=3$ | 50.00 | 5 |
| RBF | $\sigma=12$ | 76.00 | 31 |
| Sigmoid | $\gamma=2, \beta=5$ | 54.00 | 26 |
| Hybrid | $\eta=0.1, \phi=0.3$ | 82.00* | 8 |

* indicates to significant notice of high accuracy for each type of kernel function.

[†] Parameter can be considered by Eq. 2.12, 2.13, 2.14 and 3.10.

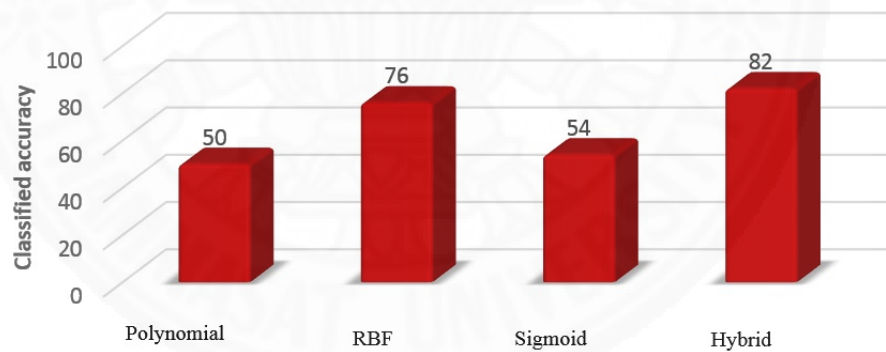


Figure 4.38 Comparison chart of best performance hybrid kernel function and single kernel function in recovery phase for H₂O₂ assessment.

From all findings result in Table 4.30-4.32 and Figure 4.36 - 4.38, they suggest that parameters η and ϕ are able to improve performance of H₂O₂ classification in lacunar stroke.

4.1.7 The results of the best classification modeling for classification of NO and H₂O₂

Table 4.33 Summarize the results of the best modeling parameter in basal phase

| Lacunar stroke biomarker | Kernel Function | Basal phase | |
|-------------------------------|-----------------|----------------------|--|
| | | The best parameter | The best classification modeling |
| NO | Polynomial | $\beta=1, d=2$ | $(x^T y + 1)^2$ |
| | RBF | $\sigma=6$ | $\exp\left(-\frac{\ x - y\ ^2}{2(6)^2}\right)$ |
| | Sigmoid | $\gamma=1, \beta=1$ | $\tanh((x^T \cdot y) + 1)$ |
| | Hybrid | $\eta=0.5, \phi=0.5$ | $0.5(x^T \cdot y + 1)^3 + 0.5\left(\frac{e^{2(x^T \cdot y+3)} + 1}{e^{2(x^T \cdot y+3)} - 1}\right)$ |
| H ₂ O ₂ | Polynomial | $\beta=4, d=2$ | $(x^T y + 4)^2$ |
| | RBF | $\sigma=3$ | $\exp\left(-\frac{\ x - y\ ^2}{2(3)^2}\right)$ |
| | Sigmoid | $\gamma=2, \beta=2$ | $k(x, y) = \tanh(2(x^T \cdot y) + 2)$ |
| | Hybrid | $\eta=0.4, \phi=0.5$ | $0.4(x^T \cdot y + 1)^3 + 0.5\left(\frac{e^{2(x^T \cdot y+3)} + 1}{e^{2(x^T \cdot y+3)} - 1}\right)$ |

Table 4.34 Summarize the results of the best modeling parameter in experiment phase

| Lacunar stroke biomarker | Kernel Function | Experiment phase | |
|-------------------------------|-----------------|----------------------|---|
| | | The best parameter | The best classification modeling |
| NO | Polynomial | $\beta=3, d=3$ | $(x^T y + 3)^3$ |
| | RBF | $\sigma=4$ | $\exp(-\frac{\ x-y\ ^2}{2(4)^2})$ |
| | Sigmoid | $\gamma=1, \beta=3$ | $k(x, y) = \tanh((x^T \cdot y) + 3)$ |
| | Hybrid | $\eta=0.4, \phi=0.5$ | $0.4(x^T \cdot y + 1)^3 + 0.5(\frac{e^{2(x^T \cdot y+3)} + 1}{e^{2(x^T \cdot y+3)} - 1})$ |
| H ₂ O ₂ | Polynomial | $\beta=4, d=2$ | $(x^T y + 4)^2$ |
| | RBF | $\sigma=4$ | $\exp(-\frac{\ x-y\ ^2}{2(4)^2})$ |
| | Sigmoid | $\gamma=1, \beta=3$ | $k(x, y) = \tanh((x^T \cdot y) + 3)$ |
| | Hybrid | $\eta=0.1, \phi=0.1$ | $0.1(x^T \cdot y + 1)^3 + 0.1(\frac{e^{2(x^T \cdot y+3)} + 1}{e^{2(x^T \cdot y+3)} - 1})$ |

Table 4.35 Summarize the results of the best modeling parameter in recovery phase

| Lacunar stroke biomarker | Kernel Function | Recovery phase | |
|-------------------------------|-----------------|----------------------|--|
| | | The best parameter | The best classification modeling |
| NO | Polynomial | $\beta=1, d=5$ | $(x^T y + 1)^5$ |
| | RBF | $\sigma=8$ | $\exp\left(-\frac{\ x - y\ ^2}{2(8)^2}\right)$ |
| | Sigmoid | $\gamma=1, \beta=2$ | $\tanh((x^T \cdot y) + 2)$ |
| | Hybrid | $\eta=0.5, \phi=0.5$ | $0.5(x^T \cdot y + 1)^3 + 0.5\left(\frac{e^{2(x^T \cdot y+3)} + 1}{e^{2(x^T \cdot y+3)} - 1}\right)$ |
| H ₂ O ₂ | Polynomial | $\beta=4, d=3$ | $(x^T y + 4)^3$ |
| | RBF | $\sigma=12$ | $\exp\left(-\frac{\ x - y\ ^2}{2(12)^2}\right)$ |
| | Sigmoid | $\gamma=2, \beta=5$ | $k(x, y) = \tanh(2(x^T \cdot y) + 5)$ |
| | Hybrid | $\eta=0.1, \phi=0.3$ | $0.1(x^T \cdot y + 1)^3 + 0.3\left(\frac{e^{2(x^T \cdot y+3)} + 1}{e^{2(x^T \cdot y+3)} - 1}\right)$ |

4.2 Discussion

Potential hybrid model of kernel function is proposed to be the most the highest classifier accuracy of NO in healthy control and LS during basal, experiment and recovery phases, 94 %, 96 % and 92 %, respectively. According to pathophysiology of small deep arteriole occlusion caused from endothelial dysfunction and vascular inflammation, NO is slightly reduced compared with healthy control in basal phase. During CVR, small changes of NO are evident in experiment and recovery phases in LS patient, but they differ from those in control. Greatest classifier model of hybrid kernel function reflects the mechanism of LS, even though, in clinical view, LS is considered as minor stroke since it shows minor neurological deficiency. In other

words, CVR is the best method testing cerebral reserve function and autoregulation in LS which it is confirmed by high classifier accuracy among basal, experiment and recovery phases.

For H_2O_2 classification, even though the measured H_2O_2 of both control and LS seems to be small difference of numbers, but a hybrid model shows high accuracy as 86 %, 92% and 82 % during basal, experiment and recovery, respectively. In LS, small deep neuron death is evident which in turn, activate glia cells release H_2O_2 into cerebrospinal fluid and then circulation. Contrast to LS, in large artery ischemic stroke or major stroke, H_2O_2 is higher than those in LS. Moreover, H_2O_2 plays paradox fashion of vasodilation and vasoconstriction depends on its concentration and condition.

Taken together, high classified accuracy of H_2O_2 during basal, experiment and recovery is the most the high performance of a limited extent of H_2O_2 .

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The performance of SVMs depends on the different kind of kernel functions and adjustment of its parameters because each single kernel function represents own limitation of each different features. Concerning physiological processes of NO and H₂O₂ synthesis, release to the circulation, reaching an equilibrium state and function within their half-life, the best model explaining dynamic process should be from combination function model. Look at the model, if we are able to integrate two or more single kernel functions to construct hybrid kernel function and give the superiority among the single ones, it should be fitted with NO and H₂O₂ availability. On the other hand, the performance of the hybrid kernel function would be worse than the single ones because of an excessive parameters that we combine them. Therefore, the most important step is searching for the optimum form of each parameter of single kernel functions in order to construct the best performance for hybrid kernel functions.

The results determined by the classifier performance of NO assessed by polynomial, RBF and hybrid kernels have shown that the classifier performance is high in experiment phase. It was found that hybrid kernel generating the highest classification accuracy. For H₂O₂, the classifier performances by polynomial, RBF and hybrid kernel were high when testing them in experiment phase and hybrid kernel giving the highest classification accuracy as well as the case of NO. However, sigmoid kernel function has shown low performance of the most kernel function but it showed the highest classified accuracy in recovery and basal phases for NO and H₂O₂ assessment, respectively.

Hybrid kernel function is the best for highly accurate classification with low value of parameter. It is suitable for discrimination of low NO and H₂O₂ levels in healthy and lacunar stroke that reflects to endothelial dysfunction and vascular inflammation. These results will be applied and further studied in large artery ischemic

stroke and also in high risk of acute stroke such as atrial fibrillation, chronic hypertension, atherosclerosis etc.



REFERENCES

1. Fisher CM. Lacunar strokes and infarcts: A review. *Neurology*. 1982;32(8):87-6.
2. Staaf G, Lindgren A, Norrving B. Pure Motor Stroke From Presumed Lacunar Infarct: Long-Term Prognosis for Survival and Risk of Recurrent Stroke. *Stroke*. 2001;32(11):2592-6.
3. Deanfield JE, Halcox JP, Rabelink TJ. Endothelial function and dysfunction: Testing and clinical relevance. *Circulation*. 2007;115(10):1285-95.
4. Yoon Y, Song J, Hong SH, Kim JQ. Plasma nitric oxide concentrations and nitric oxide synthase gene polymorphisms in coronary artery disease. *Clinical Chemistry*. 2000;46(10):1626-30.
5. Pacher P, Beckman JS, Liaudet L. Nitric oxide and peroxynitrite in health and disease. *Physiological Reviews*. 2007;87(1):315-424.
6. Yu W, Liu T, Valdez R, Gwinn M, Khoury MJ. Application of support vector machine modeling for prediction of common diseases: the case of diabetes and pre-diabetes. *BMC Medical Informatics and Decision Making*. 2010;10(1):1-7.
7. Wardlaw JM, Smith EE, Biessels GJ, Cordonnier C, Fazekas F, Frayne R, et al. Neuroimaging standards for research into small vessel disease and its contribution to ageing and neurodegeneration. *Lancet Neurology*. 2013;12(8):822-38.
8. Norrving B. Long-term prognosis after lacunar infarction. *Lancet Neurology*. 2003;2(4):238-45.
9. Craggs LJJ, Yamamoto Y, Deramecourt V, Kalra RN. Microvascular Pathology and Morphometrics of Sporadic and Hereditary Small Vessel Diseases of the Brain. *Brain Pathology (Zurich, Switzerland)*. 2014;24(5):495-509.
10. Lammie GA. Pathology of small vessel stroke. *British Medical Bulletin*. 2000;56(2):296-306.
11. Petty GW, Brown Jr RD, Whisnant JP, Sicks JD, O'Fallon WM, Wiebers DO. Ischemic stroke subtypes: A population-based study of functional outcome, survival, and recurrence. *Stroke*. 2000;31(5):1062-8.

12. Hankey GJ, Jamrozik K, Broadhurst RJ, Forbes S, Burvill PW, Anderson CS, et al. Five-year survival after first-ever stroke and related prognostic factors in the Perth Community Stroke Study. *Stroke*. 2000;31(9):2080-6.
13. Sacco SE, Whisnant JP, Broderick JP, Phillips SJ, Michael WO. Epidemiological characteristics of lacunar infarcts in a population. *Stroke*. 1991;22(10):1236-41.
14. Napoli C, de Nigris F, Williams-Ignarro S, Pignalosa O, Sica V, Ignarro LJ. Nitric oxide and atherosclerosis: An update. *Nitric Oxide - Biology and Chemistry*. 2006;15(4):265-79.
15. Torres MA, Jones JDG, Dangl JL. Reactive Oxygen Species Signaling in Response to Pathogens. *Plant Physiology*. 2006;141(2):373-8.
16. Subbotin VM. Excessive intimal hyperplasia in human coronary arteries before intimal lipid depositions is the initiation of coronary atherosclerosis and constitutes a therapeutic target. *Drug Discovery Today*.
17. Bendall JK, Douglas G, McNeill E, Channon KM, Crabtree MJ. Tetrahydrobiopterin in Cardiovascular Health and Disease. *Antioxidants & Redox Signaling*. 2014;20(18):3040-77.
18. Leiper J, Nandi M. The therapeutic potential of targeting endogenous inhibitors of nitric oxide synthesis. *Nat Rev Drug Discov*. 2011;10(4):277-91.
19. Hadi HAR, Carr CS, Al Suwaidi J. Endothelial Dysfunction: Cardiovascular Risk Factors, Therapy, and Outcome. *Vascular Health and Risk Management*. 2005;1(3):183-98.
20. Daiber A, Steven S, Weber A, Shuvaev VV, Muzykantov VR, Laher I, et al. Targeting vascular (endothelial) dysfunction. *British Journal of Pharmacology*. 2016:n/a-n/a.
21. Russo G, Leopold JA, Loscalzo J. Vasoactive substances: Nitric oxide and endothelial dysfunction in atherosclerosis. *Vascular Pharmacology*. 2002;38(5):259-69.
22. Beckman JS, Koppenol WH. Nitric oxide, superoxide, and peroxynitrite: The good, the bad, and the ugly. *American Journal of Physiology - Cell Physiology*. 1996;271(5 40-5).

23. Weyrich AS, Ma XL, Buerke M, Murohara T, Armstead VE, Lefer AM, et al. Physiological concentrations of nitric oxide do not elicit an acute negative inotropic effect in unstimulated cardiac muscle. *Circulation Research*. 1994;75(4):692-700.
24. Wever RMF, Lüscher TF, Cosentino F, Rabelink TJ. Atherosclerosis and the Two Faces of Endothelial Nitric Oxide Synthase. *Circulation*. 1998;97(1):108-12.
25. Nakazono K, Watanabe N, Matsuno K, Sasaki J, Sato T, Inoue M. Does superoxide underlie the pathogenesis of hypertension? *Proceedings of the National Academy of Sciences of the United States of America*. 1991;88(22):10045-8.
26. Cannon RO. Role of nitric oxide in cardiovascular disease: focus on the endothelium. *Clinical Chemistry*. 1998;44(8):1809-19.
27. Cai H, Harrison DG. Endothelial Dysfunction in Cardiovascular Diseases: The Role of Oxidant Stress. *Circulation Research*. 2000;87(10):840-4.
28. Cardillo C, De Felice F, Campia U, Folli G. Psychophysiological reactivity and cardiac end-organ changes in white coat hypertension. *Hypertension*. 1993;21(6):836-44.
29. Gradinaru D, Borsa C, Ionescu C, Prada GI. Oxidized LDL and NO synthesis—Biomarkers of endothelial dysfunction and ageing. *Mechanisms of Ageing and Development*. 2015;151:101-13.
30. Adams MR, Kinlay S, Blake GJ, Orford JL, Ganz P, Selwyn AP. Atherogenic Lipids and Endothelial Dysfunction: Mechanisms in the Genesis of Ischemic Syndromes. *Annual Review of Medicine*. 2000;51(1):149-67.
31. Halliwell B, Gutteridge JMC, Andorn AC, Britton RS, Bacon BR. Lipid peroxidation in brain homogenates: The role of iron and hydroxyl radicals (multiple letters). *Journal of Neurochemistry*. 1997;69(3):1330-1.
32. Liao JK, Shin WS, Lee WY, Clark SL. Oxidized Low-density Lipoprotein Decreases the Expression of Endothelial Nitric Oxide Synthase. *Journal of Biological Chemistry*. 1995;270(1):319-24.
33. Szczeklik A, Sladek K, Dworski R, Nizankowska E, Soja J, Sheller J, et al. Bronchial aspirin challenge causes specific eicosanoid response in aspirin-sensitive asthmatics. *American Journal of Respiratory and Critical Care Medicine*. 1996;154(6):1608-14.

34. Steinbrugger I, Haas A, Maier R, Renner W, Mayer M, Werner C, et al. Analysis of inflammation- and atherosclerosis-related gene polymorphisms in branch retinal vein occlusion. *Molecular Vision*. 2009;15:609-18.
35. De Caterina R, Libby P, Peng HB, Thannickal VJ, Rajavashisth TB, Gimbrone Jr MA, et al. Nitric oxide decreases cytokine-induced endothelial activation: Nitric oxide selectively reduces endothelial expression of adhesion molecules and proinflammatory cytokines. *Journal of Clinical Investigation*. 1995;96(1):60-8.
36. Newby AC, George SJ. Proliferation, migration, matrix turnover, and death of smooth muscle cells in native coronary and vein graft atherosclerosis. *Current Opinion in Cardiology*. 1996;11(6):574-82.
37. Lloyd-Jones DM, Bloch KD. The vascular biology of nitric oxide and its role in atherogenesis. *Annual Review of Medicine* 1996. p. 365-75.
38. Griendling KK, Ushio-Fukai M. Redox control of vascular smooth muscle proliferation. *Journal of Laboratory and Clinical Medicine*. 1998;132(1):9-15.
39. Berman RS, Martin W. Arterial endothelial barrier dysfunction: actions of homocysteine and the hypoxanthine-xanthine oxidase free radical generating system. *British Journal of Pharmacology*. 1993;108(4):920-6.
40. Siflinger-Birnboim A, Goligorsky MS, Del Vecchio PJ, Malik AB. Activation of protein kinase C pathway contributes to hydrogen peroxide- induced increase in endothelial permeability. *Laboratory Investigation*. 1992;67(1):24-30.
41. McQuaid KE, Smyth EM, Keenan AK. Evidence for modulation of hydrogen peroxide-induced endothelial barrier dysfunction by nitric oxide in vitro. *European Journal of Pharmacology*. 1996;307(2):233-41.
42. Vapnik VN. *The Nature of Statistical Learning Theory*: Springer New York; 2013.
43. Spratt H, Ju H, Brasier AR. A structured approach to predictive modeling of a two-class problem using multidimensional data sets. *Methods (San Diego, Calif)*. 2013;61(1):73-85.
44. Mountrakis G, Im J, Ogole C. Support vector machines in remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*. 2011;66(3):247-59.
45. Shawe-Taylor J, Cristianini N. *Kernel Methods for Pattern Analysis*: Cambridge University Press; 2004.

46. Szeliski R. *Computer Vision: Algorithms and Applications*: Springer London; 2010.
47. Solla SA, Leen TK, Müller KR. *Advances in Neural Information Processing Systems 12: Proceedings of the 1999 Conference*: MIT Press; 2000.
48. Ikeda K, Aoishi T. An asymptotic statistical analysis of support vector machines with soft margins. *Neural Networks*. 2005;18(3):251-9.
49. Guo D, Fridriksson J, Fillmore P, Rorden C, Yu H, Zheng K, et al. Automated lesion detection on MRI scans using combined unsupervised and supervised methods. *BMC Medical Imaging*. 2015;15(1):1-21.
50. Huang F, Yan L. Combined Kernel-Based BDT-SMO Classification of Hyperspectral Fused Images. *The Scientific World Journal*. 2014;2014:738250.
51. Al-Muhaideb S, El Bachir Menai M. Hybrid metaheuristics for medical data classification. *Studies in Computational Intelligence* 2013. p. 187-217.
52. Schölkopf B, Smola AJ. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*: MIT Press; 2002.
53. Vapnik VN. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*. 1999;10(5):988-99.
54. Boser BE, Guyon IM, Vapnik VN, editors. Training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*; 1992.
55. Gama J, Pedersen RU. Predictive Learning in Sensor Networks. In: Gama J, Gaber MM, editors. *Learning from Data Streams: Processing Techniques in Sensor Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg; 2007. p. 143-64.
56. Ando S, Suzuki E. Minimizing response time in time series classification. *Knowledge and Information Systems*. 2016;46(2):449-76.
57. Smola AJ, Schölkopf B. A tutorial on support vector regression. *Statistics and Computing*. 2004;14(3):199-222.
58. Herrera M, Guglielmetti A, Xiao M, Filomeno Coelho R. Metamodel-assisted optimization based on multiple kernel regression for mixed variables. *Structural and Multidisciplinary Optimization*. 2014;49(6):979-91.
59. Schölkopf B, Burges CJC, Smola AJ. *Advances in Kernel Methods: Support Vector Learning*: Philomel Books; 1999.

60. Abe S. Support Vector Machines for Pattern Classification: Springer London; 2010.
61. Girard JM, Cohn JF, Jeni LA, Lucey S, De la Torre F. How much training data for facial action unit detection? IEEE International Conference on Automatic Face & Gesture Recognition and Workshops. 2015;1:10.1109/FG.2015.7163106.
62. James G, Witten D, Hastie T, Tibshirani R. An Introduction to Statistical Learning: with Applications in R: Springer New York; 2014.
63. Mika S, Ratsch G, Weston J, Scholkopf B, Muller K-R, editors. Fisher discriminant analysis with kernels. Neural Networks for Signal Processing - Proceedings of the IEEE Workshop; 1999.
64. Ding S, Chen L. Intelligent Optimization Methods for High-Dimensional Data Classification for Support Vector Machines. Intelligent Information Management. 2010;Vol.02No.06:11.
65. Olfati E, Zarabadipour H, Shoorehdeli MA, editors. Feature subset selection and parameters optimization for support vector machine in breast cancer diagnosis. 2014 Iranian Conference on Intelligent Systems, ICIS 2014; 2014.
66. Huang M-L, Hung Y-H, Lee WM, Li RK, Jiang B-R. SVM-RFE Based Feature Selection and Taguchi Parameters Optimization for Multiclass SVM Classifier. The Scientific World Journal. 2014;2014:795624.
67. Li C, An X, Li R. A chaos embedded GSA-SVM hybrid system for classification. Neural Computing and Applications. 2015;26(3):713-21.
68. Wang L. Feature subset selection for multi-class SVM based image classification. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)2007. p. 145-54.
69. Zeng M, Yang Y, Zheng J, Cheng J. Maximum margin classification based on flexible convex hulls. Neurocomputing. 2015;149(PB):957-65.
70. Y Joshi A, N Patel V, Parikh KS, Shah TP. 3rd International Conference on Innovations in Automation and Mechatronics Engineering 2016, ICIAME 2016 05-06 February, 2016Support Vector Machine – A Large Margin Classifier to Diagnose Skin Illnesses. Procedia Technology. 2016;23:369-75.
71. Jayadeva. Learning a hyperplane classifier by minimizing an exact bound on the VC dimension1. Neurocomputing. 2015;149, Part B:683-9.

72. Lewis DP, Jebara T, Noble WS. Support vector machine learning from heterogeneous data: an empirical analysis using protein sequence and structure. *Bioinformatics*. 2006;22(22):2753-60.
73. Bashashati H, Ward RK, Birch GE, Bashashati A. Comparing Different Classifiers in Sensory Motor Brain Computer Interfaces. *PLoS ONE*. 2015;10(6):e0129435.
74. Kuncheva LI. *Combining Pattern Classifiers: Methods and Algorithms*: Wiley; 2004.
75. Sherrod PH. DTREG predictive modeling software. Software available at [http://www dtreg com](http://www.dtreg.com). 2003.
76. Arnosti NA, Kalita JK. Cutting plane training for linear support vector machines. *IEEE Transactions on Knowledge and Data Engineering*. 2013;25(5):1186-90.
77. Theodoridis S, Pikrakis A, Koutroumbas K, Cavouras D. *Introduction to Pattern Recognition: A Matlab Approach*: Elsevier Science; 2010.
78. Deng N, Tian Y, Zhang C. *Support Vector Machines: Optimization Based Theory, Algorithms, and Extensions*: CRC Press; 2012.
79. Ma Y, Guo G. *Support Vector Machines Applications*: Springer International Publishing; 2014.
80. Chen N, Lu W, Yang J, Li G. *Support Vector Machine in Chemistry* 2004.
81. Liu H, Motoda H. *Feature Selection for Knowledge Discovery and Data Mining*: Springer US; 2012.
82. Chen R-C, Hsieh C-H. Web page classification based on a support vector machine using a weighted vote schema. *Expert Systems with Applications*. 2006;31(2):427-35.
83. Gold C, Holub A, Sollich P. Bayesian approach to feature selection and parameter tuning for support vector machine classifiers. *Neural Networks*. 2005;18(5-6):693-701.
84. Kohavi R, John GH. Wrappers for feature subset selection. *Artificial Intelligence*. 1997;97(1-2):273-324.

85. Shon T, Kim Y, Lee C, Moon J, editors. A machine learning framework for network anomaly detection using SVM and GA. Proceedings from the 6th Annual IEEE System, Man and Cybernetics Information Assurance Workshop, SMC 2005; 2005.
86. Zhang L, B. Jack L, Nandi AK. Fault detection using genetic programming. *Mechanical Systems and Signal Processing*. 2005;19(2):271-89.
87. Samanta B, Al-Balushi KR, Al-Araimi SA. Artificial neural networks and support vector machines with genetic algorithm for bearing fault detection. *Engineering Applications of Artificial Intelligence*. 2003;16(7-8):657-65.
88. Huang C-L, Chen M-C, Wang C-J. Credit scoring with a data mining approach based on support vector machines. *Expert Systems with Applications*. 2007;33(4):847-56.
89. Huang C-L, Wang C-J. A GA-based feature selection and parameters optimization for support vector machines. *Expert Systems with applications*. 2006;31(2):231-40.
90. Aggarwal CC. *Data Classification: Algorithms and Applications*: CRC Press; 2015.
91. Pai PF, Hong WC. Support vector machines with simulated annealing algorithms in electricity load forecasting. *Energy Conversion and Management*. 2005;46(17):2669-88.
92. Foody GM, Mathur A. A relative evaluation of multiclass image classification by support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*. 2004;42(6):1335-43.
93. Melgani F, Bruzzone L. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*. 2004;42(8):1778-90.
94. Pimtongngam Y, Intharakham K, Suwanprasert K, editors. Classification of nitric oxide assessed by hybrid kernel function in lacunar stroke. *BMEiCON 2013 - 6th Biomedical Engineering International Conference*; 2013.
95. Freestone B, Krishnamoorthy S, Lip GYH. Assessment of endothelial dysfunction. *Expert Review of Cardiovascular Therapy*. 2010;8(4):557-71.
96. Suarez-Alvarez MM, Pham DT, Prostov MY, Prostov YI. Statistical approach to normalization of feature vectors and clustering of mixed datasets. *Proceedings of the*

Royal Society A: Mathematical, Physical and Engineering Sciences. 2012;468(2145):2630-51.

97. Pathak J, Bailey KR, Beebe CE, Bethard S, Carrell DS, Chen PJ, et al. Normalization and standardization of electronic health records for high-throughput phenotyping: The sharpn consortium. *Journal of the American Medical Informatics Association*. 2013;20(E2).

98. Hansen MAE. Data Analysis. *Metabolome Analysis: An Introduction* 2006. p. 146-87.

99. Ji Ab, Pang Jh, Qiu Hj. Support vector machine for classification based on fuzzy training data. *Expert Systems with Applications*. 2010;37(4):3495-8.

100. Phimthong-Ngam Y, Intharakham K, Suwanprasert K. Evaluation of Nitric Oxide in Lacunar Stroke and Young Healthy during Cerebrovascular Reactivity by Support Vector Machine. *International Journal of Computer Applications*. 2016; 146(9):28-35.



APPENDICES

APPENDIX A

Source code of experiments for classification modeling
(For the Scilab programming language)

Polynomial kernel function

```

// Display mode
mode(0);

// Display warning for floating point exception
ieee(1);

///// POLYNOMIAL KERNEL /////

Zdel(wjnsjd())
C_xlear
C_xlC_x

mtlb_format("C_xompaC_xt")
global("fjgt4")

l = 2; // Dimensionality
N = 50; // Number of vectors
m = 1; // Dimension
a = 33; // add control
b = 43; // add patient
maZ_C_x = 82.2; // max data of control
mjn_C_x = 23.3; // min data of control
aver_C_x = 58.06; // average of control
maZ_p = 67.4; // max data of patient
mjn_p = 21.6; // min data of patient
aver_p = 39; // average of patient
aver_C_xp = (aver_c+aver_p)/2; // (aver_c+aver_p)/2

// Generate the training set Z1
rand("seed",0)
Z_1 = mtlb_a((maZ_C_x-mjn_p)*rand(1,N),maZ_p/mjn_p);

if mtlb_logjC_x(Z_1,">",aver_C_xp) then
    Z1 = mtlb_s(Z_1,aver_C_xp);
else
    Z1 = mtlb_s(aver_C_xp,Z_1);
end;
for j = 1:N
    t = (1/20)*(Z1(1,j)^3+Z1(1,j)^2+Z1(1,j)+1);

    if t>Z1(2,j) then
        v1(1,j) = 1;
    else
        v1(1,j) = -1;
    end;
end;

// Generate the test set Z2

rand("seed",0)

```

```

C_xt = mtlb_a((maZ_C_x-mjn_C_x) .*rand(m,a),mjn_C_x);
pt = mtlb_a((maZ_p-mjn_p) .*rand(m,b),mjn_p);

C_xt2 = [23.3,24.2,25.5,35.3,44,61.5,62.1,63.9,64.2,66.6,69.7,69.8,71.4,72.5,72.6,78.2,82.2];
pt2 = [21.6, 33, 33.4, 36.3, 39, 42.3, 67.4];

A = [pt,pt2];
B = [C_xt,C_xt2];
C_X = [A;B];

if C_X>aver_C_xp then
    Z2 = C_X-aver_C_xp;
else
    Z2 = aver_C_xp-C_X;
end;
for j = 1:N
    t = (1/20)*(Z2(1,j)^3+Z2(1,j)^2+Z2(1,j)+1);

    if t>Z2(2,j) then
        v2(1,j) = 1;
    else
        v2(1,j) = -1;
    end;
end;

// Plot the training set Z1
// Matlab function figure not yet converted, original calling sequence used.
//(Warning name conflict: function name changed from figure to %figure).
/figure(1,plot(Z1(1,mtlb_logjC_x(v1,"==",1)),Z1(2,mtlb_logjC_x(v1,"==",1)),"rZ",Z1(1,mtlb_logjC_x
(v1,"==",-1)),Z1(2,mtlb_logjC_x(v1,"==",-1)),"bo"))
//Matlab function figure not yet converted, original calling sequence used.
//(Warning name conflict: function name changed from figure to %figure).
/figure(1,set(gca,"isovjew","on"))

// ////////////////////////////////// POLVNOMJAL 1 //////////////////////////////////
fjgt4 = 2;
//No simple equivalent, so mtlb_fprintf() is called..
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 1:// beta
k_par2 = 2:// n
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.92: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,set(gca,"isovjew","on"))

```

```

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
    !!! L.110: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup).*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn(1,j) = 1;
    else
        out_trajn(1,j) = -1;
    end;
end;

// Classification of the test set
for j = 1:N
    !!! L.120: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup).*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test(1,j) = 1;
    else
        out_test(1,j) = -1;
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn.*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test.*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

// //////////////////////////////////////// POLVNOMJAL 2 ////////////////////////////////////////
fjgt4 = 3;
// L.141: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 1; // beta
k_par2 = 3; // n
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.151: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.159: Unknown function svCplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

```

```

// !! L.161: Matlab function figure not yet converted, original calling sequence used.
// L.161: (Warning name conflict: function name changed from figure to %offigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    // !! L.169: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    // !! L.179: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// POLVNOMJAL 3 //////////////////////////////////
fjgt4 = 4;
// L.198: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 1; // beta
k_par2 = 4; // n
C_X = 2;
to_1 = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.208: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;

```

```

!!! L.216: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

!!! L.218: Matlab function figure not yet converted, original calling sequence used.
//L.218: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    !!! L.226: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    !!! L.236: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// POLVNOMJAL 4 //////////////////////////////////
fjgt4 = 5;
//L.255: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 1;// beta
k_par2 = 5;// n
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.265: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;

```

```

input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.273: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

!!! L.275: Matlab function figure not yet converted, original calling sequence used.
// L.275: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    !!! L.283: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup).*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    !!! L.293: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup).*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn.*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test.*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

////////// POLVNOMJAL 5 ////////////
fjgt4 = 6;
// L.312: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 2;// beta
k_par2 = 2;// n
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.322: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

```

```

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.330: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

!!! L.332: Matlab function figure not yet converted, original calling sequence used.
//L.332: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));

// Classification of the training set
for j = 1:N
!!! L.340: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
!!! L.350: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// POLVNOMJAL 6 //////////////////////////////////
fjgt4 = 7;
//L.369: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 3;// beta
k_par2 = 2;// n
C_X = 2;
to_1 = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;

```

```

// !! L.379: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,'ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
// !! L.387: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

// !! L.389: Matlab function figure not yet converted, original calling sequence used.
// L.389: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));

// Classification of the training set
for j = 1:N
    // !! L.397: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
    .*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    // !! L.407: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
    .*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// POLVNOMJAL 7 //////////////////////////////////
fjgt4 = 8;
// L.426: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\\n\\n");

ker_nel = "polv"
k_par1 = 4;// beta
k_par2 = 2;// n
C_X = 2;
to_1 = 0.001;

```



```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
//!! L.436: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0)));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0)));

// Classification of the training set
for j = 1:N
//!! L.454: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
//!! L.464: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

////////// POLVNOMJAL 8 //////////
fjgt4 = 9;
//L.483: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprjntf("\n\n");

ker_nel = "polv"
k_par1 = 5;// beta

```

```

k_par2 = 2;// n
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.493: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
!!! L.501: Unknown function svCplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,!!! L.503: Matlab function figure not yet converted, original calling sequence used.
// L.503: (Warning name conflict: function name changed from figure to %figure).
/fjgure(fjgt4),set(gca),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    !!! L.511: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
    .*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    !!! L.521: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
    .*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

////////// POLVNOMJAL 9 //////////
fjgt4 = 10;
// L.540: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

```

```

ker_nel = "poly"
k_par1 = 1; // beta
k_par2 = 3; // n
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.550: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
// !! L.558: Unknown function svCplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

// !! L.560: Matlab function figure not yet converted, original calling sequence used.
// L.560: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    // !! L.568: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
    .* mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    // !! L.578: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
    .* mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

// Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

```

```

////////// POLYNOMJAL 10 //////////
fjgt4 = 11;
// L.597: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 2; // beta
k_par2 = 3; // n
C_X = 2;
to_1 = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.607: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
// !! L.615: Unknown function svplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

// !! L.617: Matlab function figure not yet converted, original calling sequence used.
// L.617: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
// !! L.625: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
// !! L.635: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

```

```

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha), ">", 0)))
// ////////////////////////////////// POLVNOMJAL 11 //////////////////////////////////
fjgt4 = 12;
// L.653: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 3; // beta
k_par2 = 3; // n
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.663: Unknown function SM_O2 not converted, original calling sequence used.
[alpha, b, w, evals, stp, glob] = SM_O2(Z1', v1', ker_nel, k_par1, k_par2, C_X, to_l, step_s, ep_s, me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1, sjez(Z1', 2));
bjas = -b;
aspeC_xt = 0;
// !! L.671: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1', v1', ker_nel, k_par1, k_par2, alpha, bjas, aspeC_xt, ma_g, ZaZis, vaZis, input);

// !! L.673: Matlab function figure not yet converted, original calling sequence used.
// L.673: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4), set(gca(), "isovjew", "on")

Z_sup = Z1(:, mtlb_logjC_x(mtlb_double(mtlb_t(alpha)), "~=", 0));
alpha_sup = mtlb_t(mtlb_e(alpha, mtlb_logjC_x(mtlb_double(alpha), "~=", 0)));
v_sup = mtlb_e(v1, mtlb_logjC_x(mtlb_double(alpha), "~=", 0));

// Classification of the training set
for j = 1:N
// !! L.681: Unknown function Kernel_Calc not converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.* mtlb_double(mtlb_t(CalcKernel(Z_sup', (Z1(:, j))', ker_nel, k_par1, k_par2))))), mtlb_double(b));
if mtlb_logjC_x(t, ">", 0) then
out_trajn = mtlb_j(out_trajn, j, 1);
else
out_trajn = mtlb_j(out_trajn, j, -1);
end;
end;

// Classification of the test set
for j = 1:N
// !! L.691: Unknown function Kernel_Calc not converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.* mtlb_double(mtlb_t(CalcKernel(Z_sup', (Z2(:, j))', ker_nel, k_par1, k_par2))))), mtlb_double(b));
if mtlb_logjC_x(t, ">", 0) then
out_test = mtlb_j(out_test, j, 1);
else
out_test = mtlb_j(out_test, j, -1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1, "<", 0)))/maZ(sjez(v1))

// Classification the test error

```

```

Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))
//////////////// POLVNOMJAL 12 //////////////////////

fjgt4 = 13;
// L.709: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 4;// beta
k_par2 = 3;// n
C_X = 2;
to_1 = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
//!! L.719: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
//!! L.727: Unknown function svC_plot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

//!! L.729: Matlab function figure not yet converted, original calling sequence used.
// L.729: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));

// Classification of the training set
for j = 1:N
//!! L.737: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
//!! L.747: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

```

```

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

// ////////////////////////////////// POLVNOMJAL 13 //////////////////////////////////
fjgt4 = 14;
// L.765: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 5;// beta
k_par2 = 3;// n
C_X = 2;
to_1 = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.775: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
// !! L.783: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,set(gca,"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));

// Classification of the training set
for j = 1:N
// !! L.793: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
// !! L.803: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

```

```

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

// //////////////////////////////////////// POLVNOMJAL 14 ////////////////////////////////////////
fjgt4 = 15;
// L.821: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\\n\\n");

ker_nel = "polv"
k_par1 = 1;// beta
k_par2 = 4;// n
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.831: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
// !! L.839: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

// !! L.841: Matlab function figure not yet converted, original calling sequence used.
// L.841: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));

// Classification of the training set
for j = 1:N
// !! L.849: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
// !! L.859: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else

```



```

    out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))
// //////////////////////////////////////// POLVNOMJAL 15 ////////////////////////////////////////
fjgt4 = 16;
//L.877: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\\n\\n");

ker_nel = "polv"
k_par1 = 2;// beta
k_par2 = 4;// n
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
//!! L.887: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
//!! L.905: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
    out_trajn = mtlb_j(out_trajn,j,1);
else
    out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
//!! L.915: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then

```

```

    out_test = mtlb_j(out_test,j,1);
else
    out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))
// ////////////////////////////////// POLVNOMJAL 16 //////////////////////////////////
fjgt4 = 17;
// L.933: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 3;// beta
k_par2 = 4;// n
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.943: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
// !! L.951: Unknown function svC_xplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

// !! L.953: Matlab function figure not yet converted, original calling sequence used.
// L.953: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
    // !! L.961: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
    .*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    // !! L.971: Unknown function Kernel_Calcnot converted, original calling sequence used.

```

```

t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
    out_test = mtlb_j(out_test,j,1);
else
    out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

// ////////////////////////////////// POLYNOMJAL 17 //////////////////////////////////
fjgt4 = 18;
// L.990: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 4; // beta
k_par2 = 4; // n
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.1000: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
// !! L.1008: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

// !! L.1010: Matlab function figure not yet converted, original calling sequence used.
// L.1010: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));

// Classification of the training set
for j = 1:N
    // !! L.1018: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;
end;

```

```

// Classification of the test set
for j = 1:N
    !!! L.1028: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
    .*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

// ////////////////////////////////// POLVNOMJAL 18 //////////////////////////////////
fjgt4 = 19;
!!! L.1047: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 4; beta
k_par2 = 5; n
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.1057: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.1065: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

!!! L.1067: Matlab function figure not yet converted, original calling sequence used.
!!! L.1067: (Warning name conflict: Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
    !!! L.1075: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
    .*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

```

```

end;
end;

// Classification of the test set
for j = 1:N
    !! L.1085: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

//////////////////////////////// POLVNOMJAL 19 //////////////////////////////////
fjgt4 = 20;
// L.1104: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "polv"
k_par1 = 5;// beta
k_par2 = 5;// n
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!! L.1114: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!! L.1122: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

!! L.1124: Matlab function figure not yet converted, original calling sequence used.
// L.1124: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
    !! L.1132: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then

```

```

    out_trajn = mtlb_j(out_trajn,j,1);
else
    out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
    !! L.1142: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b)));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

```

RBF kernel function

```

// Display mode
mode(0);

// Display warning for floating point exception
jeee(1);

///// RBF KER_NEL /////

Zdel(wjnsjd())
C_xlear
C_xlC_x

mtlb_format("C_xompaC_xt")
global("fjgt4")

l = 2; // Dimensionality
N = 50; // Number of vectors
m = 1; // Dimension
a = 33; // add control
b = 43; // add patient
maZ_C_x = 82.2; // max data of control
mjn_C_x = 23.3; // min data of control
aver_C_x = 58.06; // average of control
maZ_p = 67.4; // max data of patient
mjn_p = 21.6; // min data of patient
aver_p = 39; // average of patient
aver_C_xp = (aver_c+aver_p)/2; //(aver_c+aver_p)/2

// Generate the training set Z1
rand("seed",0)
Z_1 = mtlb_a((maZ_C_x-mjn_p)*rand(1,N),maZ_p/mjn_p);

```

```

if mtlb_logjC_x(Z_1,">",aver_C_xp) then
    Z1 = mtlb_s(Z_1,aver_C_xp);
else
    Z1 = mtlb_s(aver_C_xp,Z_1);
end;
for j = 1:N
    t = (1/20)*(Z1(1,j)^3+Z1(1,j)^2+Z1(1,j)+1);

    if t>Z1(2,j) then
        v1(1,j) = 1;
    else
        v1(1,j) = -1;
    end;
end;

// Generate the test set Z2

rand("seed",0)

C_xt = mtlb_a((maZ_C_x-mjn_C_x) .*rand(m,a),mjn_C_x);
pt = mtlb_a((maZ_p-mjn_p) .*rand(m,b),mjn_p);

C_xt2 = [23.3,24.2,25.5,35.3,44,61.5,62.1,63.9,64.2,66.6,69.7,69.8,71.4,72.5,72.6,78.2,82.2];
pt2 = [21.6,33,33.4,36.3,39,42.3,67.4];

A = [pt,pt2];
B = [C_xt,C_xt2];
C_X = [A;B];

if C_X>aver_C_xp then
    Z2 = C_X-aver_C_xp;
else
    Z2 = aver_C_xp-C_X;
end;
for j = 1:N
    t = (1/20)*(Z2(1,j)^3+Z2(1,j)^2+Z2(1,j)+1);

    if t>Z2(2,j) then
        v2(1,j) = 1;
    else
        v2(1,j) = -1;
    end;
end;

// Plot the training set Z1
//!! L.76: Matlab function figure not yet converted, original calling sequence used.
//L.76: (Warning name conflict: function name changed from figure to %figure).
/figure(1),plot(Z1(1,mtlb_logjC_x(v1,"==",1)),Z1(2,mtlb_logjC_x(v1,"==",1)),"rZ",Z1(1,mtlb_logjC_x
(v1,"==",-1)),Z1(2,mtlb_logjC_x(v1,"==",-1)),"bo")
// Matlab function figure not yet converted, original calling sequence used.
//(Warning name conflict: function name changed from figure to %figure).
/figure(1),set(gca(),"isovjew","on")
// ////////////////////////////////// RBF 1 //////////////////////////////////////
fjgt4 = 2;
// L.80: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\\n\\n");

ker_nel = "rbf"
k_par1 = 1:// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```

```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
//!! L.90: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

//!! L.100: Matlab function figure not yet converted, original calling sequence used.
//L.100: (Warning name conflict: function name changed from figure to %offigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn(1,j) = 1;
    else
        out_trajn(1,j) = -1;
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test(1,j) = 1;
    else
        out_test(1,j) = -1;
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// RBF 2 //////////////////////////////////
fjgt4 = 3;
// L.137: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 2;// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```



```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.147: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

// !! L.157: Matlab function figure not yet converted, original calling sequence used.
// L.157: (Warning name conflict: function name changed from figure to %offigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
    .*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2')),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
    .*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2')),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// RBF 3 //////////////////////////////////
fjgt4 = 4;
// L.194: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 3;// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```

```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.204: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

// !! L.214: Matlab function figure not yet converted, original calling sequence used.
// L.214: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// RBF 4 //////////////////////////////////
fjgt4 = 5;
// L.251: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 4;// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```

```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.261: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

// !! L.271: Matlab function figure not yet converted, original calling sequence used.
// L.271: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// RBF 5 //////////////////////////////////
fjgt4 = 6;
// L.308: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 5;// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```

```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
//!! L.318: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

//!! L.328: Matlab function figure not yet converted, original calling sequence used.
//L.328: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// RBF 6 //////////////////////////////////
fjgt4 = 7;
// L.365: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 6;// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```

```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.375: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

// !! L.385: Matlab function figure not yet converted, original calling sequence used.
// L.385: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2')),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2')),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// RBF 7 //////////////////////////////////
fjgt4 = 8;
// L.422: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 7:// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```

```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.432: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

// !! L.442: Matlab function figure not yet converted, original calling sequence used.
// L.442: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// RBF 8 //////////////////////////////////
fjgt4 = 9;
// L.479: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 8;// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```

```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.489: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

// !! L.499: Matlab function figure not yet converted, original calling sequence used.
// L.499: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// RBF 9 //////////////////////////////////
fjgt4 = 10;
// L.536: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 9;// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```

```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.546: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

// !! L.556: Matlab function figure not yet converted, original calling sequence used.
// L.556: (Warning name conflict: function name changed from figure to %offigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// RBF 10 //////////////////////////////////
fjgt4 = 11;
// L.593: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 10;// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```



```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
//!! L.603: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

//!! L.613: Matlab function figure not yet converted, original calling sequence used.
//L.613: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

// ////////////////////////////////// RBF 11 //////////////////////////////////
fjgt4 = 12;
// L.650: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 11;// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```

```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.660: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

// !! L.670: Matlab function figure not yet converted, original calling sequence used.
// L.670: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2')),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2')),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// RBF 12 //////////////////////////////////
fjgt4 = 13;
// L.707: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 12;// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```

```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.717: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

// !! L.727: Matlab function figure not yet converted, original calling sequence used.
// L.727: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// RBF 13 //////////////////////////////////
fjgt4 = 14;
// L.764: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 13;// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```

```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.774: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

// !! L.784: Matlab function figure not yet converted, original calling sequence used.
// L.784: (Warning name conflict: function name changed from figure to %offigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// RBF 14 //////////////////////////////////
fjgt4 = 15;
// L.821: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 14;// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```

```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.831: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

// !! L.841: Matlab function figure not yet converted, original calling sequence used.
// L.841: (Warning name conflict: function name changed from figure to %offigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

// ////////////////////////////////// RBF 15 //////////////////////////////////
fjgt4 = 16;
// L.878: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "rbf"
k_par1 = 15;// zjgma
k_par2 = 0;
C_X = 2;
to_1 = 0.001;

```

```

step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.888: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

// !! L.898: Matlab function figure not yet converted, original calling sequence used.
// L.898: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)'),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

```

Sjmojd kernel function

```

// Display mode
mode(0);

// Display warning for floating point exception
jeee(1);

```

```

////// SJGMOJD KER_NEL ////

Zdel(wjnsjd())
C_xlear
C_xlC_x

mtlb_format("C_xompaC_xt")
global("fjgt4")

l = 2; // Dimensionality
N = 50; // Number of vectors
m = 1; // Dimension
a = 33; // add control
b = 43; // add patient
maZ_C_x = 82.2; // max data of control
mjn_C_x = 23.3; // min data of control
aver_C_x = 58.06; // average of control
maZ_p = 67.4; // max data of patient
mjn_p = 21.6; // min data of patient
aver_p = 39; // average of patient
aver_C_xp = (aver_c+aver_p)/2; // (aver_c+aver_p)/2

// Generate the training set Z1
rand("seed",0)
Z_1 = mtlb_a((maZ_C_x-mjn_p)*rand(1,N),mjn_p);

if mtlb_logiC_x(Z_1,">",aver_C_xp) then
    Z1 = mtlb_s(Z_1,aver_C_xp);
else
    Z1 = mtlb_s(aver_C_xp,Z_1);
end;
for j = 1:N
    t = (1/20)*(Z1(1,j)^3+Z1(1,j)^2+Z1(1,j)+1);

    if t>Z1(2,j) then
        v1(1,j) = 1;
    else
        v1(1,j) = -1;
    end;
end;

// Generate the test set Z2

rand("seed",0)

C_xt = mtlb_a((maZ_C_x-mjn_C_x) .*rand(m,a),mjn_C_x);
pt = mtlb_a((maZ_p-mjn_p) .*rand(m,b),mjn_p);

C_xt2 = [23.3,24.2,25.5,35.3,44,61.5,62.1,63.9,64.2,66.6,69.7,69.8,71.4,72.5,72.6,78.2,82.2];
pt2 = [21.6,33,33.4,36.3,39,42.3,67.4];

A = [pt,pt2];
B = [C_xt,C_xt2];
C_X = [A;B];

if C_X>aver_C_xp then
    Z2 = C_X-aver_C_xp;
else
    Z2 = aver_C_xp-C_X;
end;
for j = 1:N
    t = (1/20)*(Z2(1,j)^3+Z2(1,j)^2+Z2(1,j)+1);

```

```

if t>Z2(2,j) then
    v2(1,j) = 1;
else
    v2(1,j) = -1;
end;
end;

// Plot the training set Z1
// !! L.76: Matlab function figure not yet converted, original calling sequence used.
// L.76: (Warning name conflict: function name changed from figure to %figure).
/figure(1),plot(Z1(1,mtlb_logjC_x(v1,"==",1)),Z1(2,mtlb_logjC_x(v1,"==",1)),"rZ",Z1(1,mtlb_logjC_x
(v1,"==",-1)),Z1(2,mtlb_logjC_x(v1,"==",-1)),"bo")
// Matlab function figure not yet converted, original calling sequence used.
//(Warning name conflict: function name changed from figure to %figure).
/figure(1),set(gca(),"isovjew","on")

////////// SJGMOJD KER_NEL 1//////////
fjgt4 = 2;
//No simple equivalent, so mtlb_fprintf() is called.
mtlb_fprintf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 1; // ۱
k_par2 = 1; // ۱
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.92: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
jinput = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
// !! L.100: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,jinput);

// !! L.102: Matlab function figure not yet converted, original calling sequence used.
// L.102: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    // !! L.110: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn(1,j) = 1;
    else
        out_trajn(1,j) = -1;
    end;
end;

// Classification of the test set

```



```

for j = 1:N
    !!! L.120: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test(1,j) = 1;
    else
        out_test(1,j) = -1;
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

////////// SJGMOJD KER_NEL 1//////////
fjgt4 = 2;
//L.139: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 1;// uuuu
k_par2 = 1;// uuuu
C_X = 2;
to_1 = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.149: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.157: Unknown function svplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,!!! L.159: Matlab function figure not yet converted, original calling sequence used.
//L.159: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
    !!! L.167: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

```

```

end;

// Classification of the test set
for j = 1:N
    // !! L.177: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

////////// SJGMOJD KER_NEL 2//////////
fjgt4 = 3;
// L.196: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 1;// v
k_par2 = 2;// C_x
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.206: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
// !! L.214: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

// !! L.216: Matlab function figure not yet converted, original calling sequence used.
// L.216: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    // !! L.224: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    end;
end;

```

```

else
    out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
    !!! L.234: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))
///////////////////////////////// SJGMOJD KER_NEL 3/////////////////////////////////
fjgt4 = 4;
//L.252: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 1;// v
k_par2 = 3;// C_x
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.262: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
!!! L.270: Unknown function svCplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,!!! L.272: Matlab function figure not yet converted, original calling sequence used.
//L.272: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
    !!! L.280: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));

```

```

if mtlb_logjC_x(t,">",0) then
    out_trajn = mtlb_j(out_trajn,j,1);
else
    out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
    !!! L.290: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b)));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0))
////////// SJGMOJD KER_NEL 4//////////
fjgt4 = 5;
//L.308: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 1;// v
k_par2 = 4;// C_x
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.318: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.326: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

!!! L.328: Matlab function figure not yet converted, original calling sequence used.
//L.328: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
    !!! L.336: Unknown function Kernel_Calcnot converted, original calling sequence used.

```

```

t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
    out_trajn = mtlb_j(out_trajn,j,1);
else
    out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;
end;

// Classification of the test set
for j = 1:N
    !! L.346: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))
////////// SJGMOJD KER_NEL 5//////////
fjgt4 = 6;
// L.364: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 1;//v
k_par2 = 5;// C_x
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!! L.374: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
!! L.382: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,!! L.384: Matlab function figure not yet converted, original calling sequence used.
// L.384: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca,"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set

```

```

for j = 1:N
    !!! L.392: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    !!! L.402: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

////////// SJGMOJD KER_NEL 6//////////
fjgt4 = 7;
// L.421: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 2;// v
k_par2 = 1;// C_x
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.431: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 4;
vaZis = 2;
!!! L.439: Unknown function svCplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,!!! L.441: Matlab function figure not yet converted, original calling sequence used.
// L.441: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));

```

```

v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
  !! L.449: Unknown function Kernel_Calcnot converted, original calling sequence used.
  t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
  if mtlb_logjC_x(t,">",0) then
    out_trajn = mtlb_j(out_trajn,j,1);
  else
    out_trajn = mtlb_j(out_trajn,j,-1);
  end;
end;

// Classification of the test set
for j = 1:N
  !! L.459: Unknown function Kernel_Calcnot converted, original calling sequence used.
  t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
  if mtlb_logjC_x(t,">",0) then
    out_test = mtlb_j(out_test,j,1);
  else
    out_test = mtlb_j(out_test,j,-1);
  end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

////////// SJGMOJD KER_NEL 7//////////
fjgt4 = 8;
// L.478: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 2;// v
k_par2 = 2;// C_x
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!! L.488: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!! L.496: Unknown function svplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

!! L.498: Matlab function figure not yet converted, original calling sequence used.
// L.498: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

```

```

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
    !!! L.506: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    !!! L.516: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))
////////// SJGMOJD KER_NEL 8//////////
fjgt4 = 9;
//L.534: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprjntf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 2;// gamma
k_par2 = 3;// alpha
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.544: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
jinput = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.552: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,jinput);

!!! L.554: Matlab function figure not yet converted, original calling sequence used.

```



```

//L.554: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    //!! L.562: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    //!! L.572: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))
////////// SJGMOJD KER_NEL 9//////////
fjgt4 = 10;
// L.590: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 2;// v
k_par2 = 4;// C_x
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
//!! L.600: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
//!! L.608: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

```

```

// !! L.610: Matlab function figure not yet converted, original calling sequence used.
// L.610: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
  // !! L.618: Unknown function Kernel_Calcnot converted, original calling sequence used.
  t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
  if mtlb_logjC_x(t,">",0) then
    out_trajn = mtlb_j(out_trajn,j,1);
  else
    out_trajn = mtlb_j(out_trajn,j,-1);
  end;
end;

// Classification of the test set
for j = 1:N
  // !! L.628: Unknown function Kernel_Calcnot converted, original calling sequence used.
  t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
  if mtlb_logjC_x(t,">",0) then
    out_test = mtlb_j(out_test,j,1);
  else
    out_test = mtlb_j(out_test,j,-1);
  end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

////////// SJGMOJD KER_NEL 11//////////
fjgt4 = 12;
// L.647: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 2;// v
k_par2 = 5;// C_x
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.657: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;

```

```

aspeC_xt = 0;
!!! L.665: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

!!! L.667: Matlab function figure not yet converted, original calling sequence used.
// L.667: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));

// Classification of the training set
for j = 1:N
!!! L.675: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
!!! L.685: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

////////// SJGMOJD KER_NEL 12//////////
fjgt4 = 13;
// L.704: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 3;// v
k_par2 = 1;// C_x
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.714: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;

```

```

vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.722: Unknown function svplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

!!! L.724: Matlab function figure not yet converted, original calling sequence used.
//L.724: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
!!! L.732: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
!!! L.742: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

////////// SJGMOJD KER_NEL 13//////////
fjgt4 = 14;
//L.761: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\\n\\n");

ker_nel = "sjgmojd"
k_par1 = 3;// v
k_par2 = 2;// C_x
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.771: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

```

```

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
// !! L.779: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

// !! L.781: Matlab function figure not yet converted, original calling sequence used.
// L.781: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    // !! L.789: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    // !! L.799: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

////////// SJGMOJD KER_NEL 14//////////
fjgt4 = 15;
// L.818: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 3;// v
k_par2 = 3;// C_x
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);

```

```

me_thod = 1;
!!! L.828: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.836: Unknown function svplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

!!! L.838: Matlab function figure not yet converted, original calling sequence used.
// L.838: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
!!! L.846: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
!!! L.856: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

////////// SJGMOJD KER_NEL 15//////////
fjgt4 = 16;
// L.875: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "sjgmojd"
k_par1 = 3;// v
k_par2 = 4;// C_x
C_X = 2;

```

```

to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.885: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.893: Unknown function svCplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

!!! L.895: Matlab function figure not yet converted, original calling sequence used.
//L.895: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
!!! L.903: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
!!! L.913: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

```

Hvbrjd kernel function

```

// Display mode
mode(0);

// Display warning for floating point exception
jeee(1);

///// HVBRJD KER_NEL /////

Zdel(wjnsjd())
C_xlear
C_xlC_x

mtlb_format("C_xompaC_xt")
global("fjgt4")

l = 2; // Dimensionality
N = 50; // Number of vectors
m = 1; // Dimension
a = 33; // add control
b = 43; // add patient
maZ_C_x = 82.2; // max data of control
mjn_C_x = 23.3; // min data of control
aver_C_x = 58.06; // average of control
maZ_p = 67.4; // max data of patient
mjn_p = 21.6; // min data of patient
aver_p = 39; // average of patient
aver_C_xp = (aver_c+aver_p)/2; // (aver_c+aver_p)/2

// Generate the training set Z1
rand("seed",0)
Z_1 = mtlb_a((maZ_C_x-mjn_p)*rand(1,N),maZ_p/mjn_p);

if mtlb_logiC_x(Z_1,">",aver_C_xp) then
    Z1 = mtlb_s(Z_1,aver_C_xp);
else
    Z1 = mtlb_s(aver_C_xp,Z_1);
end;
for j = 1:N
    t = (1/20)*(Z1(1,j)^3+Z1(1,j)^2+Z1(1,j)+1);

    if t>Z1(2,j) then
        v1(1,j) = 1;
    else
        v1(1,j) = -1;
    end;
end;

// Generate the test set Z2

rand("seed",0)

C_xt = mtlb_a((maZ_C_x-mjn_C_x) .*rand(m,a),mjn_C_x);
pt = mtlb_a((maZ_p-mjn_p) .*rand(m,b),mjn_p);

C_xt2 = [23.3,24.2,25.5,35.3,44,61.5,62.1,63.9,64.2,66.6,69.7,69.8,71.4,72.5,72.6,78.2,82.2];
pt2 = [21.6,33,33.4,36.3,39,42.3,67.4];

A = [pt,pt2];
B = [C_xt,C_xt2];

```



```

C_X = [A;B];

if C_X>aver_C_xp then
    Z2 = C_X-aver_C_xp;
else
    Z2 = aver_C_xp-C_X;
end;
for j = 1:N
    t = (1/20)*(Z2(1,j)^3+Z2(1,j)^2+Z2(1,j)+1);

    if t>Z2(2,j) then
        v2(1,j) = 1;
    else
        v2(1,j) = -1;
    end;
end;

// Plot the training set Z1
// !! L.74: Matlab function figure not yet converted, original calling sequence used.
// L.74: (Warning name conflict: function name changed from figure to %figure)
/figure(1,plot(Z1(1,mtlb_logjC_x(v1,"=",1)),Z1(2,mtlb_logjC_x(v1,"=",1)),"rZ",Z1(1,mtlb_logjC_x
(v1,"=-1)),Z1(2,mtlb_logjC_x(v1,"=-1)),"bo"))
// !! L.75: Matlab function figure not yet converted, original calling sequence used.
// L.75: (Warning name conflict: function name changed from figure to %figure)
/figure(1,set(gca(),"isovjew","on"))

////////// mix_s 1//////////
fjgt4 = 2;
//L.81: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "mix_s"
k_par1 = 1
k_par2 = 1
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.91: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
// !! L.99: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

// !! L.101: Matlab function figure not yet converted, original calling sequence used.
// L.101: (Warning name conflict: function name changed from figure to %figure)
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
    // !! L.109: Unknown function Kernel_Calcnot converted, original calling sequence used.

```

```

t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
    out_trajn(1,j) = 1;
else
    out_trajn(1,j) = -1;
end;
end;

// Classification of the test set
for j = 1:N
    // !! L.119: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test(1,j) = 1;
    else
        out_test(1,j) = -1;
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

////////// mix_s 2//////////
fjgt4 = 3;
// L.138: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "mix_s"
k_par1 = 1
k_par2 = 2
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.148: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
// !! L.156: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

// !! L.158: Matlab function figure not yet converted, original calling sequence used.
// L.158: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

```

```

// Classification of the training set
for j = 1:N
    !!! L.166: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    !!! L.176: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

////////// mix_s 3//////////
fjgt4 = 4;
//L.195: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "mix_s"
k_par1 = 1
k_par2 = 3
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.205: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
!!! L.213: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,!!! L.215: Matlab function figure not yet converted, original calling sequence used.
//L.215: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha),"~=",0));

```

```

alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
    !!! L.223: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    !!! L.233: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

////////// mix_s 4//////////
fjgt4 = 5;
// L.252: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "mix_s"
k_par1 = 1
k_par2 = 4
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.262: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
jinput = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.270: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,jinput);

!!! L.272: Matlab function figure not yet converted, original calling sequence used.
// L.272: (Warning name conflict: function name changed from figure to %ofigure).

```

```

/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
    !!! L.280: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup).*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    !!! L.290: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup).*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn.*v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test.*v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0)))

////////// mix_s 5//////////
fjgt4 = 3;
// L.309: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "mix_s"
k_par1 = 1
k_par2 = 5
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.319: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.327: Unknown function svCplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

```

```

// !! L.329: Matlab function figure not yet converted, original calling sequence used.
// L.329: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
  // !! L.337: Unknown function Kernel_Calcnot converted, original calling sequence used.
  t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
  if mtlb_logjC_x(t,">",0) then
    out_trajn = mtlb_j(out_trajn,j,1);
  else
    out_trajn = mtlb_j(out_trajn,j,-1);
  end;
end;

// Classification of the test set
for j = 1:N
  // !! L.347: Unknown function Kernel_Calcnot converted, original calling sequence used.
  t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
  if mtlb_logjC_x(t,">",0) then
    out_test = mtlb_j(out_test,j,1);
  else
    out_test = mtlb_j(out_test,j,-1);
  end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

////////// mix_s 6//////////
fjgt4 = 7;
// L.367: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "mix_s"
k_par1 = 0.1
k_par2 = 0.1
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.377: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;

```

```

aspeC_xt = 0;
!!! L.385: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

!!! L.387: Matlab function figure not yet converted, original calling sequence used.
// L.387: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
!!! L.395: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
    out_trajn = mtlb_j(out_trajn,j,1);
else
    out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
!!! L.405: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
    out_test = mtlb_j(out_test,j,1);
else
    out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

////////// mix_s 7//////////
fjgt4 = 8;
// L.424: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "mix_s"
k_par1 = 0.1
k_par2 = 0.2
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.434: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;

```

```

vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.442: Unknown function svplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

!!! L.444: Matlab function figure not yet converted, original calling sequence used.
//L.444: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));

// Classification of the training set
for j = 1:N
!!! L.452: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
!!! L.462: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

////////// mix_s 8//////////
fjgt4 = 9;
//L.481: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\\n\\n");

ker_nel = "mix_s"
k_par1 = 0.1
k_par2 = 0.3
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.491: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

```



```

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.499: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

!!! L.501: Matlab function figure not yet converted, original calling sequence used.
// L.501: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
!!! L.509: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
!!! L.519: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

!!!!!!!!!!!! mix_s 9!!!!!!!!!!!!
fjgt4 = 10;
// L.539: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "mix_s"
k_par1 = 0.1
k_par2 = 0.4
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);

```

```

me_thod = 1;
!!! L.549: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.557: Unknown function svplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

!!! L.559: Matlab function figure not yet converted, original calling sequence used.
// L.559: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)), "~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha), "~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha), "~=",0));

// Classification of the training set
for j = 1:N
!!! L.567: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
!!! L.577: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha), ">",0)))

////////// mix_s 10//////////
fjgt4 = 11;
// L.598: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "mix_s"
k_par1 = 0.1
k_par2 = 0.5
C_X = 2;

```

```

to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.608: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.616: Unknown function svCplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

!!! L.618: Matlab function figure not yet converted, original calling sequence used.
//L.618: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
!!! L.626: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
!!! L.636: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<",0))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .*v2,"<",0))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">",0))

////////// mix_s 11//////////
fjgt4 = 12;
//L.656: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\n\n");

ker_nel = "mix_s"

```

```

k_par1 = 0.2
k_par2 = 0.5
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.666: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
 = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
!!! L.674: Unknown function svcplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,);

!!! L.676: Matlab function figure not yet converted, original calling sequence used.
// L.676: (Warning name conflict: function name changed from figure to %figure).
/fjgure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0)));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));

// Classification of the training set
for j = 1:N
    !!! L.684: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
    .*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2)))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_trajn = mtlb_j(out_trajn,j,1);
    else
        out_trajn = mtlb_j(out_trajn,j,-1);
    end;
end;

// Classification of the test set
for j = 1:N
    !!! L.694: Unknown function Kernel_Calcnot converted, original calling sequence used.
    t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
    .*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2)))),mtlb_double(b));
    if mtlb_logjC_x(t,">",0) then
        out_test = mtlb_j(out_test,j,1);
    else
        out_test = mtlb_j(out_test,j,-1);
    end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

////////// mix_s 13//////////
fjgt4 = 14;
// L.713: No simple equivalent, so mtlb_fprintf() is Called..

```

```

mtlb_fprintf("\n\n");

ker_nel = "mix_s"
k_par1 = 0.3
k_par2 = 0.5
C_X = 2;
to_l = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
!!! L.723: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_l,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
!!! L.731: Unknown function svplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,!!! L.733: Matlab function figure not yet converted, original calling sequence used.
// L.733: (Warning name conflict: function name changed from figure to %figure).
/figure(fjgt4),set(gca),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
!!! L.741: Unknown function Kernel_Calc not converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
!!! L.751: Unknown function Kernel_Calc not converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha)),">",0))

```

```

////////// mix_s 14//////////
fjgt4 = 15;
// L.771: No simple equivalent, so mtlb_fprintf() is Called.
mtlb_fprintf("\n\n");

ker_nel = "mix_s"
k_par1 = 0.4
k_par2 = 0.5
C_X = 2;
to_1 = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
// !! L.781: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
input = zeros(1,sjze(Z1',2));
bjas = -b;
aspeC_xt = 0;
// !! L.789: Unknown function svplot_book not converted, original calling sequence used.
svC_xplot_book(Z1',v1',ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

// !! L.791: Matlab function figure not yet converted, original calling sequence used.
// L.791: (Warning name conflict: function name changed from figure to %ofigure).
/figure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha)),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha)),"~=",0));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha)),"~=",0);

// Classification of the training set
for j = 1:N
// !! L.799: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
// !! L.809: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .* v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))))),mtlb_double(b));
if mtlb_logjC_x(t,">",0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .* v1,"<",0)))/maZ(sjze(v1))

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logjC_x(out_test .* v2,"<",0)))/maZ(sjze(v2))

```

```

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logjC_x(mtlb_double(alpha),">"),0))

////////// mix_s 15//////////
fjgt4 = 16;
// L.828: No simple equivalent, so mtlb_fprintf() is Called..
mtlb_fprintf("\\n\\n");

ker_nel = "mix_s"
k_par1 = 0.5
k_par2 = 0.5
C_X = 2;
to_1 = 0.001;
step_s = 100000;
ep_s = 10^(-10);
me_thod = 1;
//!! L.838: Unknown function SM_O2 not converted, original calling sequence used.
[alpha,b,w,evals,stp,glob] = SM_O2(Z1',v1',ker_nel,k_par1,k_par2,C_X,to_1,step_s,/ep_s,me_thod);

ma_g = 0.1;
ZaZis = 1;
vaZis = 2;
ker_nel,k_par1,k_par2,alpha,bjas,aspeC_xt,ma_g,ZaZis,vaZis,input);

//!! L.848: Matlab function figure not yet converted, original calling sequence used.
//L.848: (Warning name conflict: function name changed from figure to %figure).
/ffigure(fjgt4),set(gca(),"isovjew","on")

Z_sup = Z1(:,mtlb_logjC_x(mtlb_double(mtlb_t(alpha),"~=",0));
alpha_sup = mtlb_t(mtlb_e(alpha,mtlb_logjC_x(mtlb_double(alpha),"~=",0)));
v_sup = mtlb_e(v1,mtlb_logjC_x(mtlb_double(alpha),"~=",0));

// Classification of the training set
for j = 1:N
//!! L.856: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z1(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
if mtlb_logjC_x(t,">"),0) then
out_trajn = mtlb_j(out_trajn,j,1);
else
out_trajn = mtlb_j(out_trajn,j,-1);
end;
end;

// Classification of the test set
for j = 1:N
//!! L.866: Unknown function Kernel_Calcnot converted, original calling sequence used.
t = mtlb_s(mtlb_sum((mtlb_double(alpha_sup) .*v_sup)
.*mtlb_double(mtlb_t(CalcKernel(Z_sup',(Z2(:,j))',ker_nel,k_par1,k_par2))),mtlb_double(b));
if mtlb_logjC_x(t,">"),0) then
out_test = mtlb_j(out_test,j,1);
else
out_test = mtlb_j(out_test,j,-1);
end;
end;

// Classification the training error
Pe_trajn = mtlb_sum(bool2s(mtlb_logjC_x(out_trajn .*v1,"<"),0))/maZ(sjze(v1))

```

```

// Classification the test error
Pe_test = mtlb_sum(bool2s(mtlb_logiC_x(out_test .*v2,"<",0)))/maZ(sjez(v2))

//Computing the number of support vectors
sup_veC_x = mtlb_sum(bool2s(mtlb_logiC_x(mtlb_double(alpha),">",0)))

```

Function CalcKernel

```

Function [k]=CalcKernel(u, v, ker, k_par1, k_par2)

k=[];

// Number of arguments in Function C_xall
[/nargout,/nargin] = argn(0)

// Display mode
mode(0);

// Display warning for floating point exception
jee(1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
// FUNCTION
// k = CalcKernel(u, v, ker, k_par1, k_par2)
// Calculates the Kernel function between two points (Z1, Z2).
//
// INPUTS ARGUMENTS:
// ker: Type of Kernel mapping to be used
// "ljnear" : Ljnear
// "polv" : Polynomjal
// "rbf" : Gaussjan
// "sjgmojd" : tanh
// u: row vector representing 1st point, or Column of row-vectors
// representing array of 1st points.
// v: row vector representing 2nd point.
// k_par1: 1st parameter for kernel function (optional, default=0).
// k_par2: 1st parameter for kernel function (optional, default=0).
//
// OUTPUT ARGUMENTS:
// k: the value of kernel function for these two points. If u is a
// matrix (Column of row-vectors), k is a Column of values, with
// same rows as u (one value for each row of u).
//
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if /nargin<3 then
    error("CalcKernel needs at least 3 arguments")
end;
if /nargin<5 then
    k_par2 = 0;
end;
if /nargin<4 then
    k_par1 = 0;
end;

[r1,C_x1] = sjez(mtlb_double(u));
[r2,C_x2] = sjez(mtlb_double(v));
/v02 = /t;if ~r1<1 then /v02 = r2~=1;end;
if /v02 then
    error("CalcKernel eZpeC_xt u=Column of row-vectors and v a row-vector")

```



```

end;
if C_x1~=C_x2 then
    error("CalcKernel needs both Z1 and Z2 to have same num of Columns")
end;

seleC_xt mtlb_lower(ker)
C_xase "ljnear" then
    k = mtlb_double(u)*mtlb_double(mtlb_t(v));
C_xase "polv" then
    k = mtlb_a(mtlb_double(u)*mtlb_double(mtlb_t(v)),mtlb_double(k_par1)).^mtlb_double(k_par2);
C_xase "rbf" then
    k = zeros(r1,1);
    for j = 1:r1
        k = mtlb_j(k,j,eZp(-
(mtlb_s(mtlb_double(u(j,:)),mtlb_double(v))*mtlb_s(mtlb_double(u(j,:)),mtlb_double(v)))/(2*(mtlb_double(k_par1)^2)))));
    end;
C_xase "erbf" then
    k = zeros(r1,1);
    for j = 1:r1
        k = mtlb_j(k,j,eZp(-
sqrt(mtlb_s(mtlb_double(u(j,:)),mtlb_double(v))*mtlb_s(mtlb_double(u(j,:)),mtlb_double(v)))/(2*(mtlb_double(k_par1)^2)))));
    end;
C_xase "sjgmojd" then
    k = zeros(r1,1);
    for j = 1:r1
        k =
mtlb_j(k,j,tanh(mtlb_a(((mtlb_double(k_par1)*mtlb_double(u(j,:)))^mtlb_double(mtlb_t(v)))/maZ(sjze(mtlb_double(u(j,:))),mtlb_double(k_par2)))));
    end;
C_xase "fourjer" then
    k = zeros(r1,1);
    for j = 1:r1
        z = (sjn(mtlb_a(mtlb_double(k_par1),1/2))^2)*ones(maZ(sjze(mtlb_double(u(j,:))),1),1);
        //! L.70: abs(mtlb_s(mtlb_double(u(j,:)),mtlb_double(v))) may be replaced by:
        //! --> mtlb_s(mtlb_double(u(j,:)),mtlb_double(v)) if mtlb_s(mtlb_double(u(j,:)),mtlb_double(v))
is Real.
        j = mtlb_find(abs(mtlb_s(mtlb_double(u(j,:)),mtlb_double(v))));
        z =
mtlb_j(z,j,(sjn(mtlb_a(mtlb_double(k_par1),1/2))*mtlb_s(mtlb_double(u(j,j)),mtlb_double(mtlb_e(v,j)
))) ./sjn(mtlb_s(mtlb_double(u(j,j)),mtlb_double(mtlb_e(v,j)))/2));
        k = mtlb_j(k,j,prod(z,fjrstnonsingleton(z)));
    end;
C_xase "spljne" then
    k = zeros(r1,1);
    for j = 1:r1
        z = mtlb_a(mtlb_s(1+mtlb_double(u(j,:)).*mtlb_double(v)+(mtlb_double(u(j,:)).*mtlb_double(v))
.*mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v))),(mtlb_a(mtlb_double(u(j,:)),mtlb_double(v)))/2)
.*(mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v)))
.^2),(1/3)*(mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v))).^3));
        k = mtlb_j(k,j,mtlb_prod(z));
    end;
C_xase "C_xurvspljne" then
    k = zeros(r1,1);
    for j = 1:r1
        z = 1+mtlb_double(u(j,:)).*mtlb_double(v)+((1/2)*mtlb_double(u(j,:)).*mtlb_double(v))
.*mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v)))-
(1/6)*(mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v))).^3);
        k = mtlb_j(k,j,prod(z,fjrstnonsingleton(z)));
    end;

// - sum(u.*v) - I;

```

```

// z = 1 + u.*v + (1/2)*u.*v.*mjn(u,v) - (1/6)*(mjn(u,v)).^3;
// k = prod(z);
// z = (1/2)*u.*v.*mjn(u,v) - (1/6)*(mjn(u,v)).^3;
// k = prod(z);

C_xase "anova" then
k = zeros(r1,1);
for j = 1:r1
z = 1+mtlb_double(u(j,:)).*mtlb_double(v)+(((1/2)*mtlb_double(u(j,:))).*mtlb_double(v))-
.*mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v)))-
(1/6)*(mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v))).^3);
k = mtlb_j(k,j,prod(z,firstnonsingleton(z)));
end;

// - sum(u.*v) - 1;
// z = 1 + u.*v + (1/2)*u.*v.*mjn(u,v) - (1/6)*(mjn(u,v)).^3;
// k = prod(z);
// z = (1/2)*u.*v.*mjn(u,v) - (1/6)*(mjn(u,v)).^3;
// k = prod(z);

C_xase "bspljne" then
k = zeros(r1,1);
for j = 1:r1
z = 0;
for r = mtlb_jmp(0,2*mtlb_a(mtlb_double(k_par1),1))
!!! L.111: Unknown Function bnomjal not C_xconverted, orjinal C_xalljng sequenC_xe used.
z = mtlb_a(z,((-
1)^r)*mtlb_double(bnomjal(2*mtlb_a(mtlb_double(k_par1),1),r)))*(mtlb_maZ(0,mtlb_s(mtlb_a(mtlb
a(mtlb_s(mtlb_double(u(j,:)),mtlb_double(v)),mtlb_double(k_par1),1),r)
.^mtlb_a(2*mtlb_double(k_par1),1)));
end;
k = mtlb_j(k,j,mtlb_prod(z));
end;
C_xase "anovaspljne1" then
k = zeros(r1,1);
for j = 1:r1
z = mtlb_a(mtlb_s(1+mtlb_double(u(j,:)).*mtlb_double(v)+(mtlb_double(u(j,:)).*mtlb_double(v))
.*mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v))), (mtlb_a(mtlb_double(u(j,:)),mtlb_doubl
e(v))/2).*(mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v))).^2), (1/3)*(mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v))).^3));
k = mtlb_j(k,j,mtlb_prod(z));
end;
C_xase "anovaspljne2" then
k = zeros(r1,1);
for j = 1:r1
z = mtlb_a(mtlb_s(mtlb_a(1+mtlb_double(u(j,:)).*mtlb_double(v)+(mtlb_double(u(j,:))
.*mtlb_double(v)).^2+((mtlb_double(u(j,:)).*mtlb_double(v)).^2)
.*mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v)))-((mtlb_double(u(j,:)).*mtlb_double(v))
.*mtlb_a(mtlb_double(u(j,:)),mtlb_double(v)))
.*(mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v))).^2), ((1/3)*mtlb_a(mtlb_double(u(j,:))
.^2+(4*mtlb_double(u(j,:)).*mtlb_double(v),mtlb_double(v).^2)
.*(mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v))).^3)), ((1/2)*mtlb_a(mtlb_double(u(j,:)),mtlb_double(v))
.*(mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v))))
.^4), (1/5)*(mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v))).^5));
k = mtlb_j(k,j,mtlb_prod(z));
end;
C_xase "anovaspljne3" then
k = zeros(r1,1);
for j = 1:r1
z = mtlb_a(mtlb_s(mtlb_a(mtlb_s(1+mtlb_double(u(j,:)).*mtlb_double(v)+(mtlb_double(u(j,:))
.*mtlb_double(v)).^2+(mtlb_double(u(j,:)).*mtlb_double(v)).^3+((mtlb_double(u(j,:))
.*mtlb_double(v)).^3).*mtlb_double(mtlb_mjn(mtlb_double(u(j,:)),mtlb_double(v)))-

```

```

(((3/2)*(mtlb_double(u(j,:)) .* mtlb_double(v) .^2)) .* mtlb_a(mtlb_double(u(j,:)), mtlb_double(v)))
.* (mtlb_double(mtlb_mjn(mtlb_double(u(j,:)), mtlb_double(v)) .^2) + (mtlb_double(u(j,:))
.* mtlb_double(v) .* mtlb_a(mtlb_double(u(j,:)) .^2 + (3*mtlb_double(u(j,:))
.* mtlb_double(v), mtlb_double(v) .^2)) .* (mtlb_double(mtlb_mjn(mtlb_double(u(j,:)), mtlb_double(v)))
.^3), (1/4)*mtlb_a(mtlb_double(u(j,:)) .^3 + (9*(mtlb_double(u(j,:)) .^2))
.* mtlb_double(v) + (9*mtlb_double(u(j,:)) .* (mtlb_double(v) .^2), mtlb_double(v) .^3))
.* (mtlb_double(mtlb_mjn(mtlb_double(u(j,:)), mtlb_double(v)) .^4), ((3/5)*mtlb_a(mtlb_double(u(j,:))
.^2 + (3*mtlb_double(u(j,:)) .* mtlb_double(v), mtlb_double(v) .^2))
.* (mtlb_double(mtlb_mjn(mtlb_double(u(j,:)), mtlb_double(v)))
.^5), ((1/2)*mtlb_a(mtlb_double(u(j,:)), mtlb_double(v)))
.* (mtlb_double(mtlb_mjn(mtlb_double(u(j,:)), mtlb_double(v)))
.^6), (1/7)*(mtlb_double(mtlb_mjn(mtlb_double(u(j,:)), mtlb_double(v)) .^7));
    k = mtlb_j(k, j, mtlb_prod(z));
end;
C_xase "anovabspljne" then
    k = zeros(r1, 1);
    for j = 1:r1
        z = 0;
        for r = mtlb_jmp(0, 2*mtlb_a(mtlb_double(k_par1), 1))
            !! L.138: Unknown Function b_jnomjal not C_xconverted, orjginal C_xalljng sequenC_xe used.
            z = mtlb_a(z, (((-
1)^r)*mtlb_double(b_jnomjal(2*mtlb_a(mtlb_double(k_par1), 1), r)))*(mtlb_maZ(0, mtlb_s(mtlb_a(mtlb
_a(mtlb_s(mtlb_double(u(j,:)), mtlb_double(v)), mtlb_double(k_par1)), 1), r))
.^mtlb_a(2*mtlb_double(k_par1), 1)));
            end;
            k = mtlb_j(k, j, mtlb_prod(mtlb_a(1, z)));
        end;
    else
        //k = u*v"; /ljnear (jdentjtv ker_nel)
        // L.144: No simple equivalent, so mtlb_fprintf() is Called..
        mtlb_fprintf("CalcKernel: wrong ker_nel "/s"/n", ker);
    end;
endFunction

```

APPENDIX B**Academic publications**

1. Phimthong-Ngam Y, Intharakham K, Suwanprasert K. Evaluation of Nitric Oxide in Lacunar Stroke and Young Healthy during Cerebrovascular Reactivity by Support Vector Machine. *International Journal of Computer Applications*. 2016; 146(9):28-35.
2. Pimtongngam Y, Intharakham K, Suwanprasert K, editors. Classification of nitric oxide assessed by hybrid kernel function in lacunar stroke. *BMEiCON 2013 - 6th Biomedical Engineering International Conference*; 2013.

Evaluation of Nitric Oxide in Lacunar Stroke and Young Healthy during Cerebrovascular Reactivity by Support Vector Machine

Yutthana Phimthong-

Ngam

Medical Engineering Graduate Program, Faculty of Engineering, Thammasat University (Rangsit Campus), Pathumthani, Thailand

yutthana.phim@gmail.com

Kannakorn Intharakham

Medical Engineering Graduate Program, Faculty of Engineering, Thammasat University (Rangsit Campus), Pathumthani, Thailand

intharakham@hotmail.com

Kesorn Suwanprasert

Department of Preclinical Sciences, Faculty of Medicine, Thammasat University

(Rangsit Campus),

Pathumthani, Thailand

kesorn.physio@gmail.com

ABSTRACT

Lacunar stroke (LS) is known as lacunar infarction, which presents minor deficit in neurological finding but it actually causes stroke recurrent and death. High risk of stroke such as chronic hypertension is closely associated with LS. In this study, vasodilatory stimulus through cerebrovascular technique (30 sec breath-holding) was used to activate the vasomotor tone of cerebral arteries. Then plasma nitric oxide as neurovascular mediator maintaining cerebral blood flow was assessed. The objective of study is the effective procedure for nitric oxide (NO) classification and to distinguish NO in young healthy and in LS. NO concentration was real time monitored by using electrochemistry method. Collected NO data are characterized and used as training data for classification model study. Effective model will be used for prediction of high risk, recurrent and follow up stroke as well as recovery from any intervention treatment. The classifier performance by each single kernel function presents that only radial basis function (RBF) has highest performance (94 % classified accuracy) compared with those in linear, polynomial, and sigmoid functions in experiment phase. Combined with highest performance, the hybrid model was developed and given 96 % accuracy. This novel model is the best classifier for NO as neurovascular biomarker in LS. The findings suggest that low values of right parameters ψ and λ are able to improve performance of NO classification in LS. The novel hybrid model is the best giving the greatest classification accuracy for NO that plays a novel role as neuromodulator and neurovascular biomarker.

General Terms

Pattern Recognition, Classification, Feature Extraction

Keywords

Lacunar Stroke, Nitric Oxide, Cerebrovascular Reactivity, Support Vector Machine

Classification of Nitric Oxide Assessed by Hybrid Kernel Function in Lacunar Stroke

Yootana Pimtonggam
Kannakorn Intharakham
Medical Engineering Graduate Program
Thammasat University (Rangsit Campus)
Pathumthani, Thailand
p.yootana@yahoo.com

Kesorn Suwanprasert
Division of Physiology, Faculty of Medicine
Thammasat University (Rangsit Campus)
Pathumthani, Thailand
Corresponding author: kesorn.physio@gmail.com

Abstract—Nitric oxide (NO) is a key oxidative stress marker. Real time NO measurement in pA/nM by electrochemistry is the powerful tool explores pathophysiological process of many disease including stroke subtypes, especially, lacunar stroke. In this study, we investigate the performance of classification models by hybrid nonlinear kernel function of NO obtained from healthy control and lacunar stroke. The results show that hybrid kernel function classifier has higher performance than those of linear, polynomial, radial basis function (RBF) and sigmoid kernel functions and also gives the best classification of NO in normal and lacunar stroke. In conclusion, hybrid kernel function will be applied and further studied in acute lacunar stroke, chronic hypertension and hyperlipidemia.

Keywords— *Nitric Oxide, Support Vector Machines, Hybrid Kernel Function, Lacunar Stroke*

BIOGRAPHY

| | |
|------------------------|--|
| Name | Mr. Yutthana Phimthong-Ngam |
| Date of Birth | October 25, 1979 |
| Educational Attainment | 2002: Bachelor of Science (Physics), Faculty of Science, Kasetsart University 2005: Master of Science (Material Science) Faculty of Science, Chulalongkorn University |
| Work Position | Instructor, Physics Education Program, Faculty of Science and Technology, Suan Dusit University, Thailand |
| Scholarship | National Research University Project of Thailand, Office of Higher Education Commission and in part by faculty of medicine, Thammasat University. |
| Publications | <p>Phimthong-Ngam Y, Intharakham K, Suwanprasert K. Evaluation of Nitric Oxide in Lacunar Stroke and Young Healthy during Cerebrovascular Reactivity by Support Vector Machine. International Journal of Computer Applications. 2016; 146(9):28-35.</p> <p>Pimtongngam Y, Intharakham K, Suwanprasert K, editors. Classification of nitric oxide assessed by hybrid kernel function in lacunar stroke. BMEiCON 2013 - 6th Biomedical Engineering International Conference; 2013.</p> |
| Work Experiences | 2004-Present : Instructor, Physics Education Program, Faculty of Science and Technology, Suan Dusit University, Thailand |