

Spring 2014

## Faults Identification in Three-Phase Induction Motors Using Support Vector Machines

Rama Hammo  
*Bowling Green State University*

Follow this and additional works at: [https://scholarworks.bgsu.edu/ms\\_tech\\_mngmt](https://scholarworks.bgsu.edu/ms_tech_mngmt)



Part of the [Artificial Intelligence and Robotics Commons](#), [Electrical and Computer Engineering Commons](#), and the [Electro-Mechanical Systems Commons](#)

---

### Recommended Citation

Hammo, Rama, "Faults Identification in Three-Phase Induction Motors Using Support Vector Machines" (2014). *Master of Technology Management Plan II Graduate Projects*. 1.  
[https://scholarworks.bgsu.edu/ms\\_tech\\_mngmt/1](https://scholarworks.bgsu.edu/ms_tech_mngmt/1)

This Dissertation/Thesis is brought to you for free and open access by the College of Technology, Architecture and Applied Engineering at ScholarWorks@BGSU. It has been accepted for inclusion in Master of Technology Management Plan II Graduate Projects by an authorized administrator of ScholarWorks@BGSU.

# **Faults Identification in Three-Phase Induction Motors Using Support Vector Machines**

Rama Hammo

A Project

Submitted to the Graduate College of Bowling Green  
State University in partial fulfillment of  
the requirements for the degree of

Master of Technology Management

May 2014

Committee:

Dr. Sri Kolla, Advisor

Dr. David Border

Dr. Todd Waggoner

## *Abstract*

Induction motor is one of the most important motors used in industrial applications. The operating condition may sometime lead the machine into different fault situations. The main types of external faults experienced by these motors are over loading, single phasing, unbalanced supply voltage, locked rotor, phase reversal, ground fault, under voltage and over voltage. The machine should be shut down when a fault is experienced to avoid damage and for the safety of the workers. Computer based relays monitor the machine and disconnect it during the faults. The relay logic used to identify these faults requires sophisticated signal processing techniques for fast and reliable operation. Artificial Intelligence (AI) techniques such as Artificial Neural Networks (ANN) have been applied in induction motor relays. Though the ANN based methods are reliable, the selection of the ANN structures and training is time consuming. Recently it is observed that the AI techniques such as Support Vector Machines (SVM) are alleviating some of the limitations of ANN method.

The objectives of this study are to develop a SVM based induction motor external faults identifier and study its performance with real-time induction motor faults data. Data collected from a 1/3 hp, 208 V three-phase squirrel cage induction motor is used in this project. Radial Bases Function kernel is used to train and test the SVM, though the effect of other Kernel functions was also studied. The proposed SVM method uses RMS values of three-phase voltages and currents as inputs. The testing results showed the efficacy of the SVM based method for identifying the external faults experienced by 3-phase induction motors. It is observed that the performance of the SVM based method is better than the ANN based method both in model creation and testing accuracy.

## *ACKNOWLEDGMENTS*

Foremost, I would like to express my sincere gratitude to my advisor Dr. Sri Kolla for the continuous support of my study and research, for his patience, motivation, and immense knowledge. I would like to thank my project committee members Dr. David Border and Dr. Todd Waggoner for their help in this project. I would also like to thank Dr. Todd Waggoner for his help in providing me with practical engineering technology experience. I would also like to thank the faculty and staff in the College of Technology, Architecture and Applied Engineering for their support. Finally, I would like to thank my family and friends for their help and support in my study.

TABLE OF CONTENTS

	Page
CHAPTER 1: INTRODUCTION.....	1
Context of the Problem.....	1
Statement of the Problem.....	3
Objectives of the Study.....	3
Significance of the Study.....	4
Limitation of the Study.....	5
Definition of Terms.....	5
CHAPTER 2: REVIEW OF THE LITERATURE.....	7
Introduction.....	7
Faults in Induction Motors.....	7
Overloading.....	8
Single Phasing.....	8
Unbalanced Supply Voltage.....	9
Locked Rotor.....	9
Phase Reversal.....	10
Ground Fault.....	10
Under Voltage.....	10
Over Voltage.....	11
Power System Relays.....	11
Artificial Intelligence Techniques.....	13
Artificial Neural Networks.....	13

Support Vector Machine .....	16
Literature Review of AI Techniques for Faults Identification in Power Systems .....	18
Summary .....	19
CHAPTER 3: METHODOLOGY .....	20
Restatement of the Problem .....	20
Restatement of the Objective .....	20
Procedure of the Study .....	21
Support Vector Machine .....	21
Libsvm Software to Implement SVM Technique .....	24
Svm-toy.exe .....	25
Scaling and Training the SVM .....	26
Testing the SVM .....	27
Induction Motors Faults Data Collection and SVM Implementation .....	28
Data Collection Procedure .....	30
Summary .....	33
CHAPTER 4: FINDINGS.....	34
Training and Testing with the SVM .....	34
Training and Testing of the Split Data.....	40
Training Data of 80% Samples and 20% Testing Data. ....	40
Training Data of 90% Samples and 10% Testing Data .....	42
Testing of 21 Samples with 90% Training Model Data .....	45
Effect of Different Kernel Functions .....	46
SVM Versus ANN Classification Performance.....	47

Summary .....	49
CHAPTER 5: Conclusion .....	50
Summary and Conclusions .....	50
Recommendation for Future Work .....	51
Reference .....	52
Appendix A .....	57

## LIST OF FIGURES

Figure	Page
Figure 2.1. Three-phase Voltage and Currents of Induction Motor Under Unbalanced Supply Voltage .....	9
Figure 2.2. Block Diagram of a Microprocessor Based Relay Scheme .....	12
Figure 2.3. Single Neuro .....	14
Figure 2.4. Feed Forward Neural Network .....	15
Figure 2.5. Inputs and Outputs of the ANN .....	16
Figure.2.6. Classification of Two Classes Using SVM .....	17
Figure 3.1. SVM Classification.....	22
Figure 3.2. Svm-toy Screen Shot .....	25
Figure 3.3. Grid Search for $\gamma$ and $c$ of the SVM Technique .....	28
Figure 3.4. Inputs and Outputs of the SVM.....	29
Figure 3.5. VARIAC .....	31
Figure 3.6. Prony Brake .....	31
Figure.3.7. Complete System.....	33
Figure 4.1. Gnuplot Output of Optimum $C$ and $\gamma$ for (-1, 1) Scaling .....	35
Figure 4.2. Screen-shot of Prediction Command for 21 Test Cases Scaling (-1, 1).....	35
Figure 4.3. Gnuplot Output of Optimum $C$ and $\gamma$ for (-1, 0) Scaling .....	36
Figure 4.4. Screen-shot of Prediction Command for 21 Test Case .....	36
Figure 4.5. No Fault Condition .....	38
Figure 4.6. Single Phasing Case .....	38
Figure 4.7. Unbalanced Voltage Fault. ....	38



Figure 4.8. Under Voltage Fault .....	39
Figure 4.9. Over Voltage Fault .....	39
Figure 4.10. Locked Rotor Fault.....	39
Figure 4.11. Overload Fault .....	40
Figure 4.12. Gnuplot Output of Optimum C and $\gamma$ for (-1,1) Scaling .....	41
Figure 4.13. Screen-shot of Prediction Command for 20% Test Cases.....	41
Figure 4.14. Gnuplot Output of Optimum C and $\gamma$ for (-1,0) Scaling .....	42
Figure 4.15. Screen-shot of Prediction Command for 20% Test Cases.....	42
Figure 4.16. Gnuplot Output of Optimum C and $\gamma$ for (-1,1)Scaling .....	43
Figure 4.17. Screen-shot of Prediction Command for 10% Test Cases.....	44
Figure 4.18. Gnuplot Output of Optimum C and $\gamma$ for (-1,0) Scaling .....	44
Figure 4.19. Screen-shot of Prediction Command for 10% Test Cases.....	45
Figure 4.20. Screen-shot of Prediction Command for 21 Samples Test Cases.....	45
Figure 4.21. Screen-shot of Prediction Command for 21 Samples Test Cases.....	46

## List of Tables

Figure 4.21. Screen-shot of Prediction Command for 21 Samples Test Cases.....	46
Table 3.1. Number of Training Patterns per Fault.....	32
Table 4.1. Testing the SVM.....	37
Table 4.2. Testing Samples.....	40
Table 4.3. Testing Samples.....	43
Table 4.4. Kernel Function Verification.....	47

## Chapter 1: Introduction

### *Context of the Problem*

The induction motor is one of the most important motors used in industrial applications. It is used to convert electrical energy into mechanical energy. Its low cost and high performance in addition to its reliability make them the most popular alternating current motors used in the industrial and commercial fields (Wan & Hong, 2001). These motors have the flexibility of application fields; they can be used in low power applications such as household appliances or in large power applications such as petroleum industry. Despite the fact of high reliability of induction motors, the operating conditions may expose the machine into different fault conditions. These faults may lead to machine shut down, thus causing industrial production losses.

Avoiding the unexpected shutdowns is important task for industries. In order to achieve this task the induction motor has to be continually monitored to identify faults in early stages. Detection of these faults in advance enables the maintenance engineers to take the necessary corrective actions as quickly as possible. The main types of external faults experienced by an induction motor are over-loading, single phasing, unbalanced supply voltage, locked rotor, phase reversal, ground faults, and under/over voltage.

The history of fault monitoring and fault isolation started with the use of electromechanical relays to protect the motor against faults (Elmore, 2004). However, these electromechanical relays are slow in operation, consume significant power, and require periodic maintenance due to mechanical parts involved. The introduction of semiconductor technology

had a positive impact on the induction motor protection field, and replaced the electromechanical relays by solid state relays as their operating speed is faster, consume less power, cheaper to manufacture and provide more reliability. Development of microprocessor technology in late 1970's enabled their application in the induction motor protective relays (IEEE, 1997). These relays allow the protection logic to be implemented by software programs.

The recent development in computer software based on intelligent systems attracted the attention of relay engineers to use them in the diagnosis of faults in power system components such as induction motors (Kezunovic, 1997). The systems based on Artificial Intelligence (AI) techniques take the place of the human experts by providing with the required information about the system performance. These AI techniques include expert systems, Artificial Neural Networks (ANN) (Wasserman, 1989) and others (Mellit & Kalogirou, 2008). In earlier research projects at Bowling Green State University (BGSU), it was shown that ANN could be used to identify fault conditions in a simulated motor (Kolla & Varatharasa, 2000) and an actual motor (Kolla & Altman, 2007). The limitations of ANN based fault-detection systems are that they require significant time for training the network with the data, and their limited reliability of fault identification for untrained data. This project will use a new AI technique called Support Vector Machines (SVM) which is shown to provide better performance in some applications (Brun & Ernst, 2004). The training process in SVM is more efficient than ANN (E. Avci & D. Avci, 2009). Statistical learning theory forms the backbone of SVM technique. The technique attempts to develop a more reliable classification system that can train faster. SVM will be applied to identify external fault conditions in a three-phase induction motor. It will be trained with fault current and voltage signals obtained from a real induction motor and the efficacy of the method will be tested.

### *Statement of the Problem*

This project develops a fault detection method for three-phase induction motors to detect external faults experienced by the motor using SVM technique. The developed SVM method is tested with real-time faults data from a 1/3 hp three-phase squirrel cage induction motor. Effect of different SVM parameters on the training and testing of the induction motor data is studied. This study also investigates the classification performance characteristics of the proposed SVM based method in comparison to ANN based method.

### *Objectives of the Study*

1. Study the usage of SVM method for pattern recognition applications and investigate the effect of different parameters on the training and testing of the method.
2. Understand the usage of Libsvm software program for pattern recognition applications.
3. Understand the characteristics of external faults on three-phase induction motors.
4. Apply the SVM method for identifying external faults in three-phase induction motors.
5. Study the performance of the proposed SVM based induction motor fault identification technique in comparison to ANN based method.

### *Significance of the Study*

Induction motors are the most common electrical machines, because of their relatively low manufacturing cost and the ease of control. They represent about 80% of the electromechanical energy conversion machines (Wan & Hong, 2001). As indicated before, identifying faults in these motors and protecting them is an important aspect to reduce losses. AI techniques have been used to identify these faults in the past to some success. These techniques include ANN and fuzzy logic. This project attempts to apply other recent machine learning techniques that showed better performance than ANN in several fields such as text categorization and image classification. SVM is one of the most successful AI techniques for pattern recognition applications proposed in recent times (Wang, 2005). This project applies the SVM method as a software tool for identification of faults in induction motors. SVM has advantages of handling large amount of data with several classes. It can effectively classify this data into classes using feature spaces. To classify the data, it uses nice mathematical techniques such as convex optimization that has the property to converge to a single global optimal solution. To the best of our knowledge, SVM technique has not been applied for identifying external faults in 3-phase induction motors in the past. This technique is expected to improve the fault identification methods for induction motors.

### *Limitation of the Study*

This study uses the induction motor faults data collected from a data acquisition system in a previous research project conducted at BGSU. This study is, therefore, limited to the BGSU ECET lab equipment used in the ANN study (Altman, 2004). This study also uses Libsvm software to implement SVM method, and it is limited to the capabilities of the Libsvm software program.

### *Definition of Terms*

**ANN:** Artificial Neural Network.

**SVM:** Support Vector Machines.

**Protective Relays:** Electromechanical or electronic devices used to trip the circuit breaker in case of faults diagnosis.

**Machine Learning:** Branch of artificial intelligence, which study the design of system's ability for learning from the data (Duda, Hart & Stork, 2001).

**Induction Motor Faults:** Electrical or mechanical faults that lead the motor to unnecessary consequences, such as increase the drawn current and rising the temperature.

**RMS Values:** Root mean square value.

**Kernel Function:** It is a function that has the ability of providing a bridge from linearity to non-linearity for algorithms, it can be expressed it terms of dot product (Konar & Chattopadhyay, 2009).

**Statistical Learning Theory:** It is a theory from statistical field deals with the problem of

finding a predictive function based on data (Wang, 2005).

**LabVIEW:** “Laboratory Virtual Instrument Engineering Workbench”, is a designed system platform developed by National Instrumentation for visual programming language.

**SCXI:** “Signal Conditioning Extension for Instrumentation”, a product from National Instrumentation used for interfacing the low level signals to signals can be send to DAQ (Data-Acquisition Board).

**VI:** “Virtual Instrument”, is LabVIEW subroutine contains a block diagram and a front panel.



## Chapter 2: Review of Literature

### *Introduction*

This chapter gives an overview of the existing literature of artificial intelligence methods and their application in fault identification. The first section describes different faults experienced by the 3-phase induction motors and their characteristics. It also discusses various protection schemes used for induction motors. A microprocessor based protective relay system is described in section 2. The third section describes the AI techniques which include the ANN and SVM. Finally, the fourth section discusses available literature on AI techniques application for fault identification in power systems and induction motors.

### *Faults in Induction Motors*

Three-phase induction motors are the most popularly used motors especially in industry because of their reliability and simplicity. These motors experience different types of faults during their operation (Elmore, 2004). These faults can be classified as internal and external faults. The external faults include:

1. Over loading
2. Single phasing
3. Unbalanced supply voltage
4. Locked rotor
5. Phase reversal

6. Ground fault
7. Under voltage
8. Over voltage

A brief description of these faults and their characteristics is given below. Protection of these motors is an important task which has been challenging to engineers. Protective relays were used to monitor these faults and disconnect the motor in case of a fault. Different relays used for protecting against these faults are also described below.

#### *Overloading*

Overload fault occurs when the mechanical torque exceeds the threshold point by applying mechanical load to the motor greater than full load rating. Overloading causes increase in phase currents, over heating the machine. In a traditional relay protection system, the over-current relay trips the motor off-line when the current transformers (CT) encounter over current in the line.

#### *Single phasing:*

Single phasing is one of the unbalanced cases of the motor. It occurs when one of the three lines are open. More current flows through the other two lines and more heat is generated in stator winding. In the traditional protection systems, a high-set instantaneous trip unit relay is used (Elmore, 2004). Single phasing also gives rise to negative sequence current. A negative sequence relay can also be used to protect against this fault.

### *Unbalanced supply voltage:*

There are many causes of unbalance supply voltages such as unbalance loading, open delta transformers and unequal tap setting. This condition leads to reduction in motor efficiency, raises the motor temperature and excessive unbalanced full load current (Sudha & Anbalagan, 2009). Three-phase voltages and currents during an unbalanced supply are shown in Figure 2.1. The protection design should detect over current condition during unbalanced supply (Elmore, 2004).

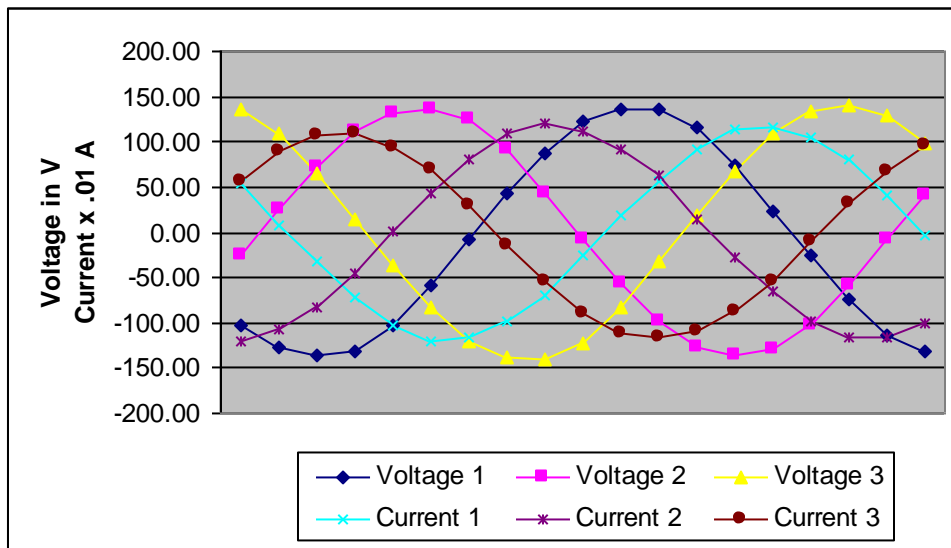


Figure 2.1. Three-phase Voltages and Currents for Induction Motor Under Unbalanced Supply Voltage.

### *Locked Rotor*

Locked rotor occurs when the voltage is applied to a non-rotating motor. The stator current may be almost six times its rated value during this condition (Sudha & Anbalagan, 2009). There are many causes for this fault to occur for instance, if the rotor shaft is connected to heavy

load the motor may experience locked rotor conditions. Locked rotor causes high current which leads to heating the rotor. Therefore locked rotor condition cannot be withstood for a long time. The allowed duration of the motor overloaded under locked rotor condition depends on the voltage applied to the motor terminals. Therefore, the protection system should be able to disconnect the motor when locked rotor condition exceeds the amount of allowed time. Traditional protective systems use over current relays with I-T characteristics (Elmore, 2004).

### *Phase Reversal*

Phase reversal occurs when any of the two phases are reversed from the normal sequence, which leads the motor to rotate in the opposite direction. When the motor starts to rotate in the opposite direction, it can cause intensive damage. Therefore, this condition should be corrected immediately. Reverse-phase relays and negative sequence relays are used for the protection (Elmore, 2004).

### *Ground Fault*

Ground faults occur when any of the phases touches the ground. Ground faults are more frequent in motors than any other power system, because of their violent condition and frequent starts. The effects of this fault are intensive such as causing hazards to human safety and interference with telecommunication. It can be detected by measuring the ground leakage current (Elmore, 2004).

### *Under Voltage*

Under voltage fault is reducing the supply voltage on the three phases by specific percentage, which makes the motor from attaining rated speed in specified time, increases the

current and overheats the machine. Low voltage protection relays are used in traditional systems. However, in order to avoid unwanted relay shutdowns due to momentary voltage drops, the AC contacts need a delay mechanism which delays the under voltage protection for a time period. This additional mechanism needs high sensitive devices and involves calibrations (Elmore, 2004).

### *Over Voltage*

Over voltage occur if the three phase voltages are greater than rated voltage. The effect of this fault is increasing current flow which leads unacceptable stress on the motor insulation due to high heat dissipation. Conventional protection systems use the over voltage relays to protection the motor during this condition (Elmore, 2004).

### *Power System Relays*

Protective relays are used to protect power system apparatus against faults such as induction motor faults described in the previous section (Elmore, 2004). The traditional protective relays based on electromechanical and solid state devices are mostly replaced by microprocessor based relays because of several advantages such as increased flexibility, improved reliability, faster response and economically viable (Sachdev, 1997).

A microprocessor based relay design scheme consists of several subsystems, such as analog processing, analog to digital conversion, and digital processor as shown in Figure. 2.2 (Sachdev, 1988). The starting point of this system is measuring the 3-phase voltages and currents

of the power system component such as induction motor. These quantities are first modified by current and voltage transformers and then processed in the analog pre-processing block. This block consists of auxiliary transformers, surge suppression circuit and low pass filter. The auxiliary transformers are used for reduction of the inputs to levels that are compatible with the digital equipment used. The surge suppression circuits protect the relay from voltage surges. Low pass filters are used to band limit the inputs to avoid aliasing. The cut-off frequency of the low pass filters is selected considering the sampling frequency and the predominant high frequency components expected in the inputs. The outputs of the analog preprocessor are digitalized using A/D converters. The digitalized values representing voltages and currents are processed in a digital processor that performs two main functions. The first function is to estimate the parameters of the fundamental frequency voltage and current phasors. The second function of the digital processor is to use the parameters of the calculated phasors to compute the required additional parameters and thereby decide if the motor is experiencing an abnormal operating condition. Recently AI techniques are used in this relay logic block. In this project the logic is implemented using SVM method to identify the faults in the induction motor. A brief description of AI techniques is given in the next section.

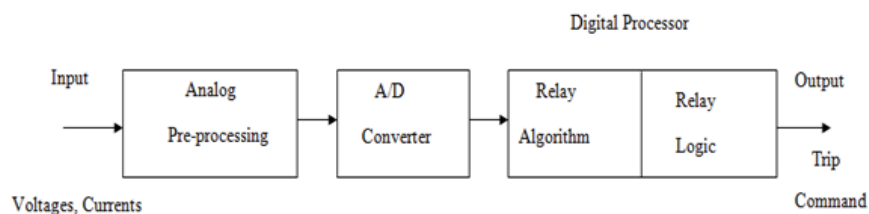


Figure 2.2. Block Diagram of a Microprocessor Based Relay Scheme.

### *Artificial Intelligence Techniques*

The concept of AI came from the notation of implementing the human's intelligence in a machine, so it can perform the same kind of functions that features human thinking processes. AI is composed of several branches such as expert systems (ES), genetic algorithms (GAs), fuzzy logic (FL), problem solving and planning (PSP), logic programming (LP), ANN, SVM, and more branches (Krishnamoorthy & Rajeev, 1996). These techniques have the ability to deal with nonlinear problems and to deal with large amount of data; once they are trained they can produce a model for the prediction of the new data. The AI techniques have been implemented in many disciplines such as engineering, economics, medicine, and military (Mellit & Kalogirou, 2008). This project uses SVM as an AI technique for classification and prediction of external faults experienced by three-phase induction motor. It also compares the performance of SVM with ANN for fault identification. In the following two sections, ANN and SVM techniques are described briefly from the existing literature.

#### *Artificial Neural Networks*

ANN is an intelligent system that emulates the biological neural networks into a mathematical model. It has been used by many researchers in different scientific fields to solve different kinds of problems such as pattern recognition, prediction, and optimization. ANNs are parallel computing systems consisting of a number of simple processors with many interconnections. Practically, this technique allows the network to learn rules for solving a problem by processing a set of examples. These simple processors are mutually interconnected with weights assigned to the connections. By modifying these weights according to learning rules, the ANN can be trained to recognize any pattern given by the training data. As per the

history, this technology started in 1940 due to McCulloch and Pitts pioneering work (Jain, Mao, & Mohiuddin, 1996). Furthermore, the real revolution of this technology occurred in early 1980s when it received considerable interest because of the Hopfield's energy approach (Jain, Mao, & Mohiuddin, 1996). Moreover, in 1982 the back-propagation learning algorithm for multilayer perceptrons (multilayer feed-forward networks) was proposed by Werbos (Mellit & Kalogirou, 2008). In the late 1980s, the research interest developed rapidly in neural network area.

The use of ANN for pattern recognition consists of two steps: first, the network is trained to learn the patterns using the training data; second, the trained network is used to test and monitor. In the ANN the neuron is the processor unit in the network, and these neurons are connected together in layers. Inputs are received at the first layer and they are processed in the neuron. These process steps include the multiplication of the inputs by their weights and adding them together, and then transfer them into the translation part of the neuron as shown in Figure 2.3.

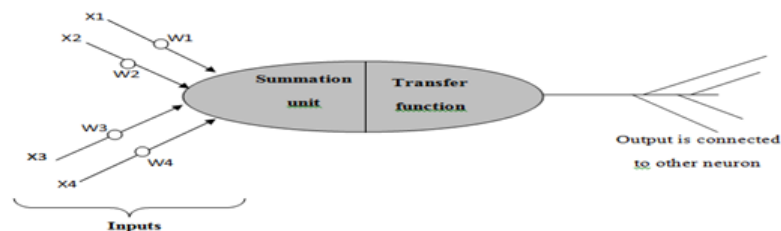


Figure 2.3. Single Neuron.

In the translation part, the incoming weighted signal will be sent through the transfer function. There are many transfer functions proposed such as tanh and sigmoid. After that the output from the input layer is transferred to hidden layers, one output from each neuron will



differ from other outputs by the link weights. Generally one hidden layer is used in a network as shown in Figure 2.4.

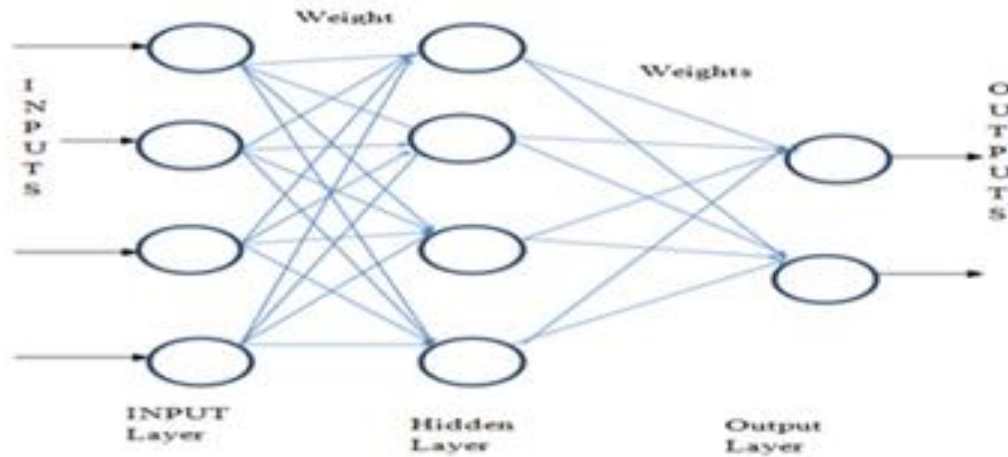


Figure 2.4. Feed Forward Neural Network.

Same processes are repeated till the signal is received at the output layer. In a previous study at BGSU (Kolla & Altman, 2007) ANN was introduced to identify external faults and no fault condition in a three-phase induction motor. A feed-forward network with back propagation training was used with one hidden layer. JavaNNS software program was used to create and train the network. Inputs were selected as RMS values of 3-phase voltages and currents, and they were collected from a 1/3 hp three-phase induction motor in real time. There were seven output neurons, corresponds to six faults and no fault condition as shown in Figure 2.5. If a particular condition exists the output goes to one otherwise it is zero. In this study, data from the same motor is used with a recent AI technique called SVM for identifying the faults. A comparison is made between the two techniques to study their performance and accuracy.

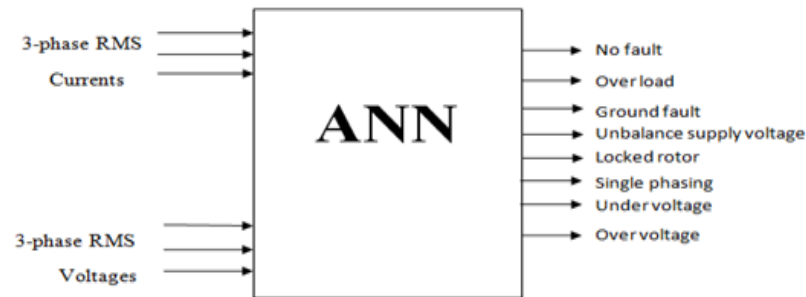


Figure 2.5. Inputs and Outputs of the ANN.

### *Support Vectors Machines*

SVM is a machine learning method for classification and regression based on statistical learning theory (Cristianini & Shawe-Taylor, 2000). The concept behind SVM is that the machine learns from the training data to classify the input data into different classes. If data is not linearly separable, it is mapped into a feature space where the training data is separable. The classifier is a hyperplane, so the most important points are the points which define the hyperplane, that are called support vectors as shown in Figure 2.6. The SVM method requires the solution of an optimization problem to find this hyperplane (Vapnik, 1995). A detailed mathematical development of SVM and the required optimization is presented in the methodology Chapter 3.

The properties of SVM make it a valuable method in classification, because it can handle very large feature spaces. The training is carried out so that the dimension of classified vectors does not have a special influence on the performance of SVM as it has on the performance of the traditional classifier (Widodo, Yang, & Han, 2007). Therefore, it is very efficient for large

classification problems especially in power systems, because the number of feature to be classified may be large.

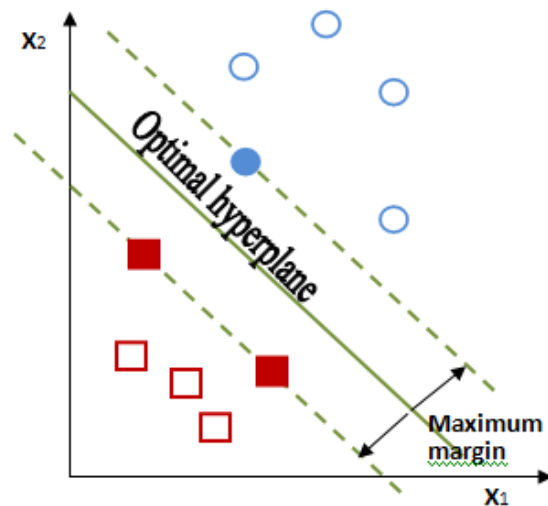


Figure 2.6. Classification of Two Classes Using SVM.

Applications of SVM for condition monitoring and fault identification were introduced in many areas. For example, they were reported in successfully classifying human cytochrome P450 enzyme (Kriegl, Arnold, Beck, & Fox, 2005), also they were used in financial analysis by (Min & Lee, 2005). Furthermore, SVM has been used for monitoring and identifying mechanical faults in power systems machinery; for example, ball bearing faults in induction motors (Jack & Nandi, 2002), bearing faults detection of induction motor by (Konar & Chattopadhyay, 2009). Also, Korkeakoulu and Hogskola (2004) used SVM based classification in condition monitoring of induction motors. An account of SVM application for power system fault identification from the existing literature is given in the next section.

### *Literature Review of AI Techniques for Faults Identification in Power Systems*

Power systems components include generators, transformers, transmission lines, and motors. These electrical components may experience several types of faults due to insulation failures and other conditions. Protection of the electrical components against these faults is an important aspect that has been developed since the power systems were established. As indicated in Section 2, relaying systems are used to protect power system components. Microprocessor based relays are the most popularly used technology (Frag & Kamel, 1999). Some of these relays use three-phase voltages and currents as inputs. Motor current signature analysis (MCSA) that analyzes high frequency components in currents to identify faults has also been used by many researchers (Benbouzid & Kliman, 2003). In this section some of the existing literature that uses AI techniques in the relay logic part of the microprocessor based relays are discussed. A detailed literature of the existing AI and other techniques for induction motor stator fault identification is given in (Siddique, Yadava & Singh, 2005).

Chow & Yee applied ANN to detect incipient faults in single-phase induction motors in the early 90's (Chow & Yee, 1991). They identified stator winding faults and bearing wear using motor current and speed as inputs. A fault identification scheme for three-phase induction motors using feed-forward ANN was proposed by Kolla and Altman (2007). They identified the external faults in the motor. There are several other applications of ANN for fault identification in power systems described by Kezunovic (1997). The problem with the traditional neural networks is difficulties with the generalization which produces models that can over-fit the data. SVM

technique is shown to alleviate some of these problems in traditional ANN.

SVM techniques have been applied for different types of faults identification in power systems (Ravikumar, Thukaram & Kincha, 2009). SVM is applied for internal fault identification in induction motors by (Widodo & Yang, 2007), (Poyhonen, et al., 2005) and (Nguyen & Lee, 2008). Also Fang and Mahave combined the MCSA and SVM techniques to identify induction motor faults (Fang & Ma, 2006). However, to the best of our knowledge there has not been much work done in applying SVM technique to identify external faults in three-phase induction motors. The aim of this project is to apply SVM method for identifying the three-phase induction motor external faults.

### *Summary*

In this chapter, a literature review was provided to complete this project successfully. The first section is an overview of three-phase induction motor and external faults that could be experienced during its operation. The second section covered power system relays for the protection of power systems apparatus. The artificial intelligence techniques including ANN and SVM are described in section three. The last section covered the literature review for the AI techniques for fault identification in power systems.

## Chapter: 3 Methodology

The methodology used to address the research problem is described in this chapter. The first two sections restate problem and objectives of this study. The next section describes the procedures used for this study which includes a detailed account of SVM technique. The Libsvm software used for implementing the SVM techniques is also described in that section. Section 4 gives the details of data collection to apply SVM technique for classification of induction motor faults. A summary of the chapter is given in Section 5.

### *Restatement of the problem*

This project develops a fault detection method for three-phase induction motors to detect external faults experienced by the motor using SVM technique. The developed SVM method is tested with real-time faults data from a 1/3 hp three-phase squirrel cage induction motor. Effect of different SVM parameters on the training and testing of the induction motor data is studied. This study also investigates the classification performance characteristics of the proposed SVM based method in comparison to ANN based method.

### *Restatement of the objective*

1. Study the usage of SVM method for pattern recognition applications and investigate the effect of different parameters on the training and testing of the method.
2. Understand the usage of Libsvm software program for pattern recognition applications.
3. Understand the characteristics of external faults on three-phase induction motors.

4. Apply the SVM method for identifying external faults in three-phase induction motors.
5. Study the performance of the proposed SVM based induction motor fault identification technique in comparison to ANN based method.

### *Procedure of the study*

As described in the objectives of this study, the starting point is the understanding of SVM as a classifier machine learning technique based on the statistical learning theory. A brief description of SVM from the literature is given below to explain various parameters used in the method. A systematic method to select these parameters to identify different faults using the Libsvm software forms the basis of the procedure for the study in this project.

### *Support Vectors Machines*

As explained in Chapter 2, the SVM method is based on mapping the input data into higher dimensional space and finding the optimal hyperplane that separates the input data with maximal margin if it is not linearly separable. The support vectors are those with nearest training patterns from the hyperplane (Richard et.al, 2001). The mapping of the data into higher dimensional space can be done by one of the four Kernel Functions. The linearly separable case is explained below first, and then the mapping using Kernel functions for nonlinear separable cases are explained.

The first case considered has the training data in only two classes and are linearly separated as shown in Figure 3.1. The training set is given by Equation (1) below:

$$(X_1, y_1) \dots (X_n, y_n), X_i \in \mathbb{R}^n \quad (1)$$

$$y_i \in \{-1, +1\}$$

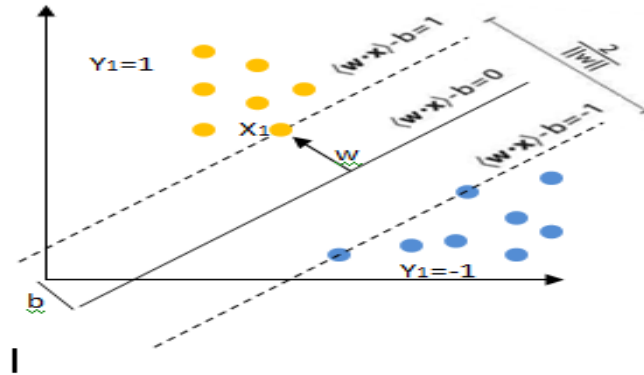


Figure 3.1. SVM Classification.

The separating hyperplane can be expressed by Equation (2) below:

$$W^T X_i + b = 0, \quad (2)$$

where  $W$  is the weight vector, which is perpendicular to the hyperplane, “ $b$ ” is the bias that expands the margin. The classified data are expressed by Equations (3) and (4)

$$W^T X_i + b \geq 1, \text{ if } y_{i=1} \quad (3)$$

$$W^T X_i + b \leq -1, \text{ if } y_{i=-1}. \quad (4)$$

Precision of the training data depends on the optimal separating hyperplane that separates the data by maximizing the margin  $2/\|w\|$ . This problem can be solved by quadratic



optimization method as formulated by Vapnik (1995) and stated below:

Minimize

$$\left(\frac{1}{2}\right) * ||w||^2 + C * \sum_{i=1}^n \xi_i$$

Subject to  $y_i(\mathbf{W}^T \mathbf{X}_i + \mathbf{b}) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad 1 \leq i \leq n,$

Where,  $\xi_i$  are the slack variables, which measure the miscalculation of the data  $\mathbf{X}_i$

$C$  is the error penalty constant.

After solving the above optimization problem, and applying the optimal solution to the  $\mathbf{W}$  and  $\mathbf{b}$ , the following decision function is obtained:

$$f(\mathbf{X}) = \text{sgn}(\mathbf{W}^T \mathbf{X}_i + \mathbf{b}) \quad (5)$$

The second case occurs when the trained data are beyond the boundary of the linear separation. Nonlinear classification has to be applied by mapping the data into feature space using Kernel function. There are many Kernel functions used such as

1. Linear:  $K(\mathbf{X}_i, \mathbf{X}_j) = \mathbf{X}_i^T \mathbf{X}_j$  (6)

2. Polynomial:  $K(\mathbf{X}_i, \mathbf{X}_j) = (\gamma \cdot \mathbf{X}_i^T \mathbf{X}_j + r)^d, \quad \gamma > 0$  (7)

3. Radial Basis Function:  $K(\mathbf{X}_i, \mathbf{X}_j) = \exp(-\gamma ||\mathbf{X}_i - \mathbf{X}_j||^2), \quad \gamma > 0.$  (8)

4. Sigmoid:  $K(\mathbf{X}_i, \mathbf{X}_j) = \tanh(\gamma \cdot \mathbf{X}_i^T \mathbf{X}_j + r),$  (9)

where  $r, d, \gamma$  are Kernel parameters.

The most flexible and popular Kernel function used is Radial Basis Function (Wang, 2005) given in Equation (8), above. In this project this function is primarily used for mapping the input vectors into feature space, though an attempt is made to study the performance of the other Kernel functions.

In general, SVM creates one hyperplane to separate the two classes. To classify multiple classes, “one-against-one” and “one-against-the rest” strategies are implemented (Wu, Liu & Wang, 2013). Solution methodology used for this optimization problem in case of the induction motor faults identification includes using a software program called Libsvm, which is described below.

#### *Libsvm Software to Implement SVM Technique*

Libsvm is combined software package for implementing SVM technique. Libsvm was developed by Chih-Chung Chang and Chih-Jen Lin in the Department of Computer Science in National Taiwan University (Chang & Lin, 2011). It was written in C++ programming language. Sequential Minimal Optimization (SMO) algorithm is used to perform the training of the SVMs (Wang, 2005). Moreover, its library contains an automatic model selection tool for C-SVM classification. It has the advantages of ease of use for implementing the SVM applications. In addition, there are many software packages introduced for SVM applications such as SVMLight(C), as well as a complete machine learning toolboxes which include Torch(C++), Spider (Matlab), and Weka, that are available through [www.Kernel-machines.org](http://www.Kernel-machines.org). In order to understand the usage of SVM method, the Libsvm package provides a simple graphical applet called svm-toy, a self-explanatory interface to show how SVM separates the data into classes. It

can separate up to three classes of input data. Libsvm uses one-against-one method to classify multiple classes (Chang& Lin, 2001). A detailed description of svm-toy applet is given below.

### *Svm-toy*

The svm-toy applet is started by using the command *Svm-toy.exe* in the MS-DOS window. The program allows the user to select points in different classes indicated by different colors as shown in Figure 3.2.

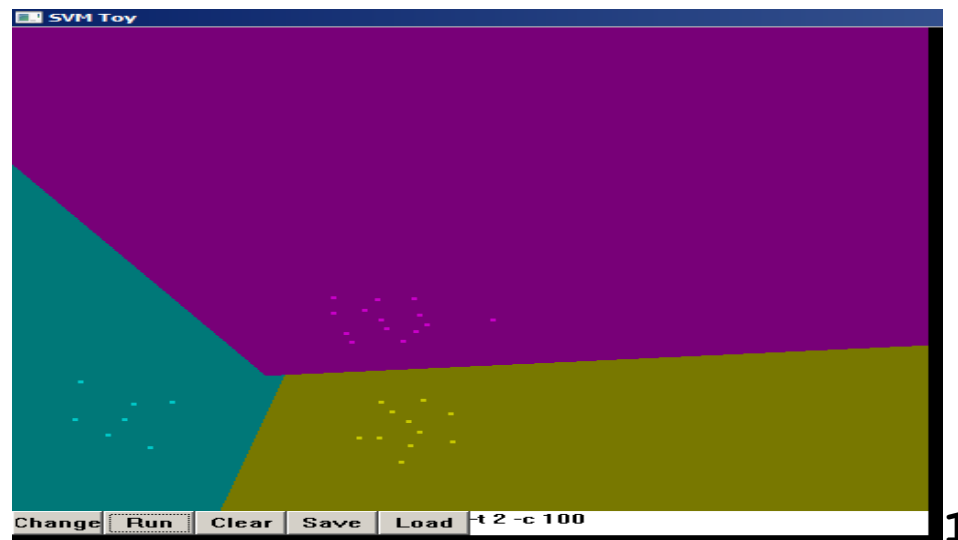


Figure 3.2. Svm-toy Screen Shot.

The selected points are separated into classes by the program after clicking the run button as shown in the Figure 3.2 with different color regions. The svm-toy applet allows selecting training parameters such as  $c$  and type of Kernel function as represented by different values in the screen shot in Figure 3.2. The applet also has other functionalities such as Save and Load.

### *Scaling and Training the SVM*

The format of the Libsvm data file starts with the label corresponding to 'y' and attributes corresponding to 'X'. In general the attribute space can be n-dimensional. Generally the Libsvm expects the data to be in the range of -1 to 1. In case if the data is not within this range, Libsvm provides a command to scale the values of the data in a particular range. The scaling command for the training data is shown below:

```
Svm-scale.exe -l -1 -u 1 -s range abc.txt > abc.scale
```

A new file (abc.scale) is created which has the training data in scaled form between -1, to 1. Scaling the testing data with the same range as training data can be achieved by recalling the saved file range. The command used for scaling the testing data is:

```
Svm-scale.exe -r range cba.txt >cba.scale.
```

The new created file cba.scale contains the scaled testing data in the range of -1 to1.

After the scaling is performed, the svm model is created through training using the training data. This training is performed using the following command:

```
Svm-train.exe abc.scale
```

The abc.scale in the above command contains the scaled training data. After executing that command in DOS prompt, an output model file is created (abc.scale.model). This file can be used to predict the classes for the testing data.

### *Testing the SVM*

Testing the data (prediction), is the process of classifying the output data, where the labels are assigned by the trained Libsvm model. Libsvm uses the following DOS command to test the data. The command requires the scaled test data file (cba.scale) and svm model file (abc.scale.model) as input; it gives predicted classes in the specified output file (p1p.out).

```
Svm-predict.exe cba.scale abc.scale.model p1p.out
```

If the classification accuracy is not satisfactory; Libsvm has tools to improve the prediction by adjusting Kernel function parameters. As mentioned before, RBF Kernel function is the most popularly used function. This function uses parameters  $\gamma$ , and  $C$  described in Equation (8). Libsvm provides in its tools library a command grid.py, which can give optimum values for  $\gamma$  and  $C$ , so that the accuracy can be improved. The grid.py command works in collaboration with gnuplot graphics software. After executing the following command:

```
Grid.py -gnuplot.exe C:\Users\Rama\Desktop\Libsvm-3.17\windows\bin\gnuplot abc.scale.
```

in the DOS prompt, the optimum values of  $C$  and  $\gamma$  can be found. The prediction accuracy will be increased if these values are used in creating the training model.

The graphical output that shows the decision boundaries from the gnuplot program is shown in Figure 3.3. The figure shows the values of  $\gamma$  and  $C$  for the chosen RBF as the optimum parameters are being determined by grid.py (Chang & Lin, 2011).

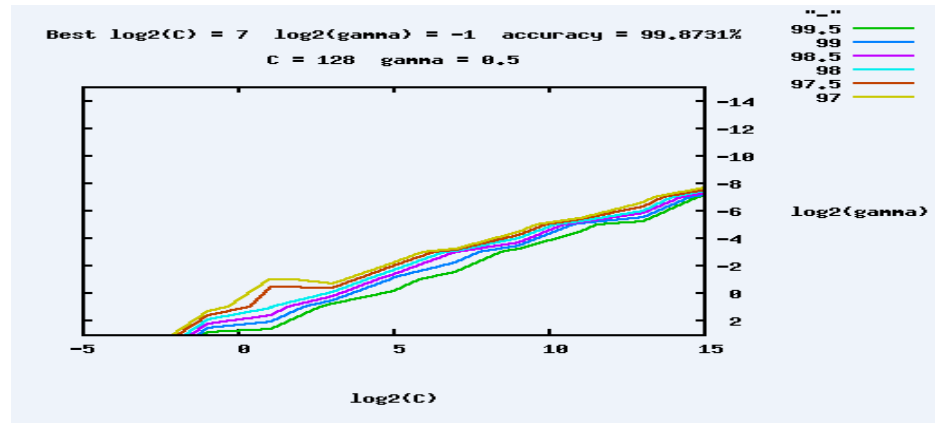


Figure 3.3. Grid Search for  $\gamma$  and  $C$  of the SVM Technique.

These optimum values are used to train the SVM with training data (abc.scale) using the following command:

```
Svm-train.exe -c value -g value abc.scale
```

The new model file is used to predict the classes in the testing data (cba.scale) using the following command, as given before.

```
Svm-predict.exe cba.scale abc.scale.model p2p.out
```

The new predicted output is given in file p2p.out.

### *Induction Motor Faults Data Collection and SVM Implementation*

The SVM technique described in previous section is applied to induction motor faults identification. Figure 3.4 shows the inputs and outputs of the SVM-based faults identification technique.

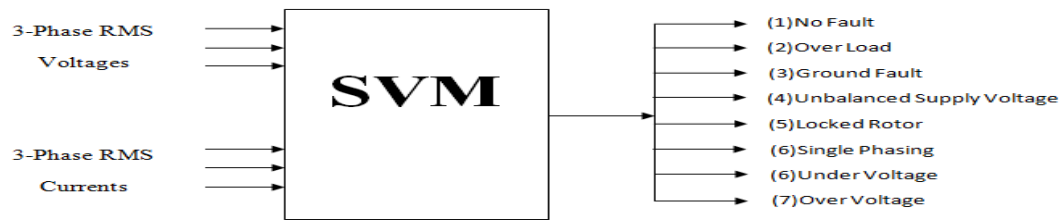


Figure 3.4. Inputs and Outputs of the SVM.

The Libsvm software described in the previous section is used. The data that is used in this project was collected in real time from 1/3 hp three-phase squirrel induction motor by Altman (2004), a graduate student at BGSU. The faults simulated were under voltage, over voltage, unbalanced voltage, single phasing, locked rotor, and overload. Three phase voltages and currents representing each of these faults are arranged in the required format for the Libsvm program. The format is:

<label1><index1>:<value1><index2>:<value2>.....

Label is an integer indicates the class number (y); Index is an integer starts from 1. Value is a real number representing the input (X). For example, the three-phase voltages and currents are shown below for few cases of faults. The complete data for all faults is given in Appendix A.

1 1: 2.6573950 2: 2.6461120 3: 2.6721640 4: 0.7466173 5: 0.7661240 6: 0.7626198  
 1 1: 2.6579960 2: 2.6453840 3: 2.6896300 4: 0.8143297 5: 0.7855652 6: 0.8147647  
 2 1: 2.6390520 2: 2.5931930 3: 2.6670100 4: 0.0062597 5: 0.6447509 6: 0.6415373  
 2 1: 2.6496810 2: 2.6034040 3: 2.6744000 4: 0.0060545 5: 0.6397968 6: 0.636463

### *Data Collection Procedure*

As explained before, this project uses the same induction motor faults data used in a previous study at BGSU for ANN method (Altman, 2004). This allows an appropriate comparison of the proposed SVM method with ANN method for faults identification. Faults were simulated in a real time on a 1/3 hp Hamden three-phase squirrel cage induction motor. As indicated before, the faults simulated were under voltage, over voltage, unbalanced voltage, single phasing, locked rotor, and overload. Three VARIACS shown in Figure 3.5 were used to implement under voltage, over voltage, and unbalanced voltage faults.



Figure 3.5. VARIAC.

Overload and locked rotor fault conditions were simulated by a prony brake shown in Figure 3.6. Single phasing fault was created by disconnecting a phase power line. These simulated faults data were collected in real-time through a LabVIEW™ program, and the data for the voltage and current waveforms was stored in a text file. The data acquisition system that collects the voltage and current samples is shown in Figure 3.7. This data was used in this project to train and test the SVM.

The data acquisition system reduced the three-phase voltages and converted three-phase



currents into its proportional voltages. The conversion range was  $\pm 10$  volts, the scaling factor for voltages was 41.283 V/V, and the transfer function for currents to voltages conversion was 2.3741 A/V. A voltage transducer was used to step the voltage down. A second order active low-pass Butterworth filter was used so that only the signal up to 60 Hz was passed to the DAQ board on the SCXI 1000 chassis shown in figure 3.7. The SCXI chassis contains simultaneous sampling analogue module 1140 to collect the voltages and currents, and eight-channel power relay output module 1161 to simulate circuit breakers. The chassis communicated with multi-function I/O card PCI-MIO-16E-1 placed in the PC.



Figure 3.6. Prony Brake.

A LabVIEW<sup>TM</sup> program was written in the previous work to simultaneously acquire the signal conditioned instantaneous motor voltages and currents data values. The program acquired data at a sampling rate of 1000 scans/sec. The program then converted these data values into

RMS values. These signals were used to train and test the faults identification system. The number of signals collected for each fault is listed in Table 3.1. There were also 21 other signals representing various faults collected and used for testing.

Condition	Number of Patterns
No Fault (NF)	154
Single Phasing (SP)	85
Unbalanced Voltage (UB)	450
Under Voltage (UV)	49
Over Voltage (OV)	10
Locked Rotor (LR)	10
Overload (OL)	30
Total	788

Table 3.1. Number of Training Patterns Per Fault.

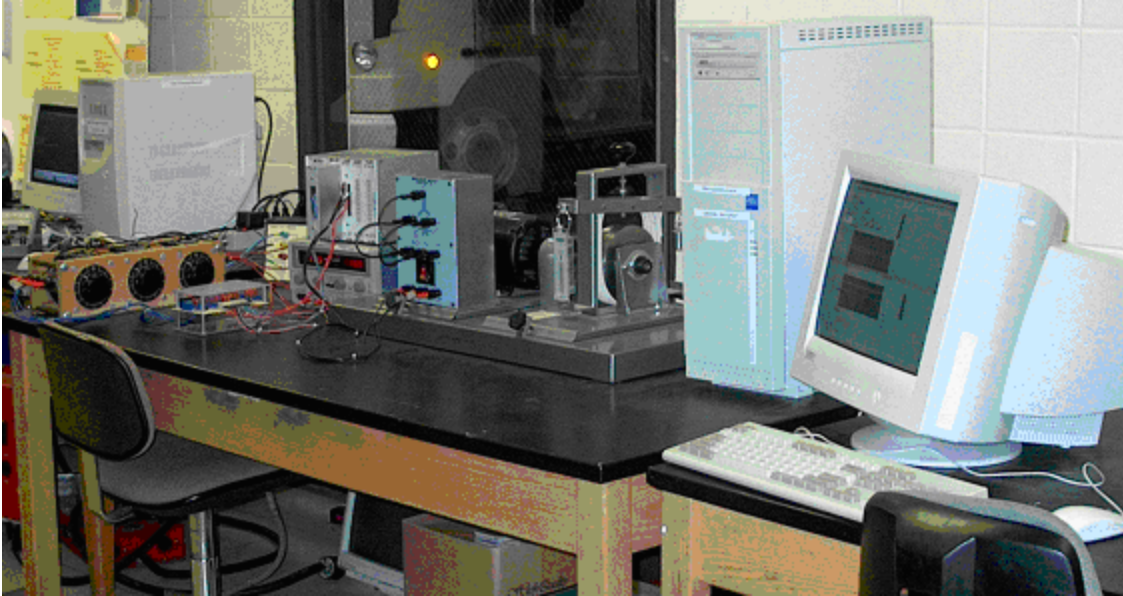


Figure.3.7. Complete System.

In addition the data in Table 3.1 is divided into two parts with some samples used to train the SVM and the remaining data used for the testing part. Different Kernel functions are used and the performance of these functions is studied. The Kernel function parameters are varied as explained in the previous section. Once the data is trained the trained model is used to predict the faults. The results are compared with other existing techniques such as ANN. These results are presented in the next chapter.

### *Summary*

This chapter provided a detailed description of the procedure used to achieve the objectives of the study. The restated statement and the objectives were followed by the description of the SVM method. The training and testing procedure using Libsvm software was also described. Finally the data collection procedure from the three-phase induction motor was also presented.

## Chapter 4: Findings

This chapter provides the results from testing the SVM technique with real-time acquired induction motor testing data. In this chapter, comparison of SVM technique performance with ANN technique is also described. The first section provides information from the training of the 3-phase induction motor data and the testing with a different 21 samples that are not part of the training. The second section provides information from the training of 80% of the total training data from previous section for training and using the 20% for testing. The third section provides information from the training of 90% of the data and using the 10% for testing. The fourth section uses 90% training data as a model and 21 samples that are not part of the training used for testing. The fifth section gives the studies of the performance of different Kernel functions for training and testing the 3-phase induction motor data. The last section describes the comparison of the performance of SVM and ANN techniques for faults identification.

### *Training and testing with the SVM*

The training of the SVM was done using Radial Bases Function as the Kernel. The training was done by using the 788 patterns of the 3-Phase induction motor data given in Appendix A. The testing was done by using the 21 testing samples that were not part of the training. The optimization of the Kernel parameters was achieved using grid search of the Libsvm. Different scaling factors for the data were also used in the training process. The first case considered was applying RBF Kernel with a scaling range between (-1, 1). After searching for the best parameters values for  $\gamma$  and C using grid.py, the optimum values obtained were

( $C=128$ ,  $\gamma =0.5$ ). The search results are shown in Figure 4.1. The classification accuracy obtained for these parameter values was 90.47% (19/21 samples) as shown in Figure 4.2.

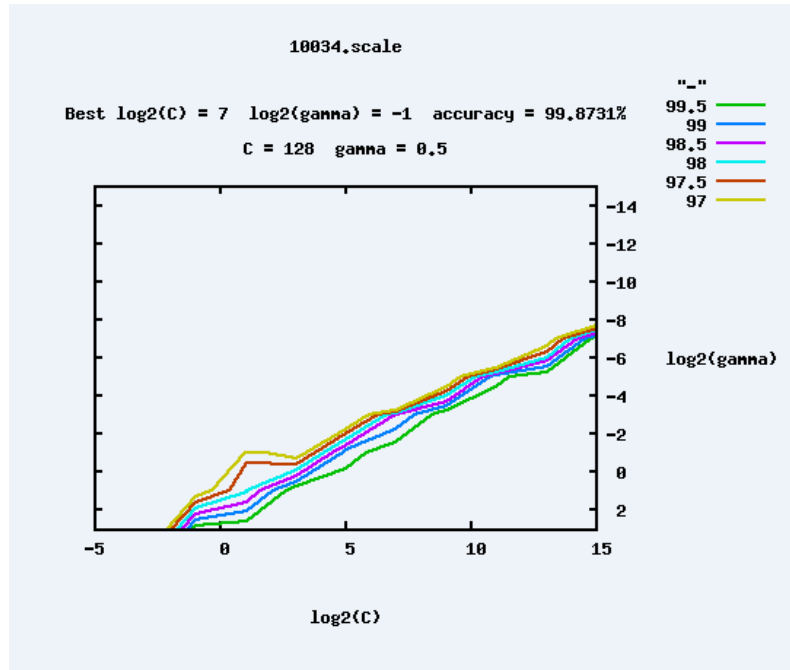


Figure 4.1. Gnuplot Output of Optimum C and  $\gamma$  for (-1,1) Scaling.

```
C:\Users\Rama\Desktop\libsvm-3.17\windows>svm-predict.exe 21rbf.scale 100rbf.scale.model 48.out
Accuracy = 90.4762% (19/21) (classification)
C:\Users\Rama\Desktop\libsvm-3.17\windows>
```

Figure 4.2. Screen-shot of Prediction Command for 21 Test Cases Scaling (-1, 1).

Since the classification accuracy is not 100% with this scaling, a second case was considered using the same Kernel function with a different scaling range between (-1, 0). For this case, the optimum Kernel parameters were found to be ( $C=512$ ,  $\gamma =0.5$ ). The search results are shown in

Figure 4.3. The classification accuracy (testing) was found to be 100% for the 21 samples. The screen-shot of the execution of Libsvm for this case is shown in Figure 4.4.

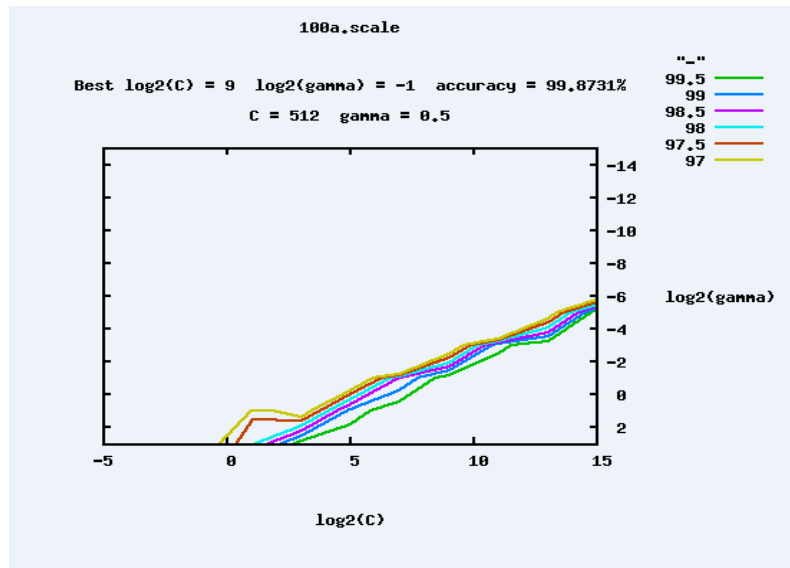


Figure 4.3. Gnuplot Output of Optimum C and  $\gamma$  for (-1,0) Scaling.

```
*
optimization finished, #iter = 3
nu = 0.000622
obj = -6.364239, rho = 0.000000
nSV = 2, nBSV = 0
Total nSV = 110

C:\Users\Rama\Desktop\libsvm-3.17\windows>svm-predict.exe 18.scale 318.scale.model 89.out
Accuracy = 100% (21/21) (classification)

C:\Users\Rama\Desktop\libsvm-3.17\windows>
```

Figure 4.4. Screen-shot of Prediction Command for 21 Test Cases.

This testing data consisted of the voltages and currents obtained during simulated fault conditions on 1/3 hp motor in real time. The input voltage and current waveforms for one of the sets used to test the SVM are shown in Figures 4.5 through 4.11. Figure 4.5 shows the voltage and current waveforms from a no fault condition. Figure 4.6 shows the waveforms from a single

phasing fault case. Figure 4.7 shows the waveforms from an unbalanced voltage fault. Figure 4.8 shows the waveforms from an under voltage fault. Figure 4.9 shows the waveforms from an over voltage fault. Figure 4.10 shows the waveforms from a locked rotor fault. Figure 4.11 shows the waveforms from an overload fault. RMS values of voltages and currents for these cases were calculated and reported in Table 4.1. The third row for each of the fault cases in the table corresponds to these seven voltages and currents waveform figures. The table also shows another two sets of data that was used as a part of the data to test the performance of the SVM. As it can be observed from the table, the SVM method has correctly identified all the seven types of conditions for these three sets of data.

Faults	Inputs						Outputs
	V1	V2	V3	I1	I2	I3	
No Fault (1)	2.650	2.640	2.696	0.428	0.428	0.431	1
	2.694	2.652	2.690	0.517	0.521	0.502	1
	2.622	2.614	2.666	0.410	0.418	0.417	1
Single Phasing (2)	2.688	2.609	2.721	0.625	0.629	0.013	2
	2.694	2.642	2.703	0.644	0.648	0.004	2
	2.616	2.600	2.642	0.628	0.632	0.004	2
Unbalanced Voltage (3)	2.624	1.906	2.022	0.538	0.249	0.332	3
	1.682	2.475	2.702	0.186	0.634	0.628	3
	1.624	2.019	2.585	0.207	0.449	0.576	3
Under Voltage (4)	1.088	1.087	1.084	0.231	0.239	0.225	4
	1.966	1.969	1.977	0.305	0.318	0.300	4
	1.976	1.967	1.995	0.306	0.312	0.307	4
Over Voltage (5)	2.888	2.878	2.863	0.482	0.499	0.496	5
	2.856	2.839	2.871	0.478	0.477	0.467	5
	2.882	2.872	2.868	0.488	0.484	0.459	5
Locked Rotor (6)	2.657	2.613	2.687	1.671	1.651	1.669	6
	2.649	2.614	2.647	3.090	3.096	3.043	6
	2.573	2.565	2.609	3.052	3.073	2.996	6
Overload (7)	2.638	2.602	2.675	0.807	0.788	0.800	7
	2.681	2.639	2.679	0.831	0.833	0.820	7
	2.604	2.600	2.643	0.882	0.893	0.884	7

Table 4.1. Testing the SVM.

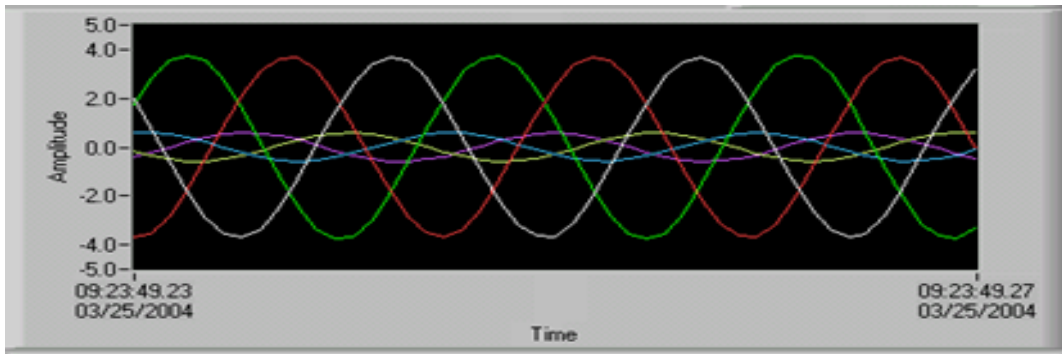


Figure 4.5. No Fault Condition.

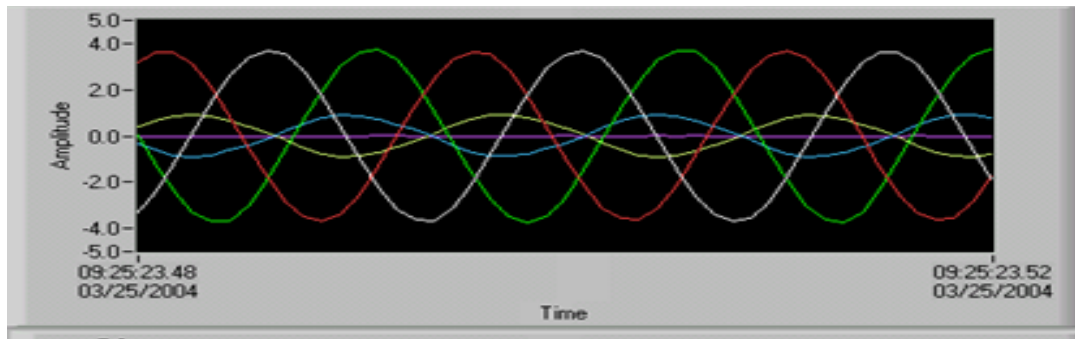


Figure 4.6. Single Phasing Case.

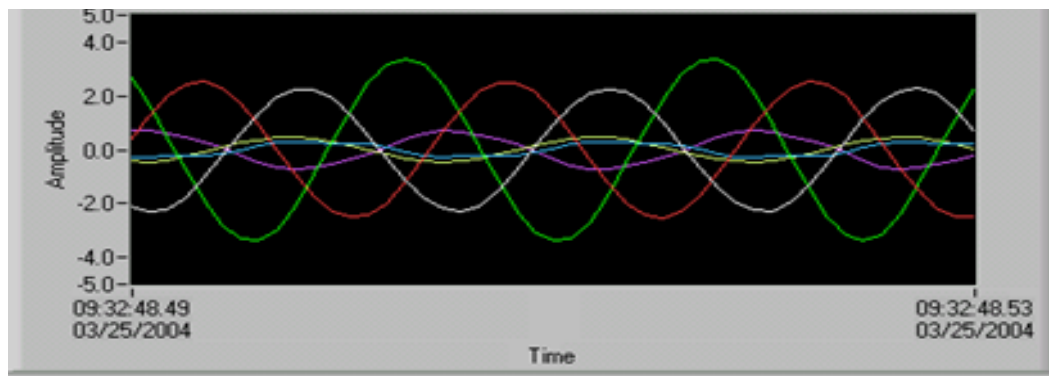


Figure 4.7. Unbalanced Voltage Fault.





Figure 4.8. Under Voltage Fault.

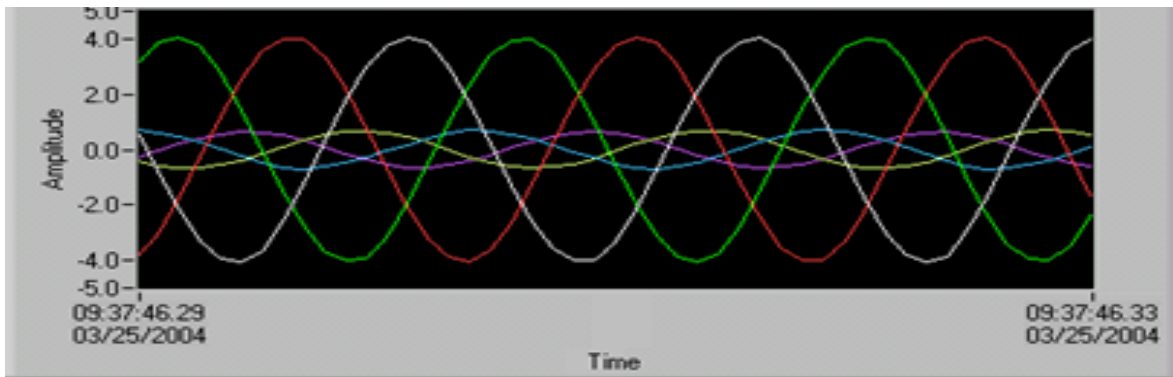


Figure 4.9. Over Voltage Fault.

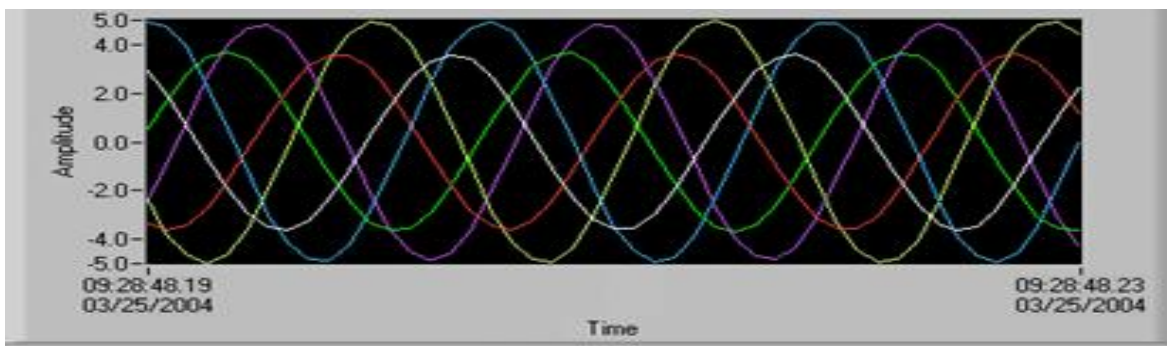


Figure 4.10. Locked Rotor Fault.

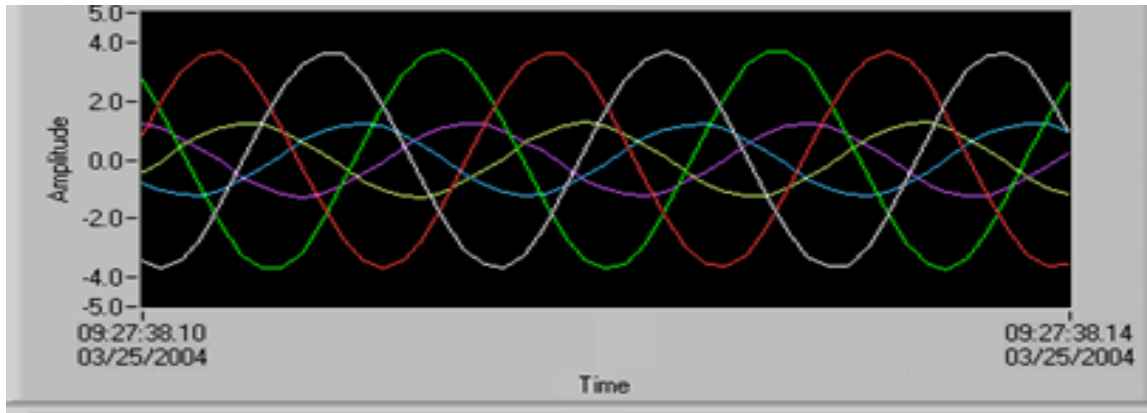


Figure 4.11. Overload Fault.

### *Training and Testing of the Split Data*

Training data of 80% samples and 20% testing data:

In this section testing is performed by using 80% of the training data from the previous section as the training model data and the remaining 20% of the data is used for testing. The 80% samples represent 632 samples of the total 788 samples. The testing samples represent 156 cases. These 156 samples correspond to fault cases shown in Table 4.2:

Condition	Number of Patterns
No Fault (NF)	30
Single Phasing	17
Unbalanced Voltage	90
Under Voltage	9
Over Voltage	2
Locked Rotor	2
Over Load	6

Table 4.2. Testing Samples.

The same RBF Kernel is used for this study also. The scaling of the data was done between (-1, 1). For this case, the optimum Kernel parameters were found to be ( $C=32$ ,  $\gamma =0.5$ ). The search results are shown in Figure 4.12.

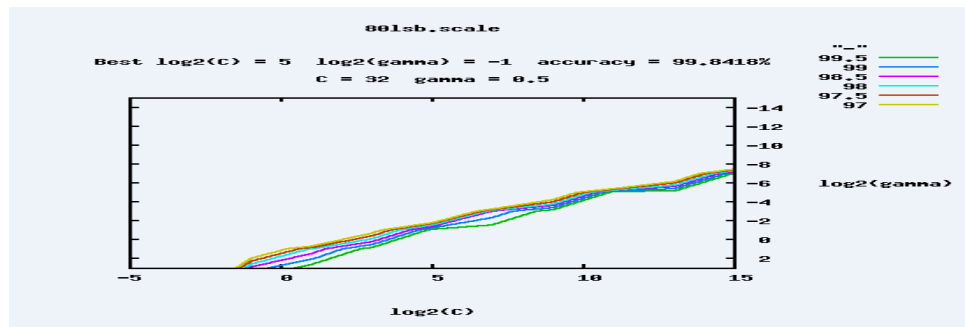


Figure 4.12. Gnuplot Output of Optimum  $C$  and  $\gamma$  for (-1,1) Scaling.

The classification accuracy (testing) was found to be 100% for the 156 samples. The screenshot of the execution of Libsvm for this case is shown in Figure 4.13.

```

Administrator: C:\windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Rama>cd C:\Users\Rama\Desktop\libsvm-3.17\windows
C:\Users\Rama\Desktop\libsvm-3.17\windows>svm-predict.exe 201sb.scale 801sb.scale.model 987.out
Accuracy = 100% (156/156) (classification)
C:\Users\Rama\Desktop\libsvm-3.17\windows>

```

Figure 4.13. Screen-shot of Prediction Command for 20% Test Cases.

The scaling of the data was also done between (-1, 0) and tested to study the effect of the different scaling for this case. In this case, the optimum Kernel parameters were found to be ( $C=32$ ,  $\gamma =0.5$ ). The search results are shown in Figure 4.14.

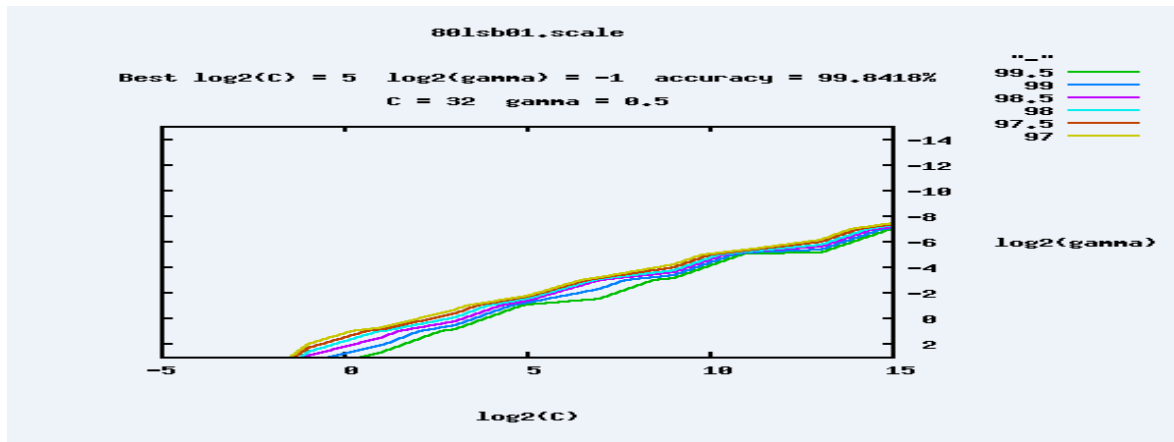


Figure 4.14. Gnuplot Output of Optimum C and  $\gamma$  for (-1,0) Scaling.

The classification accuracy (testing) was found to be 100% for the 156 samples. The screen-shot of the execution of Libsvm for this case is shown in Figure 4.15.

```

Administrator: C:\windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Rama>cd C:\Users\Rama\Desktop\libsvm-3.17\windows
C:\Users\Rama\Desktop\libsvm-3.17\windows>svm-predict.exe 201sb01.scale 801sb01.
scale.model 969.out
Accuracy = 100% (156/156) <classification>
C:\Users\Rama\Desktop\libsvm-3.17\windows>_

```

Figure 4.15. Screen-shot of Prediction Command for 20% Test Cases.

It can be observed that though scaling made a difference in the testing results of previous section, there was no effect of scaling for this case which was found to be 100% accuracy.

Training data of 90% samples and 10% testing data:

Another aspect of study was testing by using 90% of the training data from the previous

section as the training model data and the remaining 10% of the data is used for testing. The 90% samples represent 710 samples of the total 788 samples. The testing samples represent 78 cases. These 78 samples correspond to fault cases shown in Table 4.3:

Condition	Number of Patterns
No Fault (NF)	15
Single Phasing	8
Unbalanced Voltage	45
Under Voltage	5
Over Voltage	1
Locked Rotor	1
Over Load	3

Table 4.3. Testing Samples.

The RBF Kernel was used for this study also. The scaling of the data was done between (-1, 1).

For this case, the optimum Kernel parameters were found to be ( $C=32$ ,  $\gamma =0.5$ ). The search results are shown in Figure 4.16.

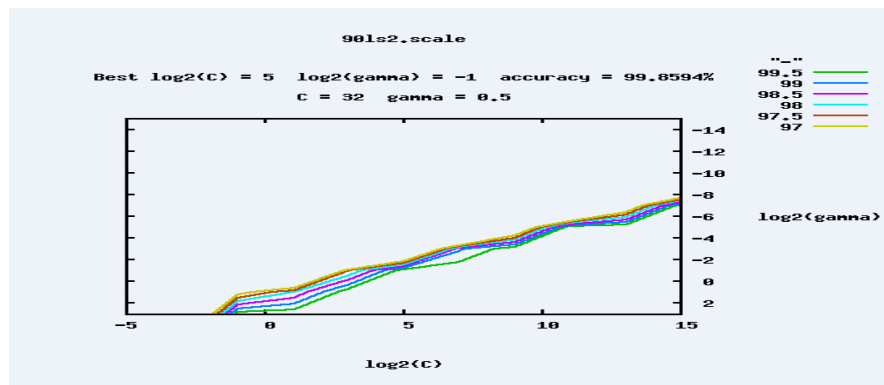


Figure 4.16. Gnuplot Output of Optimum C and  $\gamma$  for (-1,1) Scaling.

The classification accuracy (testing) was found to be 100% for the 78 samples. The screen-shot

of the execution of Libsvm for this case that gives 78 out of 78 classifications is shown in Figure 4.17.

```

Administrator: C:\windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Rama>cd C:\Users\Rama\Desktop\libsvm-3.17\windows
C:\Users\Rama\Desktop\libsvm-3.17\windows>svm-predict.exe 10ls2.scale 90ls2.scale.model 369.out
Accuracy = 100% (78/78) (classification)
C:\Users\Rama\Desktop\libsvm-3.17\windows>_

```

Figure 4.17. Screen-shot of Prediction Command for 10% Test Cases.

The scaling of the data was also done between  $(-1, 0)$  and tested to study the effect of the different scaling. For this case, the optimum Kernel parameters were found to be  $(C=512, \gamma=0.5)$ . The search results are shown in Figure 4.18.

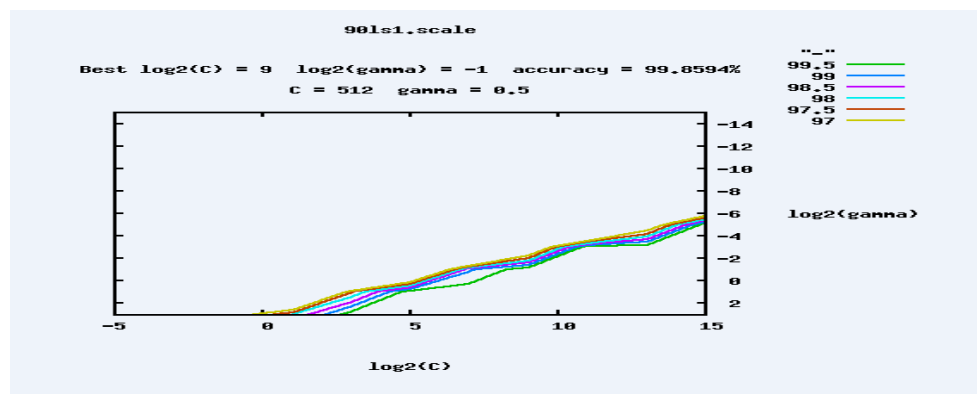
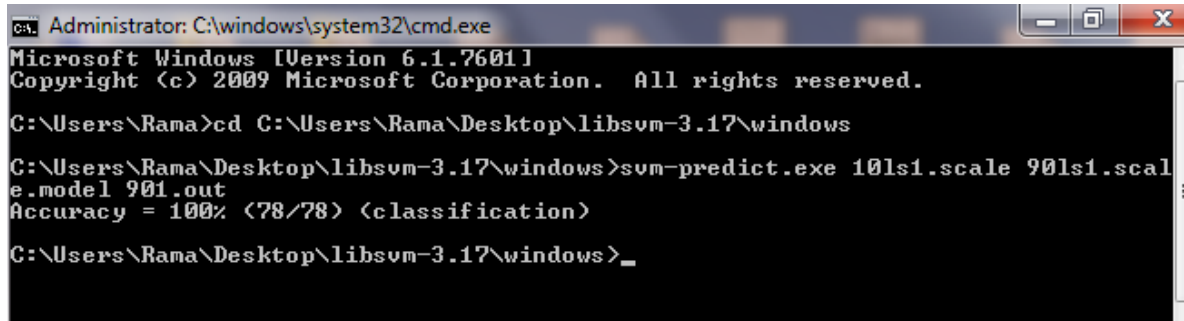


Figure 4.18. Gnuplot Output of Optimum C and  $\gamma$  for  $(-1,0)$  Scaling.

The classification accuracy (testing) was found to be 100% for the 78 samples. The screen-shot of the execution of Libsvm for this case is shown in Figure 4.19.



```
Administrator: C:\windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

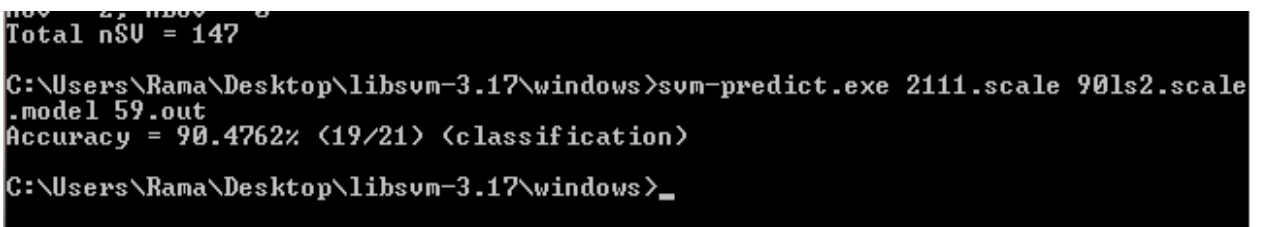
C:\Users\Rama>cd C:\Users\Rama\Desktop\libsvm-3.17\windows
C:\Users\Rama\Desktop\libsvm-3.17\windows>svm-predict.exe 10ls1.scale 90ls1.scale.model 901.out
Accuracy = 100% (78/78) (classification)
C:\Users\Rama\Desktop\libsvm-3.17\windows>_
```

Figure 4.19. Screen-shot of Prediction Command for 10% Test Cases.

It can be observed that though scaling made a difference in the testing results of previous section, there was no effect of scaling for this case also, giving the testing accuracy to be 100%.

Testing of 21 samples with 90% training model data:

Another aspect of the study conducted was testing the 21 samples from earlier section with 90% of training data from this section as a model. The first case considered was with the model of scaling between  $(-1, 1)$  and  $\gamma$  and  $C$  parameter values same as before. The classification accuracy (testing) was found to be 90.476% (19 out of 21 samples identified correctly). The screen-shot of the execution of Libsvm for this case is shown in Figure 4.20.



```
Administrator: C:\windows\system32\cmd.exe
Total nSV = 147
C:\Users\Rama\Desktop\libsvm-3.17\windows>svm-predict.exe 2111.scale 90ls2.scale.model 59.out
Accuracy = 90.4762% (19/21) (classification)
C:\Users\Rama\Desktop\libsvm-3.17\windows>_
```

Figure 4.20. Screen-shot of Prediction Command for 21 Samples Test Cases.

To improve the classification accuracy for the 21 samples case, the scaling of the data was also done between  $(-1, 0)$  with the same  $\gamma$  and  $C$  values as before. The classification accuracy (testing) was found to be 100%, (21 out of 21 samples classified correctly). The screen-shot of the execution of Libsvm for this case is shown in Figure 4.20. As before, the scaling had an effect on the classification accuracy for this 21 samples case also.

```
obj = -6.364239, rho = 0.000000
nSV = 2, nBSV = 0
Total nSV = 106

C:\Users\Rama\Desktop\libsvm-3.17\windows>svm-predict.exe 2101.scale 90ls1.scale
.model 36.out
Accuracy = 100% (21/21) (classification)

C:\Users\Rama\Desktop\libsvm-3.17\windows>_
```

Figure 4.21. Screen-shot of Prediction Command for 21 Samples Test Cases.

### *Effect of Different Kernel Functions*

This section gives results of the study of the effect of different Kernel functions on the classification accuracy. The Kernel functions considered were Linear, Polynomial, RBF and Sigmoid. The effect of different scaling factors and the grid search usage on these different Kernel functions is also considered in this study. Table 4.4 presents different Kernel Function classification rates with different scaling factors and grid search modifications for  $C$  and  $\gamma$  values. As it can be observed, the best results were achieved by Polynomial Kernel Function when the scaling is between  $(-1, 1)$ , which gave the testing classification of 100%. Also the RBF with the scaling between  $(-1, 0)$  gave the classification accuracy of 100% as discussed in the



previous section.

<b>Kernel Function Type</b>	<b>Scaling Factor.</b>	<b>Grid Search Effect</b>	<b>The Classification Accuracy</b>
Linear Kernel Function T=0.	(-1, 0)	No mod.	33.3%
		Mod. C=512, $\gamma =0.5$	71.42%
	(-1, 1)	No mod.	57.14%
		Mod. C=128, $\gamma =0.5$	71.42%
Polynomial Kernel Function. T=1.	(-1, 0)	No mod	28.57%
		Mod. C=512, $\gamma =0.5$	95.23%
	(-1, 1)	No mod	33.33%
		Mod. C=128, $\gamma =0.5$	100%
Radial Bases Function. T=2.	(-1, 0)	No mod	33.33%
		Mod. C=512, $\gamma =0.5$	100%
	(-1, 1)	No mod	33.33%
		Mod. C=128, $\gamma =0.5$	90.47%
Sigmoid Kernel Function T=3.	(-1, 0)	No mod	19.04%
		Mod. C=512, $\gamma =0.5$	28.57%
	(-1, 1)	No mod	33.3%
		Mod. C=128, $\gamma =0.5$	38.095%

Table 4.4. Kernel Function Verification.

#### *SVM versus ANN classification Performance*

The faults classification performance of SVM and ANN methods was studied by comparing the training times (model creation time) and prediction accuracy of them.

The model creation and training phase with ANN involves selecting number of neurons in the input and output layers (Jack & Nandi, 2002), to match the input fault data and output number of faults to identify. The number of hidden layers and the number of neurons in each layer is a trial and error process requiring significant time and effort to come up with an

appropriate model for ANN. For the same induction motor data, the previous study (Altman, 2004) came up with one hidden layer that has eight neurons after a significant trial and error method. The training of the selected ANN to obtain the weights using back propagation algorithm in the previous study involved choosing the parameters  $\alpha$  (Momentum coefficient), and  $\eta$  (Learning Coefficient), which is also done by trial and error method. During the training, the Sum of Squared Error (SSE) that represents the effectiveness of training was only reduced to 0.013284 after 160,000 training iterations.

In SVM method, model creation by obtaining support vectors uses a well-developed software tool in Libsvm that does not require trial and error method like selecting number of layers and neurons in the ANN method. To select  $\gamma$  and C parameters in the SVM method, Libsvm has a well-developed software tool using the grid search, unlike the trial and error ( $\alpha$ ) and ( $\eta$ ) parameter search in ANN. Thus the model creation phase with SVM is less time consuming than the ANN method.

Testing performance of the SVM is also better than ANN using the same set of induction motor faults data. The ANN method used in previous study by Altman (2004) required the output to be considered either 1 or 0 within 5% of their values in identifying the faults. In the SVM method there is no need to use any rounding for the results as the faults are classified into one or other type.

### *Summary*

This chapter provided the results obtained from testing the SVM method. It started with training the SVM with a set of 788 data samples and testing the SVM with a data of 21 samples that are not part of the training data. The second section gave results of training and testing the SVM method using 80% and 90% of 788 samples for training and rest of the samples for testing. The third section provided the studies of different Kernel functions and their effect on the testing accuracy. Finally the results of comparison between SVM and ANN methods were given. The results obtained from this chapter confirmed the efficacy of SVM based method for identifying faults in three-phase induction motor.

## Chapter 5: Conclusion

In this chapter, a summary for the work that has been done in this project for achieving the objectives of the study is provided. Also conclusions from this work and suggestions for future work are presented.

### *Summary and Conclusions*

The objective of this study is to develop a SVM based induction motor external faults identifier. This faults identifier detects six types of external faults experienced by the three-phase induction motor and also no fault condition. These faults are single phasing, unbalanced supply voltage, under voltage, over voltage, locked rotor, and over Load. The real time collected data from a 1/3 hp, 208 V three-phase squirrel cage induction motor is used to train and test the SVM. Libsvm software program is used for implementing the SVM method. In addition, the Libsvm tools library is used for searching the optimum parameters for achieving the adequate accuracy.

A RBF Kernel is used in this project for creating the trained model; also different Kernel functions are introduced in this project to study their effectiveness on the testing accuracy. It is observed that the scaling range has an impact on the prediction accuracy for some Kernel functions in the SVM based induction motor faults identification method considered in this study. One of the testing cases considered used a part of the data for creating the model and the remaining part for testing the accuracy of the prediction. Finally, a comparison between SVM based method and ANN method is discussed. The findings of this project attained the goals of this study: developing a SVM based induction motor external faults identifier, and achieving the better performance of the SVM based method than ANN based method (Altman, 2004).

To the best of our knowledge, the SVM technique is not applied for external fault identification of three-phase induction motor before this study, though it has been used for other types of faults in induction motors (Poyhonen, et al., 2005). SVM has the advantages of simplicity of creating the model and testing its accuracy. The time consumed for the model creation phase with SVM method is less than the ANN method, and SVM method accuracy is better than ANN based method.

#### *Recommendation for Future Work*

1. The first recommendation for future work would be to build the real-time implementation system for the three-phase induction motor faults identification based on the proposed SVM method in this project. Testing the SVM classifications of the faults in a real-time with several fault conditions should also be performed.
2. Another suggestion would be to apply SVM based method for identifying faults in different power systems components such as synchronous generators and transformers.
3. The proposed scheme in this project uses the RMS values of the voltages and currents as input signals to train and test the SVM. The recommendation for future is to use different signal such as instantaneous values of voltages and currents, and magnitude and phase angles of voltages and currents.
4. Another suggestion would be to study the effect of simultaneous faults occurrence on the performance of the SVM based faults identifying method.

## Reference

- Altman, S. D. (2004). *Real-Time Implementation of a Fault Identification Scheme for a Three-Phase Induction Motor Using Artificial Neural Network* (Unpublished master's thesis). Bowling Green State University, Bowling Green, OH.
- Benbouzid, M. E., & Kliman, G. B. (2003). What stator current processing-based techniques to use for induction motor rotor faults diagnosis. *IEEE Transactions on Energy Conversion*, 18, 238-244.
- Brun, Y., & Ernst, M. D. (2004). Finding latent code errors via machine learning over program executions. *IEEE Computer Society*. Retrieved from <http://homes.cs.washington.edu/~mernst/pubs/machlearn-errors-icse2004.pdf>
- Ravikumar, B., Thukaram, D., & Khincha, H. P. (2009). An approach using support vector machines for distance relay coordination in transmission system, *IEEE Transactions on Power Delivery*, vol. 24, 79-88.
- Cristianini, N., & Shawe, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. UK: University Press Cambridge.
- Chow, M.-Y., & Yee, S.-O. (1991). Using neural networks to detect incipient faults in induction motors. *Journal of Neural Network Computing*, vol. 2, 26-32.
- Chih-Chung, C. & Chih-Jen, L. (2011). LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1--27:27. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>

Cortes, C., & Vapnik, V. N. (1995). Support-Vector Networks. Retrieved from Machine Learning website: <http://www.springerlink.com/content/k238jx04hm87j80g/>

Duda, R. O., Hart, P. E., & Stork, D. G. (2001) *Pattern Classification*. John Wiley & Sons, Inc.

Elmore, W. A. (2004). *Protective Relaying Theory and Applications*. New York, NY: Marcel Dekker, Inc.

Engin, A. , & Derya, A. (2009). Using Combination of Support Vector Machines for Automatic Analog Modulation Recognition.

Fang, R., & Ma, H. (2006). Application of MCSA and SVM to induction machine rotor fault diagnosis. *Proc. of the 6th World Congress on Intelligent Control and Automation, China*, 5543-5547.

IEEE Operation Center (1997). Advancement in Microprocessor Based Protection and Communication, IEEE Tutorial Course Text, 97-TP-120-0, Piscataway, NJ.

Farag, W. A., & Kamel, M. I. (1999). Microprocessor-based protection system for three-phase induction motor. *Electric Machines and Power Systems*, vol. 27, 453-464.

Jain, A .K., Mao, J., & Mohiuddin, K .M. (1996). Artificial Neural Networks: A Tutorial. Retrieved from [http://www.cogsci.ucsd.edu/~ajyu/Teaching/Cogs202\\_sp12/Readings/jain\\_ann96.pdf](http://www.cogsci.ucsd.edu/~ajyu/Teaching/Cogs202_sp12/Readings/jain_ann96.pdf).

Jack, L. B., & Nandi, A. K. (2002). Faults detection using support vector machines and artificial neural network, augmented by genetic algorithms. *Mechanical System and Signal Processing*, 16, 373-390.

- Kezunovic, M. (1997). A survey of neural network application to protective relaying and fault analysis. *Engineering Intelligent Systems*, vol. 5, No. 4, 185-192.
- Kolla, S., & Altman, S. (2007). Artificial neural network based fault identification scheme implementation for a three-phase induction motor. *ISA Transactions*, vol. 46, No.2, 261-266.
- Krishnamoorthy, C., & S. Rajeev. (1996). *Artificial Intelligence and Expert systems for Engineers*. CRC Press.
- Kolla, S., & Varatharasa, L. (2000). Identifying three-phase induction motor faults using artificial neural networks. *ISA Transactions*, vol. 39, no. 4, 433-439.
- Korkeakoulu, T., & Hogskola, T. (2004). Support Vector Machine Based Classification in Condition Monitoring of Induction Motors. *Helsinki University of Technology Control Engg. Lab, ESPOO* .
- Kriegel, J. M., Arnhold, T., Beck, B., & Fox, T. (2005). A support vector machine approach to classify human cytochrome P450 3A4 inhibitors. *Journal of Computer-Aided Molecular Design*, 19, 189-201.
- Konar, P., & Chattopadhyay, P. (2011). Bearing fault detection of induction motor using wavelet and Support Vector Machines (SVMs). *Applied Soft Computing*, 4203-4211.
- Min, J. H., & Lee, Y. C. (2005). Bankruptcy using support vectors machine with optimal choice of kernel function parameters. *Experts Systems with Application*, 28, 603-614.
- Mellit, A., & Kalogirou, S. A. (2008). Artificial intelligence techniques for photovoltaic application: A review. *Progress in Energy and Combustion Science*, vol. 34, no. 5, 574-



632.

- Nguyen, N.-T., & Lee, H.-H. (2008). An application of support vector machines for induction motor fault diagnosis with using genetic algorithms. *Proc. of ICIC 2008, LNAI5227*, 190-200. Berlin: Springer.
- Poyhonen, S., Arkkio, A., Jover, P., & Hyotyniemi, H. (2005). Coupling Pairwise support vector machine for faults classification. *Control Engineering Practice*, vol. 13, 759-769.
- Sudha, M., & Anbalagan, P. (2009). A Protection Scheme for Three-Phase Induction Motor from Incipient Faults Using Embedded Controller. *Asian Journal of Scientific Research* , 28-50.
- Sachdev, M. S. (1988). Microprocessor relays and protection systems. *IEEE Tutorial Course Txt 88EH0269-1-P WR* .
- Sachdev, M. S. (1997). Computer relaying. *IEEE Tutorial Course Text, Publication No. 97TP120-0* , 1-79.
- Siddique, A., Yadava, G. S., & Singh, B. (2005). A review of stator fault monitoring techniques of induction motors. *IEEE Transaction on Energy Conversion*, vol. 20, 106-114.
- Wan, T. W. , & Hong, H. (2001). An on-line neuro-fuzzy approach fordetecting faults in induction motors. *Electrical Machine Drives Conference, IEMDC 20001*, Cambridge, MA, 878-883.
- Wasserman, P. (1989). *Neural Computing*. Van Nostrand Reinhold, New York, NY.
- Wang, L. (2005). *Support Vector Machine: Theory and Applications*. Poland: Springer.

Weston, J. Support Vectors Machine Tutorial. *NEC Labs America*. Retrieved from

[http://www.cs.columbia.edu/~kathy/cs4701/documents/jason\\_svm\\_tutorial.pdf](http://www.cs.columbia.edu/~kathy/cs4701/documents/jason_svm_tutorial.pdf).

Widodo, A., & B-S. Yang. (2007). Support vector machine in machine condition and fault diagnosis. *Mechanical System and Signal Processing*, vol.21, 2560-2574.

Wu, C., Liu, F., & Wang, M. (2013). Recognition of Control Chart Patterns Using Support Vector Machine Based Classifier with Features Extracted from Clustering. *Metalurgia International*, 5-10.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, vol. 27, no. 11, 1134-1142.

## Appendix A

## Training Data

No. of patterns: 788

No. of inputs: 6

No. of outputs 7

	V1	V2	V3	I1	I2	I3
1	1: 2.661025	2: 2.624276	3: 2.701274	4: 0.4907675	5: 0.478549	6: 0.4933681
1	1: 2.660319	2: 2.624661	3: 2.7007	4: 0.4911136	5: 0.4787216	6: 0.4925841
1	1: 2.658824	2: 2.623414	3: 2.699373	4: 0.4907824	5: 0.4780662	6: 0.4924988
1	1: 2.659359	2: 2.623496	3: 2.700214	4: 0.4909006	5: 0.4778042	6: 0.4928816
1	1: 2.659918	2: 2.624527	3: 2.701876	4: 0.4911553	5: 0.4781739	6: 0.4923389
1	1: 2.660713	2: 2.624278	3: 2.700593	4: 0.4912516	5: 0.4782521	6: 0.4925644
1	1: 2.660079	2: 2.623479	3: 2.699637	4: 0.4908446	5: 0.478162	6: 0.4926641
1	1: 2.65911	2: 2.623836	3: 2.699883	4: 0.4906552	5: 0.4780388	6: 0.4923748
1	1: 2.660678	2: 2.624341	3: 2.70043	4: 0.4913352	5: 0.4774227	6: 0.4927595
1	1: 2.661025	2: 2.624276	3: 2.701274	4: 0.4907675	5: 0.478549	6: 0.4933681
1	1: 2.653508	2: 2.618927	3: 2.695269	4: 0.6115847	5: 0.597014	6: 0.6095109
1	1: 2.65609	2: 2.619306	3: 2.695886	4: 0.6115017	5: 0.5960048	6: 0.6103757
1	1: 2.655185	2: 2.620171	3: 2.695862	4: 0.6117206	5: 0.596801	6: 0.6094408
1	1: 2.654937	2: 2.619618	3: 2.6959	4: 0.6114231	5: 0.5967827	6: 0.6096965
1	1: 2.655788	2: 2.619876	3: 2.695217	4: 0.6120566	5: 0.5962501	6: 0.6095838
1	1: 2.655232	2: 2.620571	3: 2.695642	4: 0.6120045	5: 0.5961067	6: 0.6097577
1	1: 2.655519	2: 2.61995	3: 2.696013	4: 0.6117751	5: 0.5963824	6: 0.6097241
1	1: 2.65538	2: 2.621187	3: 2.696468	4: 0.6115075	5: 0.597183	6: 0.6089038
1	1: 2.65524	2: 2.620778	3: 2.696097	4: 0.6114631	5: 0.5966644	6: 0.6093523
1	1: 2.654109	2: 2.6177	3: 2.696032	4: 0.6112516	5: 0.5972841	6: 0.6095978
1	1: 2.652949	2: 2.617502	3: 2.691755	4: 0.679997	5: 0.6630695	6: 0.675884
1	1: 2.653947	2: 2.61846	3: 2.69309	4: 0.6802169	5: 0.6630145	6: 0.6758427
1	1: 2.654646	2: 2.619546	3: 2.694069	4: 0.6801607	5: 0.6630943	6: 0.6760761
1	1: 2.653576	2: 2.619166	3: 2.692964	4: 0.6800552	5: 0.6637426	6: 0.6757043
1	1: 2.65484	2: 2.620561	3: 2.694237	4: 0.6799366	5: 0.6642037	6: 0.6756256

1	1: 2.65575	2: 2.621506	3: 2.696075	4: 0.6797796	5: 0.6641113	6: 0.6756669
1	1: 2.652686	2: 2.619627	3: 2.692855	4: 0.6804986	5: 0.6638437	6: 0.6753921
1	1: 2.653393	2: 2.619608	3: 2.692457	4: 0.6804592	5: 0.6633366	6: 0.6759489
1	1: 2.654018	2: 2.619407	3: 2.692362	4: 0.6803204	5: 0.6632937	6: 0.6759761
1	1: 2.65416	2: 2.61867	3: 2.69252	4: 0.6800834	5: 0.6631575	6: 0.6757772
1	1: 2.651632	2: 2.6181	3: 2.692459	4: 0.7442518	5: 0.7277177	6: 0.7385438
1	1: 2.651218	2: 2.618053	3: 2.691295	4: 0.7447775	5: 0.7273625	6: 0.738741
1	1: 2.650345	2: 2.616702	3: 2.689826	4: 0.7448472	5: 0.7271636	6: 0.7396483
1	1: 2.650834	2: 2.616907	3: 2.690293	4: 0.7445992	5: 0.7273633	6: 0.7399066
1	1: 2.650428	2: 2.616399	3: 2.690159	4: 0.7444701	5: 0.72786	6: 0.7399503
1	1: 2.651311	2: 2.617485	3: 2.691184	4: 0.7447671	5: 0.7276557	6: 0.7399828
1	1: 2.651568	2: 2.617853	3: 2.691633	4: 0.7451041	5: 0.727931	6: 0.7398077
1	1: 2.652178	2: 2.619033	3: 2.693182	4: 0.745323	5: 0.7283805	6: 0.7391569
1	1: 2.653515	2: 2.619257	3: 2.693602	4: 0.7455138	5: 0.7279445	6: 0.7393124
1	1: 2.65034	2: 2.616713	3: 2.69016	4: 0.7459358	5: 0.7284408	6: 0.7394889
1	1: 2.697322	2: 2.649326	3: 2.695952	4: 0.434414	5: 0.4142901	6: 0.4124829
1	1: 2.696793	2: 2.648861	3: 2.694307	4: 0.4350086	5: 0.4139784	6: 0.4129565
1	1: 2.697142	2: 2.648501	3: 2.69492	4: 0.4344819	5: 0.4135517	6: 0.4123409
1	1: 2.697197	2: 2.648599	3: 2.694938	4: 0.4337505	5: 0.4135587	6: 0.4126464
1	1: 2.696565	2: 2.648551	3: 2.694751	4: 0.4345582	5: 0.414148	6: 0.413638
1	1: 2.696184	2: 2.648384	3: 2.696225	4: 0.4334154	5: 0.4146245	6: 0.4135129
1	1: 2.695967	2: 2.648392	3: 2.694897	4: 0.4328751	5: 0.4138973	6: 0.4135326
1	1: 2.696562	2: 2.648019	3: 2.695978	4: 0.4344367	5: 0.4144415	6: 0.4141197
1	1: 2.696226	2: 2.648179	3: 2.696942	4: 0.4341586	5: 0.414401	6: 0.413102
1	1: 2.697268	2: 2.648663	3: 2.696495	4: 0.4333051	5: 0.4137648	6: 0.4131066
1	1: 2.697365	2: 2.648982	3: 2.696256	4: 0.4343351	5: 0.4144157	6: 0.4133662
1	1: 2.696713	2: 2.648403	3: 2.695808	4: 0.43451	5: 0.4140275	6: 0.4128283
1	1: 2.697153	2: 2.647973	3: 2.695733	4: 0.4130617	5: 0.4215126	6: 0.4117201
1	1: 2.696436	2: 2.648303	3: 2.67724	4: 0.4153801	5: 0.4021312	6: 0.4133645
1	1: 2.697099	2: 2.64851	3: 2.676638	4: 0.4154756	5: 0.401545	6: 0.4129759
1	1: 2.69736	2: 2.649193	3: 2.675709	4: 0.4147182	5: 0.4011172	6: 0.4120305
1	1: 2.696463	2: 2.649131	3: 2.671853	4: 0.4529331	5: 0.4341953	6: 0.450655
1	1: 2.696757	2: 2.648782	3: 2.66626	4: 0.5218772	5: 0.4979789	6: 0.5183506
1	1: 2.695643	2: 2.648685	3: 2.664089	4: 0.5214739	5: 0.4979619	6: 0.5183697

1	1: 2.695727	2: 2.648845	3: 2.664536	4: 0.5217751	5: 0.4978073	6: 0.5178254
1	1: 2.695743	2: 2.648656	3: 2.664514	4: 0.5231938	5: 0.4972155	6: 0.5175851
1	1: 2.696513	2: 2.649003	3: 2.665331	4: 0.5235548	5: 0.496991	6: 0.5186568
1	1: 2.695711	2: 2.648852	3: 2.685261	4: 0.4036345	5: 0.4230579	6: 0.4139806
1	1: 2.695124	2: 2.648784	3: 2.686463	4: 0.4039669	5: 0.4234286	6: 0.4132729
1	1: 2.646549	2: 2.648902	3: 2.686672	4: 0.4039952	5: 0.4230519	6: 0.4131241
1	1: 2.641693	2: 2.600013	3: 2.687235	4: 0.4044011	5: 0.4235151	6: 0.4132784
1	1: 2.641789	2: 2.599556	3: 2.688995	4: 0.4046342	5: 0.4240693	6: 0.4143183
1	1: 2.639579	2: 2.597977	3: 2.688344	4: 0.4048383	5: 0.4239191	6: 0.4146179
1	1: 2.639855	2: 2.598443	3: 2.689093	4: 0.4050134	5: 0.4242126	6: 0.4145411
1	1: 2.63905	2: 2.597599	3: 2.689892	4: 0.4047301	5: 0.4245637	6: 0.4150462
1	1: 2.639127	2: 2.597583	3: 2.689994	4: 0.4132204	5: 0.4327742	6: 0.4234085
1	1: 2.638714	2: 2.596648	3: 2.686446	4: 0.4178508	5: 0.43715	6: 0.4279297
1	1: 2.637364	2: 2.595923	3: 2.689517	4: 0.4182805	5: 0.4375683	6: 0.4283726
1	1: 2.634876	2: 2.5936	3: 2.688503	4: 0.4179565	5: 0.4377481	6: 0.4281828
1	1: 2.634664	2: 2.593509	3: 2.690073	4: 0.4185042	5: 0.4379684	6: 0.4286435
1	1: 2.634139	2: 2.59151	3: 2.690111	4: 0.4182101	5: 0.4375099	6: 0.4285923
1	1: 2.633987	2: 2.590933	3: 2.690223	4: 0.4180262	5: 0.437491	6: 0.428648
1	1: 2.634248	2: 2.590949	3: 2.690487	4: 0.4183157	5: 0.4373334	6: 0.4283918
1	1: 2.634075	2: 2.59103	3: 2.690613	4: 0.4178572	5: 0.4375141	6: 0.4277986
1	1: 2.634677	2: 2.59151	3: 2.689216	4: 0.4176738	5: 0.4373995	6: 0.4275657
1	1: 2.63442	2: 2.591173	3: 2.690975	4: 0.4184826	5: 0.4374706	6: 0.4282448
1	1: 2.634634	2: 2.592161	3: 2.689744	4: 0.4177855	5: 0.4369351	6: 0.4287377
1	1: 2.634549	2: 2.592067	3: 2.692192	4: 0.4156703	5: 0.4342401	6: 0.4262839
1	1: 2.652089	2: 2.64112	3: 2.68001	4: 0.5457847	5: 0.5656723	6: 0.5598875
1	1: 2.651276	2: 2.639724	3: 2.682293	4: 0.5426228	5: 0.5620964	6: 0.5561645
1	1: 2.651127	2: 2.640616	3: 2.681654	4: 0.5426836	5: 0.5617902	6: 0.5565209
1	1: 2.651925	2: 2.640871	3: 2.692761	4: 0.4156144	5: 0.4342483	6: 0.4264356
1	1: 2.653449	2: 2.643286	3: 2.693767	4: 0.4158215	5: 0.4343628	6: 0.4259275
1	1: 2.652623	2: 2.642312	3: 2.692896	4: 0.4149959	5: 0.4345885	6: 0.4260514
1	1: 2.653241	2: 2.641718	3: 2.692696	4: 0.4154069	5: 0.4349161	6: 0.4256034
1	1: 2.652621	2: 2.642102	3: 2.692859	4: 0.4153117	5: 0.4351994	6: 0.4256902
1	1: 2.652579	2: 2.642174	3: 2.692337	4: 0.4150669	5: 0.4349843	6: 0.4257393
1	1: 2.653431	2: 2.642523	3: 2.692934	4: 0.4151706	5: 0.4352474	6: 0.4261055

1	1: 2.654109	2: 2.64296	3: 2.691661	4: 0.4149314	5: 0.4349117	6: 0.4261536
1	1: 2.654102	2: 2.642343	3: 2.69321	4: 0.4156473	5: 0.4350146	6: 0.4262911
1	1: 2.653137	2: 2.642589	3: 2.691524	4: 0.4157361	5: 0.4344165	6: 0.4259987
1	1: 2.653778	2: 2.643097	3: 2.675662	4: 0.620412	5: 0.6393502	6: 0.6342357
1	1: 2.654194	2: 2.643574	3: 2.675408	4: 0.6801454	5: 0.6987316	6: 0.6951651
1	1: 2.65549	2: 2.643652	3: 2.675695	4: 0.677575	5: 0.6959341	6: 0.692548
1	1: 2.656063	2: 2.645024	3: 2.675057	4: 0.6747949	5: 0.6932238	6: 0.6903382
1	1: 2.655919	2: 2.644959	3: 2.673281	4: 0.6715187	5: 0.6910382	6: 0.6873876
1	1: 2.655133	2: 2.64429	3: 2.675755	4: 0.6700318	5: 0.6890166	6: 0.6850387
1	1: 2.653988	2: 2.643141	3: 2.675185	4: 0.6694633	5: 0.6871263	6: 0.6848475
1	1: 2.65658	2: 2.644599	3: 2.676314	4: 0.6675897	5: 0.6859077	6: 0.68242
1	1: 2.657052	2: 2.645792	3: 2.674477	4: 0.6656589	5: 0.6838555	6: 0.6810809
1	1: 2.656885	2: 2.645248	3: 2.674548	4: 0.6651244	5: 0.6838646	6: 0.6807088
1	1: 2.656238	2: 2.645165	3: 2.673669	4: 0.6640027	5: 0.6829661	6: 0.679563
1	1: 2.65649	2: 2.645886	3: 2.673125	4: 0.6628118	5: 0.6822056	6: 0.6787031
1	1: 2.65845	2: 2.646958	3: 2.674316	4: 0.6615382	5: 0.6801103	6: 0.6765433
1	1: 2.65761	2: 2.64657	3: 2.674271	4: 0.6614415	5: 0.680144	6: 0.6765336
1	1: 2.657395	2: 2.646112	3: 2.672164	4: 0.7466173	5: 0.766124	6: 0.7626198
1	1: 2.658366	2: 2.646966	3: 2.671093	4: 0.7430073	5: 0.762291	6: 0.7589417
1	1: 2.657933	2: 2.646438	3: 2.671285	4: 0.7379	5: 0.7576979	6: 0.7534468
1	1: 2.657607	2: 2.64599	3: 2.669959	4: 0.7382876	5: 0.7566167	6: 0.7541529
1	1: 2.657114	2: 2.64607	3: 2.670923	4: 0.7452279	5: 0.7477898	6: 0.7423192
1	1: 2.656556	2: 2.645507	3: 2.671249	4: 0.7507145	5: 0.7392567	6: 0.7430079
1	1: 2.655658	2: 2.6451	3: 2.687579	4: 0.7521474	5: 0.7250581	6: 0.7523645
1	1: 2.655022	2: 2.643788	3: 2.688266	4: 0.7491037	5: 0.7237691	6: 0.7506114
1	1: 2.655042	2: 2.644289	3: 2.687682	4: 0.7481878	5: 0.723229	6: 0.7483666
1	1: 2.6555	2: 2.644512	3: 2.688714	4: 0.7490288	5: 0.7239705	6: 0.7500601
1	1: 2.65719	2: 2.646107	3: 2.689996	4: 0.7481137	5: 0.7227195	6: 0.7501977
1	1: 2.658417	2: 2.647034	3: 2.690814	4: 0.7450285	5: 0.7190654	6: 0.746648
1	1: 2.657996	2: 2.645384	3: 2.68963	4: 0.8143297	5: 0.7855652	6: 0.8147647
1	1: 2.658478	2: 2.646319	3: 2.707983	4: 0.427676	5: 0.4178647	6: 0.4412994
1	1: 2.659029	2: 2.647599	3: 2.709454	4: 0.4170238	5: 0.4083359	6: 0.4316729
1	1: 2.656146	2: 2.644919	3: 2.710732	4: 0.4181483	5: 0.4094471	6: 0.4324603
1	1: 2.653172	2: 2.642973	3: 2.710961	4: 0.4180101	5: 0.4094621	6: 0.4324501

1	1: 2.654418	2: 2.643044	3: 2.710733	4: 0.4182363	5: 0.4100225	6: 0.4325742
1	1: 2.655461	2: 2.644923	3: 2.710531	4: 0.4189846	5: 0.4105237	6: 0.4327792
1	1: 2.654501	2: 2.644346	3: 2.71189	4: 0.4191359	5: 0.4113073	6: 0.4327083
1	1: 2.653847	2: 2.643722	3: 2.712264	4: 0.4188605	5: 0.4113317	6: 0.4327599
1	1: 2.658205	2: 2.646481	3: 2.708875	4: 0.4280554	5: 0.4194082	6: 0.4410295
1	1: 2.659536	2: 2.647071	3: 2.70831	4: 0.4513036	5: 0.4393579	6: 0.462926
1	1: 2.659633	2: 2.647491	3: 2.707507	4: 0.4431409	5: 0.4316343	6: 0.4550797
1	1: 2.659842	2: 2.647347	3: 2.707987	4: 0.4454578	5: 0.4339773	6: 0.4565783
1	1: 2.659231	2: 2.646948	3: 2.706961	4: 0.4508948	5: 0.4388415	6: 0.4624864
1	1: 2.658519	2: 2.646097	3: 2.707444	4: 0.4609749	5: 0.4475574	6: 0.4728047
1	1: 2.657581	2: 2.646291	3: 2.703551	4: 0.4942202	5: 0.4782696	6: 0.5032329
1	1: 2.657764	2: 2.646167	3: 2.702805	4: 0.511809	5: 0.495457	6: 0.5203554
1	1: 2.658664	2: 2.647373	3: 2.712137	4: 0.4259365	5: 0.4170007	6: 0.4385732
1	1: 2.657801	2: 2.647148	3: 2.711155	4: 0.4262641	5: 0.4170549	6: 0.4393252
1	1: 2.657986	2: 2.646044	3: 2.711289	4: 0.4262034	5: 0.4232072	6: 0.4375025
1	1: 2.657416	2: 2.646575	3: 2.695097	4: 0.4296375	5: 0.4253608	6: 0.4291741
1	1: 2.658294	2: 2.647373	3: 2.694492	4: 0.4252739	5: 0.4268809	6: 0.4297068
1	1: 2.658563	2: 2.647855	3: 2.695126	4: 0.4263157	5: 0.4267686	6: 0.4297866
1	1: 2.658965	2: 2.647678	3: 2.686654	4: 0.538016	5: 0.5347058	6: 0.5378957
1	1: 2.658453	2: 2.646374	3: 2.686501	4: 0.5385297	5: 0.5351173	6: 0.5391469
1	1: 2.659259	2: 2.647441	3: 2.685979	4: 0.5382825	5: 0.5352463	6: 0.5390682
1	1: 2.659412	2: 2.648036	3: 2.6954	4: 0.4264752	5: 0.4269881	6: 0.4295657
1	1: 2.659789	2: 2.647579	3: 2.696819	4: 0.4270584	5: 0.4271924	6: 0.4297657
1	1: 2.658994	2: 2.646827	3: 2.69709	4: 0.4273545	5: 0.4278249	6: 0.4307352
1	1: 2.653513	2: 2.64219	3: 2.69762	4: 0.4277468	5: 0.4283656	6: 0.4305478
1	1: 2.650308	2: 2.640349	3: 2.696875	4: 0.4277991	5: 0.4282332	6: 0.4313022
1	1: 2.649964	2: 2.639867	3: 2.696419	4: 0.4277886	5: 0.4283571	6: 0.4313133
2	1: 2.639052	2: 2.593193	3: 2.66701	4: 0.0062597	5: 0.6447509	6: 0.6415373
2	1: 2.644109	2: 2.597327	3: 2.669648	4: 0.0061618	5: 0.643999	6: 0.6405873
2	1: 2.647625	2: 2.598815	3: 2.671626	4: 0.0061937	5: 0.6435176	6: 0.6402169
2	1: 2.647963	2: 2.599153	3: 2.67209	4: 0.0062368	5: 0.6425905	6: 0.6393902
2	1: 2.651849	2: 2.601844	3: 2.673132	4: 0.006156	5: 0.6422666	6: 0.63901
2	1: 2.650816	2: 2.601661	3: 2.673722	4: 0.006123	5: 0.641548	6: 0.6385528
2	1: 2.628087	2: 2.602545	3: 2.672384	4: 0.0061942	5: 0.6407932	6: 0.6373292

2	1: 2.649681	2: 2.603404	3: 2.6744	4: 0.0060545	5: 0.6397968	6: 0.636463
2	1: 2.650662	2: 2.604571	3: 2.675484	4: 0.0060614	5: 0.6391505	6: 0.6359752
2	1: 2.650347	2: 2.604759	3: 2.675032	4: 0.0061104	5: 0.6387148	6: 0.6352944
2	1: 2.625339	2: 2.624541	3: 2.669839	4: 0.6252548	5: 0.0145094	6: 0.6125097
2	1: 2.621724	2: 2.62397	3: 2.668056	4: 0.6244646	5: 0.0146452	6: 0.611628
2	1: 2.622185	2: 2.623376	3: 2.668053	4: 0.6242134	5: 0.0145427	6: 0.6114829
2	1: 2.621597	2: 2.623556	3: 2.66816	4: 0.6240045	5: 0.0144969	6: 0.6110426
2	1: 2.618958	2: 2.619706	3: 2.665676	4: 0.6227452	5: 0.0145174	6: 0.6098157
2	1: 2.61912	2: 2.621497	3: 2.667474	4: 0.6227121	5: 0.014418	6: 0.6098734
2	1: 2.618631	2: 2.621098	3: 2.66667	4: 0.6226041	5: 0.0142608	6: 0.6097428
2	1: 2.61782	2: 2.619749	3: 2.665758	4: 0.6222544	5: 0.014206	6: 0.6092628
2	1: 2.624021	2: 2.6242	3: 2.669475	4: 0.6250926	5: 0.0145195	6: 0.6122303
2	1: 2.622269	2: 2.624548	3: 2.669342	4: 0.6243864	5: 0.0144204	6: 0.6116893
2	1: 2.624702	2: 2.608564	3: 2.660888	4: 0.622824	5: 0.6266529	6: 0.0142696
2	1: 2.625152	2: 2.610523	3: 2.661465	4: 0.6229203	5: 0.6269173	6: 0.0142131
2	1: 2.626952	2: 2.611647	3: 2.661433	4: 0.6232172	5: 0.6272398	6: 0.0142552
2	1: 2.62758	2: 2.611288	3: 2.661562	4: 0.6231534	5: 0.6270218	6: 0.0143278
2	1: 2.629568	2: 2.611594	3: 2.663084	4: 0.6234588	5: 0.6272577	6: 0.0142738
2	1: 2.628732	2: 2.612757	3: 2.662817	4: 0.623511	5: 0.6273411	6: 0.0143363
2	1: 2.628282	2: 2.614013	3: 2.660388	4: 0.6229229	5: 0.6266871	6: 0.0144627
2	1: 2.624851	2: 2.608232	3: 2.660788	4: 0.6226623	5: 0.6263832	6: 0.0141404
2	1: 2.625761	2: 2.610769	3: 2.660925	4: 0.6231158	5: 0.6270782	6: 0.0142361
2	1: 2.627837	2: 2.611571	3: 2.661799	4: 0.6231391	5: 0.6271357	6: 0.0143238
2	1: 2.651062	2: 2.64164	3: 2.695483	4: 0.0059256	5: 0.635686	6: 0.6395525
2	1: 2.652171	2: 2.642029	3: 2.695508	4: 0.0059842	5: 0.6359173	6: 0.6396987
2	1: 2.651511	2: 2.641484	3: 2.696072	4: 0.005867	5: 0.6360843	6: 0.6400133
2	1: 2.65127	2: 2.641222	3: 2.721691	4: 0.6358515	5: 0.0180078	6: 0.6350757
2	1: 2.650634	2: 2.640998	3: 2.723218	4: 0.6362488	5: 0.018181	6: 0.6353615
2	1: 2.650586	2: 2.64078	3: 2.721407	4: 0.6353304	5: 0.0181311	6: 0.6345674
2	1: 2.648657	2: 2.639017	3: 2.721568	4: 0.635591	5: 0.0183736	6: 0.6347775
2	1: 2.648983	2: 2.639854	3: 2.720525	4: 0.6355465	5: 0.0181334	6: 0.6347592
2	1: 2.649559	2: 2.639871	3: 2.716151	4: 0.6340056	5: 0.6385465	6: 0.0085759
2	1: 2.649984	2: 2.640135	3: 2.717418	4: 0.6343415	5: 0.6389635	6: 0.0085832
2	1: 2.649557	2: 2.639409	3: 2.716575	4: 0.6340744	5: 0.6385038	6: 0.0084875



2	1: 2.650535	2: 2.639366	3: 2.716651	4: 0.6339869	5: 0.6386513	6: 0.0087568
2	1: 2.6514	2: 2.641298	3: 2.716903	4: 0.6342775	5: 0.6388557	6: 0.0086837
2	1: 2.650317	2: 2.640077	3: 2.717748	4: 0.6341964	5: 0.6387232	6: 0.008864
2	1: 2.648514	2: 2.638497	3: 2.716745	4: 0.6342112	5: 0.6387797	6: 0.0087673
2	1: 2.649291	2: 2.639254	3: 2.690204	4: 0.0059362	5: 0.6855779	6: 0.6894485
2	1: 2.649141	2: 2.639075	3: 2.690852	4: 0.0059136	5: 0.6855621	6: 0.689427
2	1: 2.648866	2: 2.639196	3: 2.692173	4: 0.0059035	5: 0.6861497	6: 0.6898886
2	1: 2.647392	2: 2.638237	3: 2.694569	4: 0.0058949	5: 0.6868173	6: 0.6908325
2	1: 2.648971	2: 2.639116	3: 2.717915	4: 0.6882356	5: 0.0174556	6: 0.6873411
2	1: 2.649124	2: 2.639234	3: 2.717932	4: 0.6885232	5: 0.0173706	6: 0.6877941
2	1: 2.64898	2: 2.639116	3: 2.717281	4: 0.6887263	5: 0.0175646	6: 0.6878852
2	1: 2.648118	2: 2.63885	3: 2.718129	4: 0.6887667	5: 0.0173646	6: 0.6880867
2	1: 2.648632	2: 2.638591	3: 2.718193	4: 0.6888208	5: 0.0171492	6: 0.6880753
2	1: 2.646077	2: 2.636828	3: 2.718166	4: 0.6887856	5: 0.0172653	6: 0.6878704
2	1: 2.645513	2: 2.636638	3: 2.719331	4: 0.6867101	5: 0.6909426	6: 0.0090622
2	1: 2.645542	2: 2.636956	3: 2.719579	4: 0.6869455	5: 0.6910623	6: 0.0090868
2	1: 2.645947	2: 2.638145	3: 2.720622	4: 0.687193	5: 0.6913996	6: 0.0090688
2	1: 2.646446	2: 2.63677	3: 2.721074	4: 0.6873272	5: 0.691329	6: 0.0090835
2	1: 2.644443	2: 2.635756	3: 2.721169	4: 0.6877181	5: 0.691697	6: 0.0092696
2	1: 2.645974	2: 2.637226	3: 2.72084	4: 0.6875339	5: 0.6915793	6: 0.0091268
2	1: 2.645472	2: 2.636754	3: 2.720566	4: 0.6876772	5: 0.6915676	6: 0.0089998
2	1: 2.670274	2: 2.636588	3: 2.721604	4: 0.6873366	5: 0.6913805	6: 0.0091837
2	1: 2.673924	2: 2.635932	3: 2.687713	4: 0.0056385	5: 0.9767016	6: 0.9808123
2	1: 2.674132	2: 2.619226	3: 2.6875	4: 0.0054999	5: 0.9851511	6: 0.9892578
2	1: 2.675325	2: 2.601152	3: 2.687416	4: 0.0056243	5: 0.9886547	6: 0.992827
2	1: 2.675141	2: 2.601195	3: 2.685054	4: 0.0055254	5: 0.9916154	6: 0.9955447
2	1: 2.674113	2: 2.600739	3: 2.708484	4: 0.9889787	5: 0.0168184	6: 0.9888266
2	1: 2.674911	2: 2.600459	3: 2.70828	4: 0.9847451	5: 0.0165438	6: 0.9847818
2	1: 2.674748	2: 2.600274	3: 2.708417	4: 0.9873922	5: 0.0164193	6: 0.9874103
2	1: 2.674631	2: 2.600208	3: 2.708328	4: 0.9892695	5: 0.0165024	6: 0.9892967
2	1: 2.674161	2: 2.599297	3: 2.719579	4: 0.9884981	5: 0.9909444	6: 0.008858
2	1: 2.676485	2: 2.600618	3: 2.719054	4: 0.9897136	5: 0.9923492	6: 0.0087745
2	1: 2.676883	2: 2.602028	3: 2.719336	4: 0.9908766	5: 0.9934195	6: 0.0089922
2	1: 2.677131	2: 2.601172	3: 2.719327	4: 0.988566	5: 0.9911124	6: 0.0090497

2	1: 2.675903	2: 2.599575	3: 2.720678	4: 0.9902932	5: 0.9927527	6: 0.0091499
2	1: 2.679408	2: 2.603977	3: 2.702923	4: 0.0044858	5: 0.6300898	6: 0.6338261
2	1: 2.684002	2: 2.607824	3: 2.70151	4: 0.0045288	5: 0.6297895	6: 0.6334688
2	1: 2.686174	2: 2.609325	3: 2.700067	4: 0.0053338	5: 0.6299559	6: 0.6335398
2	1: 2.687533	2: 2.608335	3: 2.724302	4: 0.628893	5: 0.0049897	6: 0.6274013
2	1: 2.687124	2: 2.609382	3: 2.723326	4: 0.6288567	5: 0.0048804	6: 0.627361
2	1: 2.687497	2: 2.609831	3: 2.723994	4: 0.6289026	5: 0.0047561	6: 0.627362
2	1: 2.686947	2: 2.609763	3: 2.724014	4: 0.6291794	5: 0.0047997	6: 0.6275301
2	1: 2.686833	2: 2.609658	3: 2.720394	4: 0.6249679	5: 0.6285737	6: 0.0131279
2	1: 2.688366	2: 2.608813	3: 2.721451	4: 0.6251115	5: 0.6287187	6: 0.013141
3	1: 0.9225996	2: 2.620864	3: 2.625168	4: 0.1721304	5: 0.7720316	6: 0.6629209
3	1: 0.9211622	2: 2.619409	3: 2.624676	4: 0.1717272	5: 0.7721959	6: 0.6629794
3	1: 0.9207521	2: 2.620637	3: 2.624294	4: 0.1718667	5: 0.7717378	6: 0.6624818
3	1: 0.9199523	2: 2.622002	3: 2.625915	4: 0.1724143	5: 0.7726538	6: 0.663118
3	1: 0.9195703	2: 2.6214	3: 2.626511	4: 0.1721125	5: 0.7724191	6: 0.6627584
3	1: 0.9191691	2: 2.621913	3: 2.625237	4: 0.1726076	5: 0.7722042	6: 0.6627471
3	1: 0.9218292	2: 2.62031	3: 2.625566	4: 0.1718554	5: 0.7723441	6: 0.6629364
3	1: 0.9206891	2: 2.618773	3: 2.62415	4: 0.1719643	5: 0.7718436	6: 0.6629226
3	1: 0.9199519	2: 2.620756	3: 2.625503	4: 0.1718144	5: 0.7720957	6: 0.6626348
3	1: 0.9198515	2: 2.621627	3: 2.6253	4: 0.1720716	5: 0.7721062	6: 0.6626533
3	1: 2.600413	2: 1.303555	3: 2.667813	4: 0.6857667	5: 0.1734909	6: 0.7607303
3	1: 2.59924	2: 1.301566	3: 2.667279	4: 0.6852569	5: 0.1739493	6: 0.7607235
3	1: 2.596522	2: 1.30034	3: 2.666048	4: 0.6847681	5: 0.1735806	6: 0.7603071
3	1: 2.600644	2: 1.299432	3: 2.665714	4: 0.6858547	5: 0.173042	6: 0.7598948
3	1: 2.601971	2: 1.299378	3: 2.667823	4: 0.6856466	5: 0.1731501	6: 0.760505
3	1: 2.602494	2: 1.300968	3: 2.671745	4: 0.6862939	5: 0.174582	6: 0.7623009
3	1: 2.604725	2: 1.29962	3: 2.670581	4: 0.6863873	5: 0.173785	6: 0.7614975
3	1: 2.604409	2: 1.299682	3: 2.671277	4: 0.685783	5: 0.1733134	6: 0.7606852
3	1: 2.600881	2: 1.302418	3: 2.667618	4: 0.6858006	5: 0.1737468	6: 0.7610977
3	1: 2.592163	2: 1.300948	3: 2.666902	4: 0.6834224	5: 0.1747393	6: 0.7604709
3	1: 2.639939	2: 2.578533	3: 1.369383	4: 0.7661272	5: 0.6655651	6: 0.1651184
3	1: 2.637952	2: 2.578868	3: 1.367732	4: 0.7660856	5: 0.6655678	6: 0.1649561
3	1: 2.63637	2: 2.578954	3: 1.366762	4: 0.7656196	5: 0.6651156	6: 0.1649571
3	1: 2.636644	2: 2.580678	3: 1.366535	4: 0.766041	5: 0.6653493	6: 0.1649446

3	1: 2.635708	2: 2.580811	3: 1.366881	4: 0.7659558	5: 0.6649157	6: 0.1647416
3	1: 2.633009	2: 2.578487	3: 1.364784	4: 0.7640203	5: 0.6633822	6: 0.1641112
3	1: 2.632717	2: 2.579411	3: 1.365297	4: 0.7648808	5: 0.6642228	6: 0.1643883
3	1: 2.631879	2: 2.57869	3: 1.364772	4: 0.7647428	5: 0.6641751	6: 0.1645545
3	1: 2.629497	2: 2.576563	3: 1.363402	4: 0.7630999	5: 0.6625988	6: 0.1637454
3	1: 2.639939	2: 2.578533	3: 1.369383	4: 0.7661272	5: 0.6655651	6: 0.1651184
3	1: 1.3733	2: 2.612208	3: 2.690273	4: 0.1756413	5: 0.545693	6: 0.4813709
3	1: 1.393875	2: 2.613278	3: 2.689921	4: 0.176599	5: 0.5454496	6: 0.4802281
3	1: 1.370728	2: 2.609634	3: 2.686523	4: 0.176628	5: 0.5440311	6: 0.4791166
3	1: 1.371299	2: 2.611626	3: 2.688412	4: 0.1770007	5: 0.5443295	6: 0.4796076
3	1: 1.37072	2: 2.611242	3: 2.688011	4: 0.1757156	5: 0.5455513	6: 0.4804097
3	1: 1.370074	2: 2.610919	3: 2.687196	4: 0.1770468	5: 0.5443598	6: 0.4787651
3	1: 1.370629	2: 2.612086	3: 2.688321	4: 0.1761639	5: 0.5459392	6: 0.4799649
3	1: 1.370306	2: 2.612053	3: 2.688004	4: 0.1766888	5: 0.5453934	6: 0.4794296
3	1: 1.369544	2: 2.611253	3: 2.688216	4: 0.1775777	5: 0.5450937	6: 0.4783716
3	1: 1.369484	2: 2.610282	3: 2.686703	4: 0.1750569	5: 0.5461357	6: 0.4805211
3	1: 2.621595	2: 1.919321	3: 2.703427	4: 0.4752193	5: 0.1755215	6: 0.5520643
3	1: 2.621668	2: 1.919544	3: 2.705695	4: 0.4751609	5: 0.1759361	6: 0.5526993
3	1: 1.809361	2: 1.918804	3: 2.705779	4: 0.4747395	5: 0.1759178	6: 0.5525107
3	1: 1.809552	2: 1.917573	3: 2.705622	4: 0.4741605	5: 0.1763637	6: 0.5522856
3	1: 1.809611	2: 1.917886	3: 2.706892	4: 0.4742794	5: 0.1763938	6: 0.552551
3	1: 1.808397	2: 1.915491	3: 2.704026	4: 0.4735866	5: 0.1761771	6: 0.5514545
3	1: 1.810073	2: 1.917782	3: 2.706678	4: 0.4744418	5: 0.1763147	6: 0.5524352
3	1: 1.810501	2: 1.91744	3: 2.706755	4: 0.4745241	5: 0.17616	6: 0.552688
3	1: 1.810646	2: 1.917362	3: 2.707914	4: 0.4748766	5: 0.1761178	6: 0.5527049
3	1: 1.811211	2: 1.917706	3: 2.708271	4: 0.4750451	5: 0.1760048	6: 0.5528433
3	1: 2.653705	2: 2.614147	3: 2.028877	4: 0.543493	5: 0.475775	6: 0.1814861
3	1: 2.654049	2: 2.613937	3: 2.027018	4: 0.543914	5: 0.4748901	6: 0.1820608
3	1: 2.654544	2: 2.614204	3: 2.02644	4: 0.5442713	5: 0.474587	6: 0.1822782
3	1: 2.655128	2: 2.615594	3: 2.026343	4: 0.5441885	5: 0.4748455	6: 0.1822233
3	1: 2.655285	2: 2.61528	3: 2.02612	4: 0.5437202	5: 0.4744471	6: 0.1822292
3	1: 2.653131	2: 2.614144	3: 2.024545	4: 0.5428688	5: 0.4747292	6: 0.1817223
3	1: 2.655784	2: 2.615468	3: 2.025563	4: 0.5428583	5: 0.4747836	6: 0.1821259
3	1: 2.653248	2: 2.614509	3: 2.024135	4: 0.5427941	5: 0.4746458	6: 0.1813862

3	1: 2.656427	2: 2.616841	3: 2.026238	4: 0.5429251	5: 0.4754656	6: 0.1818401
3	1: 2.655554	2: 2.616243	3: 2.025561	4: 0.5428445	5: 0.4753714	6: 0.1818746
3	1: 2.385922	2: 2.629629	3: 2.701563	4: 0.3177741	5: 0.4651335	6: 0.4200537
3	1: 2.385229	2: 2.630017	3: 2.702202	4: 0.3162158	5: 0.4657916	6: 0.4209398
3	1: 2.385149	2: 2.630552	3: 2.703093	4: 0.3192305	5: 0.4634379	6: 0.4181119
3	1: 2.387486	2: 2.632321	3: 2.704263	4: 0.3178545	5: 0.4660915	6: 0.4209347
3	1: 2.386165	2: 2.632894	3: 2.703356	4: 0.316889	5: 0.4646793	6: 0.4204536
3	1: 2.38595	2: 2.632178	3: 2.702839	4: 0.3167746	5: 0.4659957	6: 0.4208159
3	1: 2.386482	2: 2.632602	3: 2.701921	4: 0.3175223	5: 0.4653084	6: 0.4202846
3	1: 2.386478	2: 2.631784	3: 2.701433	4: 0.3165661	5: 0.464897	6: 0.4202667
3	1: 2.385639	2: 2.63159	3: 2.700718	4: 0.3169631	5: 0.4652731	6: 0.4204615
3	1: 2.385577	2: 2.63038	3: 2.700629	4: 0.3159978	5: 0.4640188	6: 0.4199329
3	1: 2.663012	2: 2.35629	3: 2.703289	4: 0.4238377	5: 0.3155979	6: 0.4555021
3	1: 2.660393	2: 2.353268	3: 2.702329	4: 0.4230552	5: 0.3155315	6: 0.4544868
3	1: 2.661469	2: 2.354127	3: 2.704253	4: 0.4228573	5: 0.3155259	6: 0.4575427
3	1: 2.663279	2: 2.355311	3: 2.70546	4: 0.4226383	5: 0.3159655	6: 0.4568847
3	1: 2.662695	2: 2.35337	3: 2.704921	4: 0.423315	5: 0.3153975	6: 0.4583284
3	1: 2.661803	2: 2.353524	3: 2.705673	4: 0.4223462	5: 0.315266	6: 0.4565728
3	1: 2.663421	2: 2.354231	3: 2.705387	4: 0.4233003	5: 0.316494	6: 0.4578612
3	1: 2.662612	2: 2.354356	3: 2.704991	4: 0.4226563	5: 0.3154855	6: 0.455925
3	1: 2.663044	2: 2.354925	3: 2.705945	4: 0.4230778	5: 0.3160763	6: 0.4570387
3	1: 2.664813	2: 2.355774	3: 2.706889	4: 0.4235811	5: 0.3162653	6: 0.4572154
3	1: 2.656384	2: 2.633445	3: 2.432497	4: 0.4622566	5: 0.4264065	6: 0.3130925
3	1: 2.655165	2: 2.632166	3: 2.429615	4: 0.4624754	5: 0.4253971	6: 0.3127446
3	1: 2.654612	2: 2.631246	3: 2.428245	4: 0.4614996	5: 0.425031	6: 0.3136229
3	1: 2.652956	2: 2.632832	3: 2.427749	4: 0.461677	5: 0.4251734	6: 0.3130102
3	1: 2.657572	2: 2.633795	3: 2.428521	4: 0.4629409	5: 0.4254745	6: 0.3132823
3	1: 2.657873	2: 2.633445	3: 2.428229	4: 0.4620567	5: 0.4247509	6: 0.313193
3	1: 2.659496	2: 2.635507	3: 2.429089	4: 0.4634762	5: 0.42552	6: 0.3135886
3	1: 2.658673	2: 2.632818	3: 2.42757	4: 0.4618028	5: 0.4245296	6: 0.3128626
3	1: 2.661912	2: 2.636072	3: 2.430581	4: 0.4635791	5: 0.4258782	6: 0.3140867
3	1: 2.662791	2: 2.637258	3: 2.431277	4: 0.463476	5: 0.4252959	6: 0.3137005
3	1: 1.357003	2: 1.929112	3: 2.677621	4: 0.2826269	5: 0.3186511	6: 0.5535332
3	1: 1.354794	2: 1.926796	3: 2.678251	4: 0.2823911	5: 0.318702	6: 0.5531879

3	1: 1.698297	2: 1.925821	3: 2.678703	4: 0.281425	5: 0.3189677	6: 0.552309
3	1: 1.35424	2: 1.924621	3: 2.678444	4: 0.2811888	5: 0.3189747	6: 0.5515713
3	1: 1.9525	2: 1.925309	3: 2.680472	4: 0.2804667	5: 0.3207786	6: 0.5542581
3	1: 1.943104	2: 1.906913	3: 2.680069	4: 0.2846058	5: 0.3163432	6: 0.5570244
3	1: 1.901354	2: 1.906907	3: 2.680907	4: 0.2840009	5: 0.3166821	6: 0.5566862
3	1: 1.937755	2: 1.905722	3: 2.680084	4: 0.2834245	5: 0.3164	6: 0.5560867
3	1: 1.938035	2: 1.906907	3: 2.681655	4: 0.2825291	5: 0.3174368	6: 0.5559007
3	1: 1.936823	2: 1.906577	3: 2.681901	4: 0.2817123	5: 0.3180473	6: 0.5563063
3	1: 1.947697	2: 2.602883	3: 1.992684	4: 0.3178189	5: 0.5436338	6: 0.2663363
3	1: 1.947218	2: 2.605506	3: 1.990721	4: 0.3195043	5: 0.5450477	6: 0.2652533
3	1: 1.915162	2: 2.605071	3: 1.991196	4: 0.3194352	5: 0.5448654	6: 0.2651909
3	1: 1.909033	2: 2.606556	3: 1.991803	4: 0.3201001	5: 0.5451954	6: 0.2649039
3	1: 1.911644	2: 2.604775	3: 1.989579	4: 0.3198528	5: 0.5443847	6: 0.2643039
3	1: 1.503935	2: 2.606694	3: 1.993159	4: 0.3201847	5: 0.5446446	6: 0.2649194
3	1: 1.351323	2: 2.607838	3: 1.991863	4: 0.3208329	5: 0.5451593	6: 0.2643603
3	1: 1.350992	2: 2.607679	3: 1.993737	4: 0.3196678	5: 0.544442	6: 0.2654287
3	1: 1.35142	2: 2.609432	3: 1.996179	4: 0.3199884	5: 0.5437507	6: 0.264806
3	1: 1.351808	2: 2.609183	3: 1.995304	4: 0.3200443	5: 0.5435389	6: 0.2643298
3	1: 2.623542	2: 1.905637	3: 2.02183	4: 0.5378798	5: 0.2486431	6: 0.3315674
3	1: 2.624578	2: 1.905428	3: 2.019328	4: 0.5426109	5: 0.2467208	6: 0.334955
3	1: 2.625546	2: 1.905926	3: 2.020507	4: 0.5407114	5: 0.2476339	6: 0.3335177
3	1: 2.62558	2: 1.905861	3: 2.020971	4: 0.5396098	5: 0.2478058	6: 0.3327723
3	1: 2.625347	2: 1.905757	3: 2.021078	4: 0.538861	5: 0.2482942	6: 0.3323872
3	1: 2.624219	2: 1.90531	3: 2.020568	4: 0.5402007	5: 0.2479011	6: 0.3327829
3	1: 2.623235	2: 1.904244	3: 2.017929	4: 0.5410335	5: 0.2469442	6: 0.3338147
3	1: 2.624116	2: 1.905655	3: 2.019374	4: 0.5409548	5: 0.2470299	6: 0.3336209
3	1: 2.624899	2: 1.905651	3: 2.019904	4: 0.539458	5: 0.2471872	6: 0.3329566
3	1: 2.623586	2: 1.905292	3: 2.020602	4: 0.5393709	5: 0.2478254	6: 0.3326539
3	1: 2.864564	2: 2.62464	3: 2.698914	4: 0.5095592	5: 0.3890364	6: 0.4206336
3	1: 2.86796	2: 2.625952	3: 2.700519	4: 0.5088057	5: 0.3883264	6: 0.4194417
3	1: 2.867471	2: 2.625566	3: 2.701997	4: 0.5099968	5: 0.3896588	6: 0.4201935
3	1: 2.865891	2: 2.623774	3: 2.699962	4: 0.5078855	5: 0.3881325	6: 0.4188586
3	1: 2.865511	2: 2.622754	3: 2.698978	4: 0.5087093	5: 0.3883677	6: 0.420021
3	1: 2.864275	2: 2.621032	3: 2.69568	4: 0.5063153	5: 0.3873912	6: 0.4181687

3	1: 2.865965	2: 2.623484	3: 2.698964	4: 0.5083985	5: 0.3885746	6: 0.4194363
3	1: 2.868222	2: 2.623395	3: 2.6993	4: 0.5076109	5: 0.3881262	6: 0.4194338
3	1: 2.867028	2: 2.624424	3: 2.699529	4: 0.5087548	5: 0.3886549	6: 0.4191809
3	1: 2.867276	2: 2.623892	3: 2.699292	4: 0.5081728	5: 0.3885395	6: 0.4182866
3	1: 2.656758	2: 2.864797	3: 2.690047	4: 0.4284313	5: 0.5265988	6: 0.3723344
3	1: 2.654877	2: 2.866116	3: 2.689667	4: 0.4286316	5: 0.5270982	6: 0.3723149
3	1: 2.654832	2: 2.867328	3: 2.689974	4: 0.4290054	5: 0.5270979	6: 0.3718573
3	1: 2.656101	2: 2.870345	3: 2.691667	4: 0.4294506	5: 0.5280842	6: 0.3723471
3	1: 2.653295	2: 2.869588	3: 2.688331	4: 0.4288544	5: 0.5267005	6: 0.3715188
3	1: 2.653254	2: 2.87197	3: 2.691434	4: 0.4301922	5: 0.5283831	6: 0.3707819
3	1: 2.652662	2: 2.870859	3: 2.690352	4: 0.4299784	5: 0.5280883	6: 0.3707674
3	1: 2.651539	2: 2.871018	3: 2.689484	4: 0.4296927	5: 0.5276254	6: 0.3698772
3	1: 2.654806	2: 2.872909	3: 2.690702	4: 0.4305368	5: 0.5286586	6: 0.3717536
3	1: 2.654914	2: 2.873268	3: 2.692364	4: 0.4293341	5: 0.5271548	6: 0.3713567
3	1: 2.648912	2: 2.633086	3: 2.957096	4: 0.3772411	5: 0.4365781	6: 0.5122814
3	1: 2.647305	2: 2.630706	3: 2.954706	4: 0.3758935	5: 0.4342194	6: 0.5103551
3	1: 2.649348	2: 2.633682	3: 2.958364	4: 0.3768349	5: 0.4369961	6: 0.5124472
3	1: 2.650172	2: 2.634649	3: 2.959671	4: 0.3763519	5: 0.436359	6: 0.5120184
3	1: 2.648131	2: 2.63396	3: 2.957187	4: 0.3772958	5: 0.4354494	6: 0.511977
3	1: 2.64882	2: 2.634295	3: 2.958115	4: 0.376855	5: 0.4357871	6: 0.5120213
3	1: 2.650193	2: 2.634819	3: 2.958983	4: 0.3768061	5: 0.435817	6: 0.5131088
3	1: 2.648528	2: 2.633967	3: 2.958509	4: 0.3768621	5: 0.4351365	6: 0.5120979
3	1: 2.649956	2: 2.633822	3: 2.95911	4: 0.377046	5: 0.4353712	6: 0.5127382
3	1: 2.651077	2: 2.636139	3: 2.961948	4: 0.3777254	5: 0.4363745	6: 0.5142132
3	1: 2.882632	2: 2.870531	3: 2.698823	4: 0.5260387	5: 0.4939467	6: 0.3710006
3	1: 2.885059	2: 2.874369	3: 2.701392	4: 0.5291751	5: 0.4967764	6: 0.3720371
3	1: 2.88563	2: 2.87414	3: 2.699029	4: 0.5278365	5: 0.4955971	6: 0.3715217
3	1: 2.888399	2: 2.877364	3: 2.700872	4: 0.5286131	5: 0.495825	6: 0.3723196
3	1: 2.891679	2: 2.880068	3: 2.702057	4: 0.5302366	5: 0.4969534	6: 0.3731592
3	1: 2.890955	2: 2.880229	3: 2.701598	4: 0.5281008	5: 0.4957254	6: 0.3720586
3	1: 2.892906	2: 2.880637	3: 2.70209	4: 0.5303913	5: 0.497237	6: 0.3730648
3	1: 2.891149	2: 2.880119	3: 2.700509	4: 0.5287099	5: 0.4963061	6: 0.3726102
3	1: 2.892821	2: 2.881511	3: 2.701682	4: 0.5289413	5: 0.4963882	6: 0.3730666
3	1: 2.891544	2: 2.879307	3: 2.700219	4: 0.5295868	5: 0.4965737	6: 0.3733252

3	1: 2.863127	2: 2.637574	3: 2.951216	4: 0.4743905	5: 0.3906479	6: 0.5204321
3	1: 2.840272	2: 2.635505	3: 2.952624	4: 0.4747085	5: 0.391536	6: 0.5204166
3	1: 2.85805	2: 2.636624	3: 2.955042	4: 0.4743297	5: 0.3908978	6: 0.5197293
3	1: 2.857537	2: 2.636371	3: 2.954668	4: 0.4745767	5: 0.3918271	6: 0.5198101
3	1: 2.856882	2: 2.636773	3: 2.956551	4: 0.4740886	5: 0.3924679	6: 0.5206956
3	1: 2.859195	2: 2.636991	3: 2.958117	4: 0.4734753	5: 0.3917514	6: 0.5211533
3	1: 2.861154	2: 2.639406	3: 2.960854	4: 0.4758182	5: 0.393251	6: 0.5237657
3	1: 2.860168	2: 2.637427	3: 2.959816	4: 0.4744613	5: 0.3923485	6: 0.5218581
3	1: 2.875678	2: 2.637415	3: 2.960976	4: 0.4747766	5: 0.3930194	6: 0.5211673
3	1: 2.875127	2: 2.637157	3: 2.960501	4: 0.4751707	5: 0.3933928	6: 0.5224262
3	1: 2.683471	2: 2.868088	3: 2.954926	4: 0.3973749	5: 0.5296357	6: 0.4723327
3	1: 2.685974	2: 2.870574	3: 2.956132	4: 0.3986473	5: 0.5312074	6: 0.4737445
3	1: 2.682843	2: 2.870628	3: 2.953648	4: 0.3984418	5: 0.5311017	6: 0.4727873
3	1: 2.684228	2: 2.872305	3: 2.954824	4: 0.3982594	5: 0.529968	6: 0.4728418
3	1: 2.683956	2: 2.871312	3: 2.952433	4: 0.3985905	5: 0.531103	6: 0.4736171
3	1: 2.681305	2: 2.871129	3: 2.953475	4: 0.3975068	5: 0.531044	6: 0.4730635
3	1: 2.681652	2: 2.872568	3: 2.954075	4: 0.396976	5: 0.5309119	6: 0.4727216
3	1: 2.68076	2: 2.869696	3: 2.952789	4: 0.3978765	5: 0.5313447	6: 0.4733026
3	1: 2.681603	2: 2.872939	3: 2.95502	4: 0.3980276	5: 0.5318675	6: 0.4731025
3	1: 2.682361	2: 2.871972	3: 2.954717	4: 0.3977194	5: 0.5305052	6: 0.4730106
3	1: 2.866166	2: 2.343634	3: 2.705462	4: 0.5133088	5: 0.2792532	6: 0.4733805
3	1: 2.867451	2: 2.343673	3: 2.704964	4: 0.5133116	5: 0.2791125	6: 0.4735139
3	1: 2.869396	2: 2.344887	3: 2.707598	4: 0.5145962	5: 0.2793732	6: 0.4745388
3	1: 2.87293	2: 2.346549	3: 2.709246	4: 0.5147081	5: 0.2793009	6: 0.4747439
3	1: 2.874104	2: 2.34637	3: 2.710391	4: 0.5163824	5: 0.2794495	6: 0.4764359
3	1: 2.87354	2: 2.345845	3: 2.708416	4: 0.514616	5: 0.2791101	6: 0.4744444
3	1: 2.874045	2: 2.345329	3: 2.708219	4: 0.5156051	5: 0.2794685	6: 0.4759558
3	1: 2.873209	2: 2.345315	3: 2.708171	4: 0.5145346	5: 0.2791449	6: 0.4745008
3	1: 2.873578	2: 2.345472	3: 2.709234	4: 0.515152	5: 0.279458	6: 0.4754029
3	1: 2.875236	2: 2.346485	3: 2.710647	4: 0.5159681	5: 0.279255	6: 0.4760323
3	1: 2.866533	2: 2.627301	3: 2.426344	4: 0.5530134	5: 0.4041961	6: 0.3268193
3	1: 2.863009	2: 2.627897	3: 2.427311	4: 0.5542271	5: 0.4060551	6: 0.3263036
3	1: 2.862391	2: 2.628765	3: 2.427224	4: 0.5530258	5: 0.4051556	6: 0.325681
3	1: 2.858348	2: 2.624883	3: 2.42293	4: 0.5527237	5: 0.4037808	6: 0.3264718

3	1: 2.888765	2: 2.628337	3: 2.425885	4: 0.5528683	5: 0.4045763	6: 0.3266591
3	1: 2.891084	2: 2.628305	3: 2.426714	4: 0.5529187	5: 0.4043931	6: 0.3272386
3	1: 2.890825	2: 2.628883	3: 2.426495	4: 0.5537398	5: 0.4045748	6: 0.327342
3	1: 2.891829	2: 2.630133	3: 2.427149	4: 0.553249	5: 0.4041457	6: 0.3275972
3	1: 2.891221	2: 2.62898	3: 2.426523	4: 0.5545524	5: 0.4047952	6: 0.3272193
3	1: 2.890232	2: 2.628094	3: 2.424039	4: 0.5529396	5: 0.4036552	6: 0.3265032
3	1: 2.398978	2: 2.869986	3: 2.681575	4: 0.3366791	5: 0.5630464	6: 0.3930982
3	1: 2.398624	2: 2.870721	3: 2.680924	4: 0.336545	5: 0.5630998	6: 0.3936848
3	1: 2.397643	2: 2.871122	3: 2.681896	4: 0.336612	5: 0.5628562	6: 0.3938675
3	1: 2.39782	2: 2.872357	3: 2.682282	4: 0.3365579	5: 0.5627389	6: 0.3937206
3	1: 2.397249	2: 2.87174	3: 2.682665	4: 0.3375051	5: 0.5632329	6: 0.3934723
3	1: 2.398203	2: 2.87178	3: 2.683265	4: 0.3366189	5: 0.5627291	6: 0.3937706
3	1: 2.395648	2: 2.870402	3: 2.682097	4: 0.3372375	5: 0.5634876	6: 0.393311
3	1: 2.394087	2: 2.868927	3: 2.681887	4: 0.3356911	5: 0.5611915	6: 0.3926138
3	1: 2.39627	2: 2.872326	3: 2.683399	4: 0.3372835	5: 0.5637803	6: 0.3940929
3	1: 2.395832	2: 2.871829	3: 2.682345	4: 0.3366885	5: 0.5614722	6: 0.3930647
3	1: 2.654056	2: 2.873251	3: 2.418747	4: 0.4823615	5: 0.5261681	6: 0.2742906
3	1: 2.655846	2: 2.873909	3: 2.419844	4: 0.4835795	5: 0.5269113	6: 0.2745757
3	1: 2.655558	2: 2.874553	3: 2.418647	4: 0.4823426	5: 0.5258783	6: 0.2745936
3	1: 2.654364	2: 2.87412	3: 2.419393	4: 0.4842559	5: 0.5274884	6: 0.2739938
3	1: 2.653712	2: 2.872604	3: 2.41715	4: 0.4817698	5: 0.5255313	6: 0.2736923
3	1: 2.653342	2: 2.872216	3: 2.41744	4: 0.4833592	5: 0.5266313	6: 0.2742521
3	1: 2.651094	2: 2.87055	3: 2.416067	4: 0.4816383	5: 0.5250584	6: 0.2731858
3	1: 2.653122	2: 2.872037	3: 2.41627	4: 0.4835878	5: 0.526541	6: 0.2734687
3	1: 2.653691	2: 2.873388	3: 2.417873	4: 0.4831358	5: 0.5263587	6: 0.2733474
3	1: 2.652627	2: 2.872692	3: 2.417524	4: 0.4833191	5: 0.5267664	6: 0.2730185
3	1: 2.378637	2: 2.651754	3: 2.948166	4: 0.272337	5: 0.4952164	6: 0.5170674
3	1: 2.379217	2: 2.654273	3: 2.951499	4: 0.2725792	5: 0.4970479	6: 0.5175106
3	1: 2.377339	2: 2.654188	3: 2.953896	4: 0.2726176	5: 0.4962381	6: 0.5170214
3	1: 2.377048	2: 2.653993	3: 2.953519	4: 0.2731294	5: 0.4967284	6: 0.5176833
3	1: 2.378433	2: 2.654414	3: 2.953461	4: 0.2728246	5: 0.4947698	6: 0.5174299
3	1: 2.377584	2: 2.65427	3: 2.955154	4: 0.2733927	5: 0.4967357	6: 0.5181301
3	1: 2.377208	2: 2.655399	3: 2.955883	4: 0.2737368	5: 0.4947184	6: 0.5168238
3	1: 2.374901	2: 2.653008	3: 2.953485	4: 0.2732206	5: 0.4952434	6: 0.5169247



3	1: 2.376546	2: 2.653129	3: 2.95501	4: 0.2733424	5: 0.4947922	6: 0.5175055
3	1: 2.376865	2: 2.654395	3: 2.957332	4: 0.2733145	5: 0.4954093	6: 0.5176221
3	1: 2.659465	2: 2.352135	3: 2.970046	4: 0.4076051	5: 0.3305309	6: 0.5618654
3	1: 2.66212	2: 2.353067	3: 2.972605	4: 0.4073033	5: 0.3301847	6: 0.5610484
3	1: 2.659752	2: 2.352185	3: 2.971217	4: 0.4069714	5: 0.331507	6: 0.5616516
3	1: 2.658792	2: 2.350035	3: 2.97051	4: 0.4059177	5: 0.3310979	6: 0.56022
3	1: 2.65676	2: 2.351611	3: 2.972427	4: 0.4064358	5: 0.3316989	6: 0.5605394
3	1: 2.653714	2: 2.352171	3: 2.97266	4: 0.4066124	5: 0.3317941	6: 0.5610365
3	1: 2.651886	2: 2.351112	3: 2.971762	4: 0.40647	5: 0.3312326	6: 0.5602003
3	1: 2.64985	2: 2.35141	3: 2.972881	4: 0.4061022	5: 0.3320981	6: 0.5610026
3	1: 2.650616	2: 2.351339	3: 2.972718	4: 0.4063696	5: 0.331387	6: 0.5600172
3	1: 2.648766	2: 2.350154	3: 2.971105	4: 0.4064279	5: 0.3321843	6: 0.5609437
3	1: 2.883925	2: 1.934498	3: 2.708467	4: 0.5767817	5: 0.112336	6: 0.5889852
3	1: 2.885501	2: 1.933236	3: 2.708519	4: 0.5772637	5: 0.1121746	6: 0.5895937
3	1: 2.88595	2: 1.932796	3: 2.710216	4: 0.5769671	5: 0.1126937	6: 0.5888161
3	1: 2.884518	2: 1.930957	3: 2.709268	4: 0.5759599	5: 0.1131795	6: 0.5880892
3	1: 2.886574	2: 1.931522	3: 2.709149	4: 0.5772798	5: 0.1121606	6: 0.5895276
3	1: 2.887566	2: 1.931854	3: 2.71021	4: 0.5776809	5: 0.1120085	6: 0.590033
3	1: 2.887958	2: 1.931478	3: 2.710571	4: 0.5772751	5: 0.112357	6: 0.5899563
3	1: 2.885036	2: 1.929463	3: 2.708111	4: 0.5765475	5: 0.1124928	6: 0.5883664
3	1: 2.886028	2: 1.92923	3: 2.707385	4: 0.5770617	5: 0.1118947	6: 0.5896868
3	1: 2.885723	2: 1.929338	3: 2.707751	4: 0.5770201	5: 0.1123167	6: 0.5891762
3	1: 2.890272	2: 2.619056	3: 2.030071	4: 0.6403416	5: 0.4739701	6: 0.2119101
3	1: 2.891225	2: 2.62052	3: 2.029035	4: 0.6402839	5: 0.4739807	6: 0.2122906
3	1: 2.894497	2: 2.62483	3: 2.029982	4: 0.642289	5: 0.4748568	6: 0.2126706
3	1: 2.894446	2: 2.62523	3: 2.029185	4: 0.6424979	5: 0.4748758	6: 0.212791
3	1: 2.894873	2: 2.625988	3: 2.029104	4: 0.6425535	5: 0.4744673	6: 0.2133469
3	1: 2.894258	2: 2.625977	3: 2.028267	4: 0.6423099	5: 0.4748404	6: 0.2130094
3	1: 2.893171	2: 2.623388	3: 2.028751	4: 0.6414747	5: 0.4733674	6: 0.2136012
3	1: 2.893519	2: 2.623507	3: 2.028121	4: 0.6415043	5: 0.4730741	6: 0.2140291
3	1: 2.894006	2: 2.623249	3: 2.028337	4: 0.6410341	5: 0.4728454	6: 0.214055
3	1: 2.892982	2: 2.623817	3: 2.027896	4: 0.641091	5: 0.4730057	6: 0.2137039
3	1: 1.993749	2: 2.875222	3: 2.649975	4: 0.2323019	5: 0.6592163	6: 0.4557509
3	1: 1.995466	2: 2.877611	3: 2.638955	4: 0.2363662	5: 0.6632178	6: 0.453104

3	1: 1.993528	2: 2.878207	3: 2.58154	4: 0.2525107	5: 0.6703488	6: 0.4400139
3	1: 1.992357	2: 2.877546	3: 2.599992	4: 0.2473798	5: 0.6674857	6: 0.4435414
3	1: 1.992064	2: 2.878492	3: 2.593536	4: 0.249251	5: 0.6695724	6: 0.4420643
3	1: 1.992263	2: 2.879892	3: 2.616803	4: 0.2421113	5: 0.6657251	6: 0.4468828
3	1: 1.991842	2: 2.879056	3: 2.626873	4: 0.2389827	5: 0.668257	6: 0.4516497
3	1: 1.965047	2: 2.87867	3: 2.623157	4: 0.2406563	5: 0.6684191	6: 0.4499035
3	1: 1.963098	2: 2.879515	3: 2.629265	4: 0.2391034	5: 0.6674153	6: 0.4514369
3	1: 1.959409	2: 2.876758	3: 2.628274	4: 0.2392588	5: 0.6675358	6: 0.4511543
3	1: 2.644664	2: 2.872074	3: 2.030863	4: 0.5897823	5: 0.5800314	6: 0.1214538
3	1: 2.645571	2: 2.870898	3: 2.028227	4: 0.5883817	5: 0.5783902	6: 0.122568
3	1: 2.645386	2: 2.870283	3: 2.026275	4: 0.5880713	5: 0.5785228	6: 0.122212
3	1: 2.649341	2: 2.870957	3: 2.02536	4: 0.5880567	5: 0.5777234	6: 0.1231346
3	1: 2.646765	2: 2.869272	3: 2.023882	4: 0.5879145	5: 0.5779874	6: 0.1223316
3	1: 2.649357	2: 2.870443	3: 2.024524	4: 0.5881719	5: 0.5777479	6: 0.1231432
3	1: 2.648386	2: 2.870014	3: 2.023373	4: 0.5877267	5: 0.5774469	6: 0.1228989
3	1: 2.649368	2: 2.870781	3: 2.023734	4: 0.5882636	5: 0.5779365	6: 0.1228733
3	1: 2.647758	2: 2.869171	3: 2.022265	4: 0.5884052	5: 0.5780859	6: 0.1218917
3	1: 2.648297	2: 2.869462	3: 2.02218	4: 0.5882497	5: 0.5776529	6: 0.1227032
3	1: 1.992376	2: 2.636828	3: 2.952657	4: 0.1139055	5: 0.5977876	6: 0.578835
3	1: 1.990301	2: 2.636588	3: 2.953013	4: 0.1144166	5: 0.5970471	6: 0.5782852
3	1: 1.987934	2: 2.635479	3: 2.95166	4: 0.1140213	5: 0.5973573	6: 0.5780775
3	1: 1.989388	2: 2.637558	3: 2.954909	4: 0.1139374	5: 0.5983519	6: 0.5790558
3	1: 1.987847	2: 2.638594	3: 2.957004	4: 0.1143629	5: 0.5991067	6: 0.5788473
3	1: 1.988474	2: 2.639386	3: 2.959173	4: 0.1146522	5: 0.5999875	6: 0.5790513
3	1: 1.985026	2: 2.637057	3: 2.956445	4: 0.1149476	5: 0.598698	6: 0.578118
3	1: 1.546678	2: 2.637809	3: 2.957926	4: 0.1147467	5: 0.5992159	6: 0.5785324
3	1: 1.365259	2: 2.635328	3: 2.956406	4: 0.114885	5: 0.5980103	6: 0.577589
3	1: 1.366443	2: 2.635656	3: 2.956289	4: 0.1144264	5: 0.5983336	6: 0.5779234
3	1: 2.608702	2: 1.919132	3: 2.960456	4: 0.4818839	5: 0.2180935	6: 0.6610849
3	1: 2.610705	2: 1.918984	3: 2.96278	4: 0.4817012	5: 0.2185244	6: 0.6616911
3	1: 2.611045	2: 1.917579	3: 2.961352	4: 0.4817474	5: 0.2182568	6: 0.6614475
3	1: 2.61202	2: 1.917361	3: 2.96221	4: 0.4816396	5: 0.218438	6: 0.6615601
3	1: 2.611633	2: 1.916762	3: 2.962538	4: 0.4810615	5: 0.2189182	6: 0.6614998
3	1: 2.612892	2: 1.916134	3: 2.961255	4: 0.4813614	5: 0.2184266	6: 0.6612247

3	1: 2.612995	2: 1.916012	3: 2.963639	4: 0.4809026	5: 0.2189547	6: 0.6609439
3	1: 2.614583	2: 1.916313	3: 2.964598	4: 0.4811302	5: 0.219131	6: 0.6615496
3	1: 2.615691	2: 1.91686	3: 2.965665	4: 0.481583	5: 0.2194076	6: 0.6617692
3	1: 2.615227	2: 1.915817	3: 2.965615	4: 0.4816101	5: 0.219353	6: 0.6621651
3	1: 2.848711	2: 1.280485	3: 2.702216	4: 0.7967888	5: 0.2150265	6: 0.8381048
3	1: 2.849156	2: 1.278995	3: 2.70329	4: 0.7961278	5: 0.2148275	6: 0.8373546
3	1: 2.849147	2: 1.278123	3: 2.704254	4: 0.7945525	5: 0.2140991	6: 0.8365772
3	1: 2.849499	2: 1.277442	3: 2.703124	4: 0.7945896	5: 0.2140711	6: 0.8367594
3	1: 2.851177	2: 1.276781	3: 2.704342	4: 0.794538	5: 0.214023	6: 0.8366787
3	1: 2.853186	2: 1.276866	3: 2.705264	4: 0.7950171	5: 0.2145924	6: 0.8373585
3	1: 2.852475	2: 1.276896	3: 2.706686	4: 0.7936636	5: 0.2142278	6: 0.836391
3	1: 2.852644	2: 1.275934	3: 2.704619	4: 0.7930435	5: 0.2141149	6: 0.8360413
3	1: 2.848711	2: 1.280485	3: 2.702216	4: 0.7967888	5: 0.2150265	6: 0.8381048
3	1: 2.849156	2: 1.278995	3: 2.70329	4: 0.7961278	5: 0.2148275	6: 0.8373546
3	1: 2.864136	2: 2.606253	3: 1.371891	4: 0.8712473	5: 0.6987545	6: 0.2487019
3	1: 2.866429	2: 2.608966	3: 1.370537	4: 0.8712392	5: 0.698469	6: 0.2490513
3	1: 2.86677	2: 2.609131	3: 1.368302	4: 0.8704994	5: 0.697388	6: 0.2485641
3	1: 2.86597	2: 2.61003	3: 1.367747	4: 0.8702443	5: 0.6971658	6: 0.2485745
3	1: 2.866582	2: 2.611141	3: 1.367446	4: 0.8705531	5: 0.6976136	6: 0.2488287
3	1: 2.864277	2: 2.611344	3: 1.365555	4: 0.870666	5: 0.6982818	6: 0.2488139
3	1: 2.865711	2: 2.61291	3: 1.366348	4: 0.8703048	5: 0.6975959	6: 0.2486813
3	1: 2.864136	2: 2.606253	3: 1.371891	4: 0.8712473	5: 0.6987545	6: 0.2487019
3	1: 2.866429	2: 2.608966	3: 1.370537	4: 0.8712392	5: 0.698469	6: 0.2490513
3	1: 2.86677	2: 2.609131	3: 1.368302	4: 0.8704994	5: 0.697388	6: 0.2485641
3	1: 1.321227	2: 2.883518	3: 2.703833	4: 0.2741966	5: 0.9060237	6: 0.7203761
3	1: 1.317656	2: 2.883249	3: 2.702366	4: 0.2739349	5: 0.9050196	6: 0.7200783
3	1: 1.317224	2: 2.883495	3: 2.702592	4: 0.2738253	5: 0.9048558	6: 0.7200472
3	1: 1.315834	2: 2.883223	3: 2.702509	4: 0.2733584	5: 0.9042926	6: 0.7189843
3	1: 1.315147	2: 2.884414	3: 2.703708	4: 0.2733395	5: 0.9042258	6: 0.7188001
3	1: 1.312702	2: 2.881185	3: 2.70196	4: 0.2733005	5: 0.9030778	6: 0.7181544
3	1: 1.311774	2: 2.881931	3: 2.702526	4: 0.2728506	5: 0.9025943	6: 0.7175111
3	1: 1.31211	2: 2.883107	3: 2.702363	4: 0.2734692	5: 0.9038061	6: 0.7187272
3	1: 1.20908	2: 2.884183	3: 2.703332	4: 0.2730347	5: 0.9027422	6: 0.7179093
3	1: 0.9061481	2: 2.88311	3: 2.703058	4: 0.2731746	5: 0.9025234	6: 0.7176536

3	1: 2.652579	2: 2.85952	3: 1.36939	4: 0.8381335	5: 0.7780255	6: 0.1950746
3	1: 2.65687	2: 2.861159	3: 1.368052	4: 0.8386686	5: 0.7781405	6: 0.194978
3	1: 2.657808	2: 2.861063	3: 1.367102	4: 0.8389426	5: 0.7785074	6: 0.1954481
3	1: 2.658472	2: 2.859459	3: 1.365794	4: 0.8376881	5: 0.7771115	6: 0.194529
3	1: 2.657713	2: 2.859886	3: 1.366104	4: 0.8388114	5: 0.7790508	6: 0.1958883
3	1: 2.660226	2: 2.860165	3: 1.365506	4: 0.8375675	5: 0.7770939	6: 0.1944721
3	1: 2.658793	2: 2.860836	3: 1.364833	4: 0.8366735	5: 0.7766506	6: 0.1937523
3	1: 2.659639	2: 2.860794	3: 1.3649	4: 0.8368805	5: 0.7769428	6: 0.1943538
3	1: 2.655954	2: 2.859673	3: 1.363699	4: 0.8362921	5: 0.7762194	6: 0.1938516
3	1: 2.653132	2: 2.859238	3: 1.363754	4: 0.8364016	5: 0.7767048	6: 0.1942116
3	1: 1.311705	2: 2.638706	3: 2.929888	4: 0.2176978	5: 0.8540278	6: 0.7906227
3	1: 1.310587	2: 2.639184	3: 2.932162	4: 0.2174692	5: 0.853768	6: 0.790264
3	1: 1.308391	2: 2.636251	3: 2.930315	4: 0.2171209	5: 0.852119	6: 0.7895299
3	1: 1.308665	2: 2.638446	3: 2.932846	4: 0.2168626	5: 0.8528979	6: 0.7893712
3	1: 1.308709	2: 2.638927	3: 2.93309	4: 0.2174677	5: 0.8537473	6: 0.7898716
3	1: 1.306904	2: 2.637911	3: 2.932447	4: 0.2166412	5: 0.8523822	6: 0.7884687
3	1: 1.307375	2: 2.638789	3: 2.934768	4: 0.2166576	5: 0.8524045	6: 0.788917
3	1: 1.307237	2: 2.639673	3: 2.937421	4: 0.217321	5: 0.853738	6: 0.7897837
3	1: 1.306667	2: 2.639842	3: 2.937515	4: 0.2173637	5: 0.853578	6: 0.7893758
3	1: 1.306779	2: 2.639947	3: 2.937496	4: 0.2174327	5: 0.8538302	6: 0.7899071
3	1: 2.628277	2: 1.286088	3: 2.963191	4: 0.731343	5: 0.2828016	6: 0.8906009
3	1: 2.629144	2: 1.283852	3: 2.962597	4: 0.7311126	5: 0.2823225	6: 0.8899908
3	1: 2.629211	2: 1.281597	3: 2.961973	4: 0.7300448	5: 0.2822807	6: 0.8890498
3	1: 2.633998	2: 1.281446	3: 2.963846	4: 0.7317463	5: 0.2834898	6: 0.8912933
3	1: 2.633253	2: 1.280379	3: 2.962758	4: 0.7309988	5: 0.2826631	6: 0.8897895
3	1: 2.635467	2: 1.280312	3: 2.962992	4: 0.7319694	5: 0.2834088	6: 0.8910705
3	1: 2.636501	2: 1.279694	3: 2.963483	4: 0.7320888	5: 0.2833346	6: 0.8907039
3	1: 2.633997	2: 1.277626	3: 2.961228	4: 0.7307955	5: 0.2828028	6: 0.8887949
3	1: 2.637354	2: 1.27879	3: 2.964189	4: 0.7313442	5: 0.283185	6: 0.8897198
3	1: 2.638426	2: 1.279033	3: 2.965485	4: 0.7305727	5: 0.282838	6: 0.8891887
3	1: 2.869298	2: 2.87518	3: 2.42939	4: 0.579487	5: 0.5102055	6: 0.193961
3	1: 2.867915	2: 2.874105	3: 2.426294	4: 0.5778185	5: 0.5094989	6: 0.193475
3	1: 2.870855	2: 2.875861	3: 2.426853	4: 0.5785119	5: 0.5096981	6: 0.1938725
3	1: 2.870834	2: 2.876613	3: 2.426518	4: 0.5788447	5: 0.509951	6: 0.1939896

3	1: 2.869227	2: 2.876194	3: 2.426574	4: 0.5786881	5: 0.5103472	6: 0.1932336
3	1: 2.87033	2: 2.875032	3: 2.425338	4: 0.5783918	5: 0.5098706	6: 0.193518
3	1: 2.877224	2: 2.878229	3: 2.42735	4: 0.5792007	5: 0.5099892	6: 0.1939197
3	1: 2.874111	2: 2.876889	3: 2.424884	4: 0.5794499	5: 0.5096252	6: 0.1939721
3	1: 2.874462	2: 2.87637	3: 2.425447	4: 0.5779349	5: 0.5085555	6: 0.1937221
3	1: 2.8761	2: 2.877447	3: 2.42513	4: 0.5800811	5: 0.5093909	6: 0.194473
3	1: 2.869234	2: 2.352456	3: 2.945788	4: 0.490329	5: 0.2874164	6: 0.3982192
3	1: 2.866159	2: 2.350295	3: 2.947839	4: 0.4906842	5: 0.2879876	6: 0.398044
3	1: 2.867807	2: 2.351009	3: 2.950583	4: 0.4902012	5: 0.2878592	6: 0.3971742
3	1: 2.868864	2: 2.350649	3: 2.95187	4: 0.4918291	5: 0.288295	6: 0.3992275
3	1: 2.86902	2: 2.35023	3: 2.952507	4: 0.4906214	5: 0.2876421	6: 0.3976758
3	1: 2.869033	2: 2.348606	3: 2.952369	4: 0.4914208	5: 0.2879786	6: 0.3987887
3	1: 2.867786	2: 2.347903	3: 2.953589	4: 0.4902968	5: 0.2879993	6: 0.3974267
3	1: 2.869832	2: 2.34831	3: 2.954478	4: 0.4907328	5: 0.2885434	6: 0.398367
3	1: 2.870896	2: 2.348327	3: 2.953447	4: 0.490928	5: 0.2879644	6: 0.3989019
3	1: 2.784194	2: 2.348328	3: 2.953886	4: 0.4910822	5: 0.2880568	6: 0.3985613
3	1: 2.39712	2: 2.875784	3: 2.942779	4: 0.287734	5: 0.5906616	6: 0.3429033
3	1: 2.395504	2: 2.877579	3: 2.942599	4: 0.2882628	5: 0.5907407	6: 0.3423982
3	1: 2.393123	2: 2.876936	3: 2.941219	4: 0.2879145	5: 0.5901232	6: 0.3422651
3	1: 2.392529	2: 2.879073	3: 2.942968	4: 0.288268	5: 0.5911543	6: 0.3423441
3	1: 2.392622	2: 2.88029	3: 2.944112	4: 0.2881045	5: 0.590566	6: 0.3425158
3	1: 2.390948	2: 2.879273	3: 2.942909	4: 0.2880478	5: 0.5910791	6: 0.3427403
3	1: 2.390175	2: 2.879375	3: 2.943025	4: 0.2868535	5: 0.5901292	6: 0.3424285
3	1: 2.389598	2: 2.879089	3: 2.942558	4: 0.2875883	5: 0.591306	6: 0.3430271
3	1: 2.389092	2: 2.880811	3: 2.943743	4: 0.2869338	5: 0.5901502	6: 0.3422285
3	1: 2.387308	2: 2.879532	3: 2.941964	4: 0.2877414	5: 0.5916591	6: 0.3424553
3	1: 2.898855	2: 2.877991	3: 2.026266	4: 0.6959101	5: 0.5835677	6: 0.1027822
3	1: 2.900534	2: 2.879656	3: 2.023386	4: 0.695485	5: 0.5830539	6: 0.1032374
3	1: 2.902347	2: 2.878625	3: 2.021208	4: 0.6945503	5: 0.5824337	6: 0.1034561
3	1: 2.902662	2: 2.877813	3: 2.019533	4: 0.6946336	5: 0.5817826	6: 0.1036982
3	1: 2.904124	2: 2.877234	3: 2.018381	4: 0.6940526	5: 0.5810437	6: 0.1040995
3	1: 2.902341	2: 2.877679	3: 2.017839	4: 0.6948185	5: 0.5821616	6: 0.1036352
3	1: 2.902855	2: 2.876442	3: 2.015309	4: 0.693703	5: 0.5803308	6: 0.1043272
3	1: 2.902428	2: 2.876148	3: 2.015754	4: 0.6937998	5: 0.5808678	6: 0.1041285

3	1: 2.902549	2: 2.877033	3: 2.015444	4: 0.6944572	5: 0.5814266	6: 0.1040123
3	1: 2.899304	2: 2.872547	3: 2.012637	4: 0.6920879	5: 0.5800074	6: 0.1036828
3	1: 2.897877	2: 1.926304	3: 2.95819	4: 0.574745	5: 0.1576823	6: 0.4866651
3	1: 2.89883	2: 1.923764	3: 2.956994	4: 0.5753022	5: 0.1576618	6: 0.4871841
3	1: 2.897318	2: 1.923272	3: 2.958913	4: 0.5746293	5: 0.1577675	6: 0.4869962
3	1: 2.896374	2: 1.921822	3: 2.959247	4: 0.5737883	5: 0.1583673	6: 0.4864729
3	1: 2.896022	2: 1.921167	3: 2.959394	4: 0.5736904	5: 0.1580611	6: 0.4858303
3	1: 2.893449	2: 1.919452	3: 2.958516	4: 0.5728073	5: 0.157512	6: 0.4849752
3	1: 2.893145	2: 1.91866	3: 2.958015	4: 0.5728554	5: 0.1574137	6: 0.4851336
3	1: 2.894341	2: 1.919098	3: 2.959013	4: 0.572764	5: 0.1574303	6: 0.4849042
3	1: 2.894859	2: 1.918063	3: 2.957782	4: 0.573199	5: 0.1575601	6: 0.4854648
3	1: 2.894386	2: 1.917979	3: 2.957599	4: 0.5734485	5: 0.1570611	6: 0.485257
3	1: 1.993388	2: 2.874621	3: 2.951811	4: 0.1565158	5: 0.7068903	6: 0.3994827
3	1: 1.989449	2: 2.875722	3: 2.952027	4: 0.1568166	5: 0.705988	6: 0.3991813
3	1: 1.990714	2: 2.881382	3: 2.955542	4: 0.1574917	5: 0.7071833	6: 0.399677
3	1: 1.989603	2: 2.881707	3: 2.955533	4: 0.1570817	5: 0.7073283	6: 0.3996201
3	1: 1.990178	2: 2.886694	3: 2.959225	4: 0.157963	5: 0.7082874	6: 0.3997458
3	1: 1.989892	2: 2.887299	3: 2.95906	4: 0.1574524	5: 0.7088872	6: 0.4004959
3	1: 1.987636	2: 2.88749	3: 2.960347	4: 0.1583788	5: 0.7093159	6: 0.3997871
3	1: 1.986993	2: 2.887277	3: 2.960588	4: 0.15818	5: 0.7090072	6: 0.3998096
3	1: 1.986566	2: 2.886937	3: 2.959363	4: 0.1577422	5: 0.7081607	6: 0.3997358
3	1: 1.986981	2: 2.886266	3: 2.958389	4: 0.1580077	5: 0.707935	6: 0.3995723
3	1: 2.894386	2: 2.87306	3: 1.370604	4: 0.9429961	5: 0.8098531	6: 0.1822205
3	1: 2.896339	2: 2.872651	3: 1.367719	4: 0.9428183	5: 0.8098455	6: 0.1822807
3	1: 2.895823	2: 2.869182	3: 1.364775	4: 0.9402191	5: 0.807332	6: 0.1814316
3	1: 2.902101	2: 2.871469	3: 1.364429	4: 0.9418942	5: 0.8073709	6: 0.1818118
3	1: 2.901702	2: 2.872546	3: 1.363214	4: 0.9416142	5: 0.8072713	6: 0.1818187
3	1: 2.90146	2: 2.872289	3: 1.362425	4: 0.9414435	5: 0.8077422	6: 0.1820667
3	1: 2.900975	2: 2.872383	3: 1.361516	4: 0.9402315	5: 0.8068039	6: 0.1817753
3	1: 2.891164	2: 2.871726	3: 1.360891	4: 0.9397248	5: 0.8059003	6: 0.1814507
3	1: 2.894386	2: 2.87306	3: 1.370604	4: 0.9429961	5: 0.8098531	6: 0.1822205
3	1: 2.896339	2: 2.872651	3: 1.367719	4: 0.9428183	5: 0.8098455	6: 0.1822807
3	1: 2.862875	2: 1.310295	3: 2.970051	4: 0.8172097	5: 0.2823321	6: 0.6575707
3	1: 2.862191	2: 1.306149	3: 2.969894	4: 0.8166515	5: 0.2820602	6: 0.657154

3	1: 2.861213	2: 1.303973	3: 2.970037	4: 0.8158715	5: 0.2818973	6: 0.6567931
3	1: 2.862015	2: 1.302887	3: 2.972013	4: 0.8160524	5: 0.2820284	6: 0.6571088
3	1: 2.872379	2: 1.301194	3: 2.971612	4: 0.8141873	5: 0.2808872	6: 0.6556095
3	1: 2.874311	2: 1.300923	3: 2.972992	4: 0.8150353	5: 0.2815393	6: 0.6563665
3	1: 2.875653	2: 1.3014	3: 2.975021	4: 0.8150071	5: 0.2812961	6: 0.6563923
3	1: 2.873734	2: 1.299935	3: 2.973098	4: 0.8142779	5: 0.2810717	6: 0.6554931
3	1: 2.874092	2: 1.299274	3: 2.97255	4: 0.8138343	5: 0.2809078	6: 0.6550871
3	1: 2.874286	2: 1.299629	3: 2.976068	4: 0.8128911	5: 0.2803116	6: 0.6548108
3	1: 1.348948	2: 2.884112	3: 2.960971	4: 0.2733612	5: 0.9542636	6: 0.5647501
3	1: 0.9224163	2: 2.886713	3: 2.961327	4: 0.2734528	5: 0.9546964	6: 0.5650246
3	1: 0.9217279	2: 2.889416	3: 2.963029	4: 0.2737507	5: 0.9553607	6: 0.5652831
3	1: 0.921012	2: 2.890906	3: 2.963917	4: 0.273513	5: 0.9553913	6: 0.5650934
3	1: 0.9210893	2: 2.891352	3: 2.965438	4: 0.2740591	5: 0.9563518	6: 0.5663143
3	1: 0.9206495	2: 2.895095	3: 2.966468	4: 0.2740626	5: 0.9564064	6: 0.56546
3	1: 0.9195053	2: 2.893107	3: 2.964883	4: 0.2741228	5: 0.9557357	6: 0.5655521
3	1: 0.9194766	2: 2.896158	3: 2.966534	4: 0.2736887	5: 0.9552947	6: 0.5647303
3	1: 0.9184621	2: 2.894506	3: 2.965183	4: 0.2729994	5: 0.9538033	6: 0.5640372
3	1: 0.9182459	2: 2.894907	3: 2.965013	4: 0.2731076	5: 0.954013	6: 0.5640218
4	1: 2.382402	2: 2.360674	3: 2.422607	4: 0.3529491	5: 0.3529101	6: 0.341619
4	1: 2.381336	2: 2.360096	3: 2.421237	4: 0.3528076	5: 0.3531542	6: 0.3408604
4	1: 2.381771	2: 2.360703	3: 2.422024	4: 0.3526075	5: 0.3531779	6: 0.3400515
4	1: 2.382953	2: 2.360257	3: 2.42146	4: 0.3537958	5: 0.3532522	6: 0.3402935
4	1: 2.380475	2: 2.358105	3: 2.419025	4: 0.3527327	5: 0.3525339	6: 0.3370528
4	1: 2.382225	2: 2.360513	3: 2.420996	4: 0.3530358	5: 0.3531011	6: 0.3286321
4	1: 2.382402	2: 2.360674	3: 2.422607	4: 0.3529491	5: 0.3529101	6: 0.341619
4	1: 2.381336	2: 2.360096	3: 2.421237	4: 0.3528076	5: 0.3531542	6: 0.3408604
4	1: 2.381771	2: 2.360703	3: 2.422024	4: 0.3526075	5: 0.3531779	6: 0.3400515
4	1: 2.163475	2: 2.152964	3: 2.201442	4: 0.3237105	5: 0.3294426	6: 0.3284097
4	1: 2.163955	2: 2.153846	3: 2.201543	4: 0.3238132	5: 0.3290614	6: 0.3283143
4	1: 2.164716	2: 2.153646	3: 2.201396	4: 0.3238479	5: 0.3293944	6: 0.3282555
4	1: 2.164056	2: 2.153861	3: 2.202069	4: 0.324045	5: 0.3296407	6: 0.3279101
4	1: 2.164577	2: 2.154193	3: 2.202618	4: 0.3238008	5: 0.32948	6: 0.3279611
4	1: 2.165098	2: 2.154525	3: 2.203167	4: 0.3235566	5: 0.3293193	6: 0.3280121
4	1: 2.165619	2: 2.154857	3: 2.203716	4: 0.3233124	5: 0.3291586	6: 0.3280631

4	1: 2.16614	2: 2.155189	3: 2.204265	4: 0.3230682	5: 0.3289979	6: 0.3281141
4	1: 1.874775	2: 1.8553	3: 1.876041	4: 0.2871271	5: 0.2874525	6: 0.2812617
4	1: 1.875157	2: 1.855805	3: 1.875452	4: 0.2871257	5: 0.2868409	6: 0.281019
4	1: 1.875168	2: 1.855323	3: 1.875755	4: 0.2870673	5: 0.287159	6: 0.2811552
4	1: 1.874487	2: 1.855414	3: 1.875549	4: 0.2872205	5: 0.2876024	6: 0.2810761
4	1: 1.874989	2: 1.855204	3: 1.875994	4: 0.2867131	5: 0.2871328	6: 0.281181
4	1: 1.874796	2: 1.855089	3: 1.874878	4: 0.2867769	5: 0.2870515	6: 0.2810125
4	1: 1.8750663	2: 1.8549107	3: 1.8748027	4: 0.2864599	5: 0.2867113	6: 0.2810263
4	1: 1.8752208	2: 1.8547482	3: 1.8744672	4: 0.2862381	5: 0.2864359	6: 0.2809945
4	1: 1.8753753	2: 1.8545857	3: 1.8741317	4: 0.2860163	5: 0.2861604	6: 0.2809627
4	1: 1.453689	2: 1.451406	3: 1.443113	4: 0.2451649	5: 0.2501255	6: 0.2338819
4	1: 1.453575	2: 1.451258	3: 1.44291	4: 0.244853	5: 0.2500153	6: 0.2339149
4	1: 1.453321	2: 1.450869	3: 1.442793	4: 0.245033	5: 0.2499871	6: 0.2339913
4	1: 1.453478	2: 1.450875	3: 1.442797	4: 0.2450377	5: 0.2498526	6: 0.2341033
4	1: 1.454064	2: 1.451483	3: 1.443405	4: 0.2452552	5: 0.2497707	6: 0.2342472
4	1: 1.453358	2: 1.450648	3: 1.442555	4: 0.2451787	5: 0.2497242	6: 0.2340178
4	1: 1.453138	2: 1.450073	3: 1.442078	4: 0.2449135	5: 0.2495876	6: 0.2339642
4	1: 1.453841	2: 1.450904	3: 1.442799	4: 0.2448468	5: 0.2495778	6: 0.2340336
4	1: 1.454063	2: 1.45136	3: 1.443139	4: 0.2449421	5: 0.2494111	6: 0.2336491
4	1: 1.453433	2: 1.450631	3: 1.442537	4: 0.2450867	5: 0.2495748	6: 0.2339004
4	1: 1.452803	2: 1.449902	3: 1.441935	4: 0.2452313	5: 0.2497385	6: 0.2341517
4	1: 1.452173	2: 1.449173	3: 1.441333	4: 0.2453759	5: 0.2499022	6: 0.234403
4	1: 1.086678	2: 1.087319	3: 1.083799	4: 0.2310545	5: 0.2390753	6: 0.2256249
4	1: 1.086249	2: 1.086596	3: 1.08313	4: 0.2311174	5: 0.2388403	6: 0.225809
4	1: 1.086426	2: 1.086924	3: 1.083376	4: 0.2307567	5: 0.2386741	6: 0.2258985
4	1: 1.086638	2: 1.087107	3: 1.083449	4: 0.230776	5: 0.2385777	6: 0.225558
4	1: 1.086082	2: 1.08678	3: 1.083857	4: 0.2306681	5: 0.238907	6: 0.2254433
4	1: 1.086021	2: 1.086736	3: 1.083372	4: 0.2305058	5: 0.238935	6: 0.2253883
4	1: 1.086537	2: 1.08722	3: 1.083831	4: 0.230644	5: 0.2385258	6: 0.2253385
4	1: 1.086927	2: 1.087158	3: 1.083856	4: 0.2306754	5: 0.2388429	6: 0.2251484
4	1: 1.087317	2: 1.087096	3: 1.083881	4: 0.2307068	5: 0.23916	6: 0.2249583
4	1: 1.087707	2: 1.087034	3: 1.083906	4: 0.2307382	5: 0.2394771	6: 0.2247682
4	1: 1.088097	2: 1.086972	3: 1.083931	4: 0.2307696	5: 0.2394942	6: 0.2245781
5	1: 2.859328	2: 2.867916	3: 2.84877	4: 0.4832593	5: 0.4986941	6: 0.4966687



5	1: 2.862447	2: 2.868324	3: 2.851844	4: 0.4826271	5: 0.4987077	6: 0.4967233
5	1: 2.864296	2: 2.871733	3: 2.855676	4: 0.4839867	5: 0.5001969	6: 0.4972649
5	1: 2.865128	2: 2.871906	3: 2.855436	4: 0.4828961	5: 0.4992056	6: 0.4968944
5	1: 2.86833	2: 2.876102	3: 2.859047	4: 0.4831682	5: 0.4986701	6: 0.4965286
5	1: 2.86879	2: 2.875877	3: 2.860353	4: 0.4834527	5: 0.4993634	6: 0.4968793
5	1: 2.853745	2: 2.876647	3: 2.861104	4: 0.4841212	5: 0.5006972	6: 0.4974402
5	1: 2.887868	2: 2.879664	3: 2.863563	4: 0.4840554	5: 0.5006475	6: 0.4974832
5	1: 2.88767	2: 2.878232	3: 2.862817	4: 0.4821643	5: 0.4988643	6: 0.496281
5	1: 2.88538	2: 2.880068	3: 2.863697	4: 0.4835059	5: 0.4995906	6: 0.4969449
6	1: 2.661374	2: 2.613395	3: 2.688907	4: 1.416786	5: 1.397752	6: 1.411372
6	1: 2.657179	2: 2.613409	3: 2.687374	4: 1.671357	5: 1.650515	6: 1.6687
6	1: 2.661374	2: 2.613395	3: 2.688907	4: 1.416786	5: 1.397752	6: 1.411372
6	1: 2.657179	2: 2.613409	3: 2.687374	4: 1.671357	5: 1.650515	6: 1.6687
6	1: 2.661374	2: 2.613395	3: 2.688907	4: 1.416786	5: 1.397752	6: 1.411372
6	1: 2.657179	2: 2.613409	3: 2.687374	4: 1.671357	5: 1.650515	6: 1.6687
6	1: 2.661374	2: 2.613395	3: 2.688907	4: 1.416786	5: 1.397752	6: 1.411372
6	1: 2.657179	2: 2.613409	3: 2.687374	4: 1.671357	5: 1.650515	6: 1.6687
6	1: 2.661374	2: 2.613395	3: 2.688907	4: 1.416786	5: 1.397752	6: 1.411372
6	1: 2.657179	2: 2.613409	3: 2.687374	4: 1.671357	5: 1.650515	6: 1.6687
7	1: 2.638303	2: 2.602147	3: 2.674996	4: 0.8074452	5: 0.7878412	6: 0.7996882
7	1: 2.639261	2: 2.601776	3: 2.674889	4: 0.805906	5: 0.7861279	6: 0.7985954
7	1: 2.637713	2: 2.600656	3: 2.673487	4: 0.8046407	5: 0.7842707	6: 0.7980604
7	1: 2.637658	2: 2.600486	3: 2.673771	4: 0.8031466	5: 0.7825137	6: 0.7974771
7	1: 2.638693	2: 2.600891	3: 2.673654	4: 0.8022083	5: 0.781319	6: 0.7956063
7	1: 2.638221	2: 2.600729	3: 2.674951	4: 0.800188	5: 0.7798032	6: 0.7950842
7	1: 2.638303	2: 2.602147	3: 2.674996	4: 0.8074452	5: 0.7878412	6: 0.7996882
7	1: 2.639261	2: 2.601776	3: 2.674889	4: 0.805906	5: 0.7861279	6: 0.7985954
7	1: 2.637713	2: 2.600656	3: 2.673487	4: 0.8046407	5: 0.7842707	6: 0.7980604
7	1: 2.637658	2: 2.600486	3: 2.673771	4: 0.8031466	5: 0.7825137	6: 0.7974771
7	1: 2.650827	2: 2.608411	3: 2.682292	4: 0.8562559	5: 0.83688	6: 0.8470134
7	1: 2.652089	2: 2.609181	3: 2.682892	4: 0.8572801	5: 0.8370795	6: 0.8469804
7	1: 2.650468	2: 2.608143	3: 2.682578	4: 0.8573355	5: 0.8376006	6: 0.8476639
7	1: 2.64794	2: 2.606494	3: 2.680609	4: 0.857952	5: 0.838779	6: 0.8482822
7	1: 2.648501	2: 2.606822	3: 2.681451	4: 0.858399	5: 0.8391316	6: 0.8490326

7	1: 2.649403	2: 2.607868	3: 2.680983	4: 0.8592447	5: 0.8385217	6: 0.8496575
7	1: 2.650977	2: 2.609	3: 2.682617	4: 0.8567954	5: 0.836699	6: 0.8474185
7	1: 2.653212	2: 2.609516	3: 2.683615	4: 0.8571332	5: 0.8363514	6: 0.8475973
7	1: 2.649549	2: 2.607351	3: 2.681842	4: 0.8574618	5: 0.8381995	6: 0.848157
7	1: 2.649483	2: 2.606308	3: 2.681787	4: 0.8580527	5: 0.8386351	6: 0.8490824
7	1: 2.642	2: 2.605126	3: 2.678911	4: 0.8492008	5: 0.8297097	6: 0.8431634
7	1: 2.640762	2: 2.603945	3: 2.678396	4: 0.8487684	5: 0.8301917	6: 0.8442751
7	1: 2.64096	2: 2.604793	3: 2.678446	4: 0.8486399	5: 0.8300289	6: 0.843615
7	1: 2.642	2: 2.605126	3: 2.678911	4: 0.8492008	5: 0.8297097	6: 0.8431634
7	1: 2.640762	2: 2.603945	3: 2.678396	4: 0.8487684	5: 0.8301917	6: 0.8442751
7	1: 2.64096	2: 2.604793	3: 2.678446	4: 0.8486399	5: 0.8300289	6: 0.843615
7	1: 2.642	2: 2.605126	3: 2.678911	4: 0.8492008	5: 0.8297097	6: 0.8431634
7	1: 2.640762	2: 2.603945	3: 2.678396	4: 0.8487684	5: 0.8301917	6: 0.8442751
7	1: 2.64096	2: 2.604793	3: 2.678446	4: 0.8486399	5: 0.8300289	6: 0.843615
7	1: 2.642	2: 2.605126	3: 2.678911	4: 0.8492008	5: 0.8297097	6: 0.8431634