T.C.

BAHÇEŞEHİR ÜNİVERSİTESİ

STATISTICAL LEARNING IN MODELING INTERRELATIONS AMONG VARIABLES: AN APPLICATION TO METABOLOMICS

M. S. Thesis

SAFİYE YAYLAOĞLU

İSTANBUL, 2011

T.C.

BAHÇEŞEHİR ÜNİVERSİTESİ

The Graduate School of Natural and Applied Sciences

Computer Engineering Graduate Program

STATISTICAL LEARNING IN MODELING INTERRELATIONS AMONG VARIABLES: AN APPLICATION TO METABOLOMICS

M. S. Thesis

Safiye YAYLAOĞLU

Supervisor: Asst. Prof. Dr. Olcay KURŞUN

İSTANBUL, 2011

T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ
The Graduate School of Natural and Applied Sciences
Computer Engineering Graduate Program

Title of the Master's Thesis	:	STATISTICAL LEARNING IN MODELING INTERRELATIONS AMONG VARIABLES: AN APPLICATION TO METABOLOMICS
Name/Last Name of the Student	:	Safiye YAYLAOĞLU
Date of Thesis Defense	:	27.01.2011

The thesis has been approved by the Graduate School of Natural and Applied Sciences.

Asst. Prof. Dr. F. Tunç BOZBURA
Acting Director

This is to certify that we have read this thesis and that we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Science.

Examining Committee Members

Asst. Prof. Dr. Olcay KURŞUN (Supervisor) :

Assoc. Prof. Dr. Çiğdem EROĞLU :

Asst. Prof. Dr. Tevfik AYTEKIN :

ACKNOWLEDGMENTS

I owe great thanks to many people who have helped and supported me during my study. In the first place I am heartily thankful to my supervisor, Asst. Prof. Dr. Olcay Kurşun, and his earlier M.Sc. students Okan Şakar and Heysem Kaya for their encouragement, guidance and support that helped me develop a better understanding of the subject.

Many thanks to my son Mehmet Alperen Sağlam for the joy of living. I would like to thank my husband Dr. Özgür Sağlam for his great patience and contribution. Many special thanks to my dear mother for her great encouragement and support of my life every time. I would like to thank my father, my sister, my brother and my cousin for their supports.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of my thesis.

Safiye Yaylaoğlu

ABSTRACT

STATISTICAL LEARNING IN MODELING INTERRELATIONS AMONG VARIABLES: AN APPLICATION TO METABOLOMICS

YAYLAOĞLU, Safiye

Computer Engineering Graduate Program

Supervisor: Asst. Prof. Dr. Olcay KURŞUN

January, 2011, 87 pages

In some machine learning problems, large datasets are naturally organized into some groups of variables, which are called *views* in the literature. Views can be used to predict the same target variable, such as the class of a given sample, such as in Parallel Interacting Multi-view Learning (PIML). In this thesis, we deal with a more general case, where the views are designed to predict different but related target variables. The goal here is to develop a mechanism for incorparating the interrelations among the target variables into their predictions, along with the input variables in their own views. In this study, the predictions obtained from the training phase of each view are used as additional inputs to the next iteration. Iterations are repeated until the interactions between the views in consecutive iterations become stable. The interrelations and interactions among the views are modeled using Support Vector Machines (SVM) along with optimization-related methods such as leave-one-out cross-validation, k-fold cross-validation, grid search and bootstrap resampling. The proposed method is compared with the classical regression implemented on single view in its application to a toy dataset and a real-world dataset of cancer (a metabolomics dataset obtained through nuclear magnetic resonance spectroscopy on tissue samples from healthy and cancerous human subjects in a study conducted by the biomedical engineering department at the University of North Carolina). The web of interrelations among the views might give insight to the clinicians in their research.

Keywords: Multi-view machine learning, support vector machines, parallel interactive multiview learning, prostate cancer metabolomics dataset.

ÖZET

DEĞİŞKENLER ARASINDAKİ İLİŞKİLERİN MODELLENMESİNDE İSTATİKSEL ÖĞRENME: METABOLOMİK ÜZERİNE BİR UYGULAMA

YAYLAOĞLU, Safiye

Bilgisayar Mühendisliği Yüksek Lisans Programı

Danışman: Yrd. Doç. Dr. Olcay KURŞUN

Ocak, 2011, 87 sayfa

Bazı yapay öğrenme problemlerinde büyük veri setleri literatürde *bakış* olarak bilinen doğal gruplara ayrılmıştır. Farklı bakışlar, aynı hedef değişkeni kestirmek için kullanılabilir, örneğin farklı bakışları verilen bir örneğin sınıfını kestirmede kullanılan paralel etkileşimli çok bakışlı öğrenmede (PIML) yapıldığı gibi. Buradaki amaç ise, bunun daha genel bir hali olarak, aralarında bazı istatistiksel ilişkiler olan farklı değişkenlerin, kendi bakışlarından kestiriminde nasıl birleştirilebileceğini ele alacağız. Amacımız, farklı bakışların farklı hedef değişkenlerini kestirmesi sırasında, bu farklı hedef değişkenler arasındaki bağıntıları da kullanan bir yöntem geliştirmektir. Bu çalışmada bir hedef değişken için eğitim safhasında elde edilen tahminler bir sonraki iterasyonun, kendi bakışındaki değişkenlere ilaveten ek girdi olarak kullanılmıştır. Iterasyonlar bakışların birbiri ile etkileşimi sabit hale gelinceye kadar tekrar ettirilmiştir. Bakışlar arası iletişim ve etkileşim destek vektör makinesi (DVM) ile modellenmiştir. DVM optimizasyonu için birini-dışarıda-bırak çapraz sağlama, *k*-kat çapraz sağlama, ızgara arama ve kendini yükleme tekrar örnekleme metotları uygulanmıştır. Önerilen yöntem sentetik veri kümesi ve gerçek bir kanser veri kümesi (North Carolina Üniversitesi biyomedikal mühendisliği bölümünde sağlıklı ve kanserli insan deneklerinden nükleer manyetik rezonans spektroskopisiyle elde edilmiş metabolomik bir veri kümesi) üzerinde uygulanmış ve tek bakışlı klasik bağlanım yöntemiyle karşılaştırılmıştır. Bakışlar arasındaki ilişkilerin ortaya çıkarılması ve birbirlerini nasıl etkilediklerini bu şekilde ortaya koymak, klinik çalışmalara, az da olsa, katkı sağlayabilir.

Anahtar Kelimeler: Çok bakışlı yapay öğrenme, destek vektör makinesi, paralel etkileşimli çok bakışlı öğrenme, prostat kanseri metabolomik veri kümesi.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

A Library for Support Vector Machine	:	LIBSVM
C-SVM Classification	:	C-SVC
Classical Single-view Regression	:	CSR
epsilon-SVM Regression	:	epsilon-SVR
Karush-Kuhn-Tucker	:	KKT
Leave-One-Out Cross-Validation	:	LOOCV
Mass spectrometry	:	MS
Mean Squared Error	:	MSE
nu-SVM Classification	:	nu-SVC
nu-SVM Regression	:	nu-SVR
Nuclear Magnetic Resonance	:	NMR
Nuclear Magnetic Resonance Imaging	:	NMRI
Nuclear Magnetic Resonance Spectroscopy	:	NMRS
Optimal Objective Value	:	obj
Parallel Interacting Multi-view Learning	:	PIML
Parallel Interacting Multi-view Regression	:	PIMR
Quadratic Programming	:	QP
Radial Basis Function	:	RBF
Root Mean Squared Error	:	RMSE
Standard Support Vector Regression	:	CSR
Support Vector Classification	:	SVC
Support Vector Machines	:	SVM
Support Vector Regression	:	SVR
Squared Correlation Coefficient	:	SCC

LIST OF SYMBOLS

Bias	:	b
Cancer tissue	:	C
Cancer view	:	C_{view}
Class label	:	y
Cost	:	C
Dimensionality	:	D
Dual lagrangian	:	L_D
Euclidian form of w	:	$\|\|w\|\|$
Feature vector	:	F
Gamma	:	γ
Glandular tissue	:	G
Glandular view	:	G_{view}
Hessian matrix	:	$\mathbf{H}$
Hyperplane	:	H
Input vector	:	x
Insensitive tube	:	ε
Kernel function	:	$K(\cdot)$
Lagrange multiplier for x_i	:	α_i
Lagrange parameter for x_i	:	β_i
Mapping function	:	$\Phi(\cdot)$
Metabolite of cancer view	:	Cx
Metabolite of stromal view	:	Sx
Metabolite of glandular view	:	Gx
Model or regressor	:	M
Normal to hyperplane	:	w
Primary lagrangian	:	L_p
Random number	:	R
Scalar value	:	$\times$
Slack variable	:	ξ_i

Support Vectors	:	x_s
Squared Correlation Coefficient	:	r^2
Standard deviation	:	σ
Stromal tissue	:	S
Stromal view	:	S_{view}
Target of cancer view	:	T_C
Target of glandular view	:	T_G
Target of stromal view	:	T_S

1. INTRODUCTION

Metabolomics is a new field in analytical biochemistry. Metabolomics (the metabolite network) is the cheap and correct separation, definition and measurement of all metabolites in cells, tissues or biological fluids in a short time with high throughput technologies such as Nuclear Magnetic Resonance (NMR) (Özkara 2008). The term *metabolite* is any substance essential to the metabolism of a particular metabolic process. Metabolites are used in diagnosing diseases. For example, using Choline, Creatine and Citrate metabolites, the result of the computation (Choline + Creatine)/Citrate is often investigated as a marker for Prostate Cancer (Kim *et al.* 2004) In metabolomics studies of different brain tumors and cancers, it is explored that the different metabolites are increased. Metabolomic researches on brain tumors show that for different brain tumors and cancers different metabolites increases (Griffin and Kauppinen 2007).

Metabolomics studies rely on NMR that is the absorption of electromagnetic radiation of a specific frequency by an atomic nucleus that is placed in a strong magnetic field, used especially in spectroscopic studies of molecular structure to monitor tissue metabolism and to distinguish between normal and abnormal cells. NMR can recognize all existing metabolites and calculate their concentrations in a sample. Mass spectrometry (MS) is a complementary application to NMR. MS can construct the profiles of thousands of metabolites and can analyze these profiles more precisely. Metabolite profiles can be processed and analyzed by related computer programs. (Özkara 2008). NMR is the basis of Nuclear Magnetic Resonance Spectroscopy (NMRS). NMRS (less sensitive than MS) provides a powerful complementary technique for the identification and quantitative analysis of metabolites in tissue extracts. Nuclear Magnetic Resonance Imaging (NMRI) can retrieve the image of the internal structure of the body in three dimensional space and can detect tumor tissues.

Cancer is the unbounded increase in population or size of the cells (extra tissue) caused by DNA damages. This mass of extra tissue, called tumor, can be benign or malignant. Benign tumors are not cancer. They can usually be removed from the body. Benign tumors do not spread to other parts of the body. Benign tumors are rarely a threat to life.

If benign tumors are not removed from the body, they may transform to cancerous tumors. Malignant tumors are cancer. Cancer cells can damage tissues and organs near the tumor. Also, cancer cells can break away from a malignant tumor and enter the bloodstream or lymphatic system. This is how cancer spreads to the other parts of the body[1].

In the literature there are many studies analyzing and detecting different cancer types by using computer programs. Among these, SVM classification and regression methods also take place. Ding *et al.* (2009) used SVM regression in their study and they tried to detect lung cancer by analyzing breath biomarkers. In another work Furey *et al.* (2000) has developed a method on ovarian cancer analysis by using SVM classification. Liu *et al.* (2003) diagnosed the breast cancer by using SVM classification and they compared SVM with other learning techniques.

In this study, the proposed SVM regression technique is applied on a real prostate cancer dataset (Keshari *et al.* 2009). Prostate is a part of reproductive system. Prostate cancer is one of the most prevalent types of cancer in men over the age of fifty. The cancer cells may spread to the other organs. The presence of prostate cancer is indicated by biopsy. After a biopsy, the pathologist analyzes the samples under a microscope. If cancer is present, then the pathologist reports the progress of the cancer tissues[2].

The dataset used contains the percentages of glandular, stromal and cancer in a whole tissue taken from healthy and cancer subjects. Glandular tissue is a healthy tissue, stromal tissue is benign tumor and cancer tissue is a malignant tumor. The tissues are parts of a whole for every instance of the dataset; therefore each part or tissue is related with the others. The dataset also contains the concentrations of metabolites related to these tissues to be used as the independent variables in predicting these percentages. These three views (G for glandular, S for Stromal, and C for Cancer) were the natural partition of the metabolites used by (Keshari *et al.* 2009); that is, even though many metabolites are interrelated, for the sake of simplicity and understandability, some are thought to form more closely related groups (Christoudias *et al.*, 2008, Culp *et al.*

[1] Source at: http://www.medicinenet.com/colon_cancer/article.htm [cited November, 2010]
[2] Source at: http://en.wikipedia.org/wiki/Prostate_cancer [cited November, 2010]

2009), for example, some metabolites are taken into G_{view} because they are thought to be more related with the G content in the tissue. Similarly, S_{view} is for stromal view, which indicates the variable group primarily responsible for predicting the stromal tissue percentage in the sample and vice versa for C_{view}.

In this work, we call the task of predicting a target variable from all the views merged together as Classical Single-view Regression (CSR). Using multiple views of the same semantic object (or related objects) is a recently popular topic (Wang and Chen 2009, Hardoon *et al.* 2004, Gonen and Alpaydin 2010; Kursun *et al.* 2011; Kursun and Favorov 2010). We first use the views independently in training; but iteratively, the predictions obtained from them were also used as additional inputs to other views for the next iterations. We called this approach Parallel Interacting Multi-view Regression (PIMR). Learning vector-valued functions is an ongoing research area, there are some recent work on this topic such as (Theodoros et al. 2005). However, in this study, we investigate deeper into a simpler approach by Sakar *et al.* (2009), who proposed a method called Parallel Interacting Multi-view Learning for classification. A protein dataset is used for their study. PIMR is based on the same idea, however it is different in that it aims to predict different (but related) variables in different views and also it performs regression rather than classification. By considering the interrelations among the target values, each of which is a part of a whole, PIMR increases the processing complexity in exchange of increased generalization capability of regressors, provides better results using less data in regression, reaches higher accuracy and overcomes Bellman's (1961) curse of dimensionality as compared to CSR.

In Section 2, a brief explanation of SVM is provided together with leave-one-out cross-validation, *k*-fold cross-validation, grid search, and bootstrap resampling methods. These methods are used to improve the accuracy of PIMR. In Section 3, synthetic dataset is introduced, which has a similar structure to original prostate cancer dataset and SVM is applied to both original and synthetic dataset. In Section 4, the results are provided and compared.

2. MATERIALS AND METHODS

2.1 SUPPORT VECTOR MACHINES (SVMs)

Support Vector Machine (SVM) is an efficient supervised learning algorithm that is using both classification and regression for data analysis. Today, SVM analysis performance is comparably better than the other statistical models such as Neural Networks (Kecman 2004). The current SVM methodology was first announced with the paper *A training algorithm for optimal margin classifier* at the COLT 1992 conference by Boser, Guyon and Vapnik. The soft margin classifier related with the classification case, was introduced by Cortes and Vapnik in 1995. In the same year the algorithm was extended to the case of regression by Vapnik. Recently, SVM is a very popular algorithm used in many fields of engineering researches, such as face recognition (Heisele *et al.* 2001), handwriting analysis (Venkatesh and Sureshkumar 2009), speech analysis (Campbell *et al.* 2006), text categorization (Joachims 1998). In the literature, main research fields applying these methods are pattern recognition and machine learning.

In database systems classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends. In these forms classification predicts the category that the data falls in whereas, prediction estimates data based on the given attributes as a continuous function. The synonym of data prediction is regression which is a statistical methodology that is most often used for numeric prediction.

Basic idea of SVM is to find an optimal hyperplane for linearly separable data. It is always easy to separate two different type patterns linearly. If the data class number is two, this is called binary classification. If the data is separable linearly in low dimensional space this is called linearly separable SVM. If the classification is not possible to separate linearly then the data is extended from low dimensional input space to high dimensional feature space for separating. This is called nonlinearly separable SVM. If the number of classes is more than two then it is named as multi-class SVM. These methods have usability for regression, too.

SVM has two basic techniques Support Vector Classification (SVC) and Support Vector Regression (SVR). As in database systems, main task in SVC is to separate the patterns into specific categories whereas the task of regression is to forecast the target values of samples. In other words, if the solution or output is continuous then the problem is related with regression, if it is categorical then the problem is related with classification. For the solution of a specific problem, SVM must be trained by a teacher to generate the model/predictor or classifier. The model is used for testing the data to obtain solution or output. In this sense, summarizes the basic operations of SVC and SVR. In Figure 2.1 there is a two-step process of classification and regression informally. The first step is training phase and the second is testing phase. In classification, output of training phase is represented by the term *classifier*. In regression, output of training phase is resented by the term *regressor*.

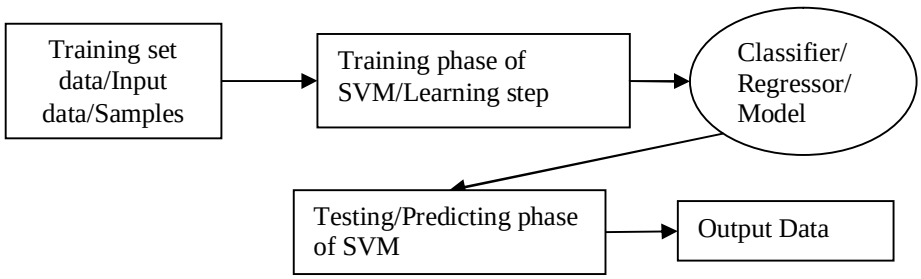


Figure 2.1 : SVC and SVR operations

2.1.1 Support Vector Classification (SVC)

In this section, binary classification and multi-class classification are detailed. If the number of different classes that the input should be mapped is more than two then classification is done with hyperplanes each of which separates a class. In this case; it is mentioned about multi-class classification. Binary classification is to separate given objects into two classes in 2-dimensional feature space, such as, objects which have many special properties or attributes. One of the useful tasks of the classification is to determine the patients healthy or unhealthy. Binary classification can be achieved by two cases namely linearly separable and nonlinearly separable.

2.1.1.1 When the Data are Linearly Separable

In classification, an instance, x is represented by a vector $x = (\times_1, \times_2, \dots, \times_n)$ where n denotes the number of the attributes/ features, or dimensionality. F is used to denote the features of the related instance, respectively $F_1, F_2, \dots, F_n$. Classification implements the function or mapping $y = f(x)$ where y is used to denote the class label or scalar output and x_i is the related instance. It is supposed that the related instance and the class label or output is represented by x_i and y_i respectively. The training dataset consists of instances $\{x_i, y_i\}$, where $i = 1, \dots, l$, $x_i \in \Re^D$. l denotes the number of instance. Each of y_i can take one of the values, either -1 or $+1$, $y_i \in \{-1, +1\}$. D denotes dimensionality.

When the data is linearly separable, it means that, it is drawn a line on a graph. If the number of features is 2, i.e. $x_i = (\times_1, \times_2)$ or $F = 2$ or $D = 2$ (2-D), it is drawn a line to separate two different classes. If the number of features is more than two, $x_i = (\times_1, \times_2, \dots, \times_n)$ i.e. $F > 2$ or $D > 2$, it is drawn a hyperplane on a graph (Fletcher 2009).

SVMs are based on finding maximum marginal hyperplane in a high dimensional feature space. There are many hyperplanes that separate the classes. These are called separating hyperplanes. From the given set of objects, the main goal is produce a classifier that determines the class label of the given new object. In classification, drawing a boundary to separate classes on the same plane is possible. But as shown in Figure 2.2, it is not clear where the boundary will be occurring. Here, there are two types of samples/classes shown on two-dimensional plane and possible occurrences of boundary are shown (Han and Kamber 2006).

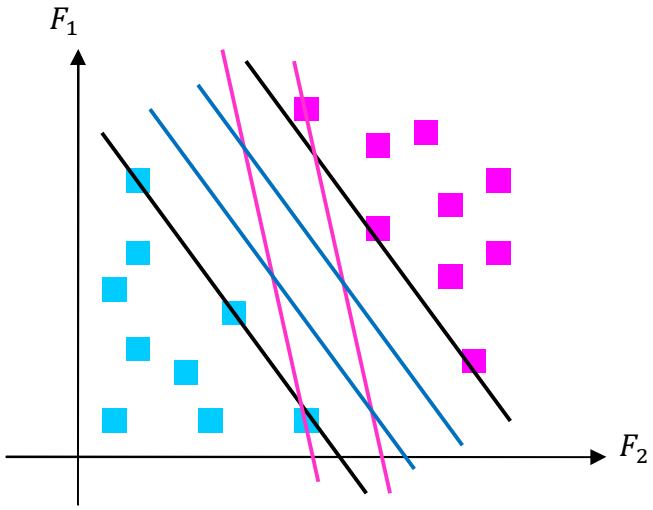


Figure 2.2 : Possible decision hyperplanes/boundaries example in SVM

The boundary must be located at the most distant place of the classes' members for minimizing the training error. SVM determines how to draw this boundary. For this operation two parallel boundaries are drawn near the instances. The distance between two parallel lines is called margin as shown in Figure 2.3.

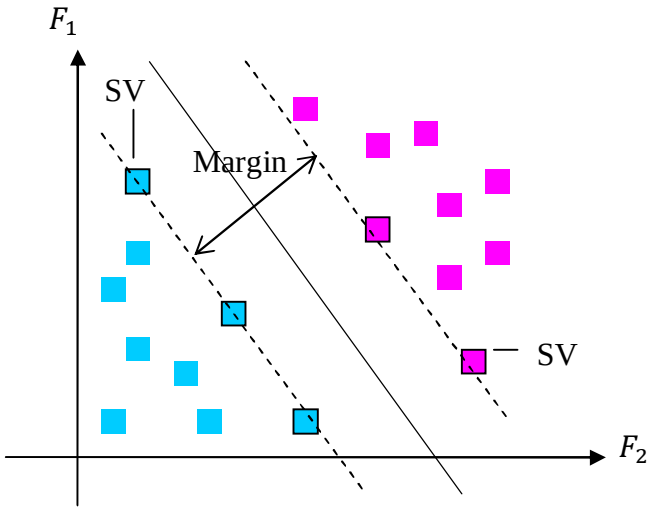


Figure 2.3 : Support vectors (SVs) and maximum margin

The margin plays very important role in SVM. The size of the margin must be maximum length. The parallel lines touch a number of points, if the margin is maximized. These points on margin boundaries are called "support vectors". A new line

is generated in the middle of the parallel separating hyperplanes. This line is called maximum margin hyperplane in binary classification, is represented by solid line in Figure 2.3 (Han and Kamber 2006). The maximum margin causes minimal error and the minimum margin causes maximal risk to error when classifying a new instance.

According to Figure 2.4, the equation $b/\|w\|$ is the distance from maximum marginal hyperplane to the origin where, b is the scalar or bias, w is the normal to the hyperplane and $\|w\|$ is the Euclidian norm of w. In addition w is identified an unknown dependency weight vector between input and output.

Referring to Figure 2.4 hyperplanes are written as;

$$H_1 : x_i \cdot w + b \geq 1 \text{ for } y_i = +1 \quad (2.1)$$

$$H_2 : x_i \cdot w + b \leq -1 \text{ for } y_i = -1 \quad (2.2)$$

Any point above the hyperplane H_1 belongs to $+1$ class and any point below the hyperplaneH_2 belongs to $+1$ class (Figure 2.4).

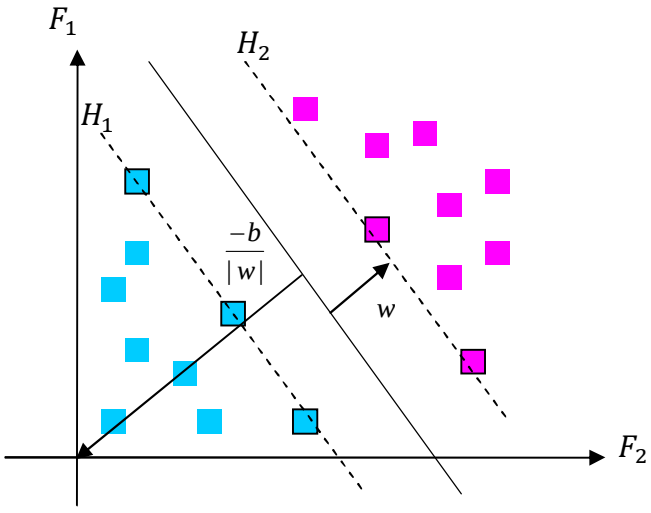


Figure 2.4 : Parallel separating hyperplanes and maximum margin hyperplane

As "·" denotes the dot production, Eq. 2.1 and Eq. 2.2 can be combined as:

$$y_i\left(x_i \cdot w + b\right) \geq +1 \text{ for } i = 1, 2, \ldots, n \tag{2.3}$$

Any point falls on H_1 and H_2, is called "Support Vector" or SV (Figure 2.3). These points can be described by:

$$H_1 : x_i \cdot w + b = 1 \tag{2.4}$$

$$H_2 : x_i \cdot w + b = -1 \tag{2.5}$$

The distance between H_1 to separating hyperplane or maximum marginal hyperplane is equal to $1/\|w\|$. According to this equation, the margin length is equal to $2/\|w\|$.
To find the maximum margin equations below are used.

$$min \frac{1}{2}\|w\|^2 \tag{2.6}$$

$$y_i(x_i \cdot w + b) \geq +1 \; \forall i \tag{2.7}$$

Here, the problem is in Eq. 2.6 and the condition is in Eq. 2.7. Lagrangian formulation is used to solve the problem in Eq. 2.6. If the Lagrangian multiplier α, $\alpha_i \geq 1 \; \forall i$, is allocated to Eq. 2.6 and Eq. 2.7, then Eq. 2.8 is obtained.

$$L_p = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i y_i \left(x_i w + b\right) + \sum_{i=1}^{l} \alpha_i \tag{2.8}$$

Solving Eq. 2.8 is very complicated. To solve the equation Karush-Kuhn-Tucker (KKT) conditions are used to transform it to a dual problem. KKT conditions are like those:

$$\frac{\partial L_p}{\partial w} = 0 \Rightarrow w = \sum_i \alpha_i y_i x_i \tag{2.9}$$

$$\frac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_i \alpha_i y_i = 0 \tag{2.10}$$

These conditions are substituted into Eq. 2.8. In this case, it is obtained a new formulation which is needed to maximize:

$$L_D \equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j \text{ such that } \alpha_i \geq 0 \; \forall i, \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{2.11}$$

$$\equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i H_{ij} \alpha_j \text{ where } H_{ij} \equiv y_i y_j (x_i x_j) \equiv y_i y_j x_i^T x_j \tag{2.12}$$

$$\equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha^T \mathbf{H} \alpha \text{ such that } \alpha_i \geq 0 \; \forall \boldsymbol{i}, \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{2.13}$$

H denotes the Hessian Matrix (Kecman 2004). L_D is dual form of L_p. It is needed for minimizing L_p and maximizing L_D. This is a convex quadratic optimization problem. It is needed Quadratic Programming (QP) solver to find α. If α substitutes into Eq. 2.9 then w is obtained. Any data point that satisfies Eq. 2.10 is Support Vector which is on the separating hyperplane. x_s denotes Support Vector. The bias can be calculated by using SVs. b and w define the separating hyperplanes, hence SVM (Fletcher 2009).

2.1.1.2 When the Data are Linearly Non-Separable

Data may not be always linearly separable (Figure 2.5). The method *soft margin* is a solution to this case. In this case, positive slack variable ξ_i, $i = 1, \ldots, n$ is used (Cortes and Vapnik 1995). Slack variable indicates tolerances of misclassification. If Eq. 2.1

and Eq. 2.2 are rewritten again with the slack variable, the following equations are obtained.

$$x_i \cdot w + b \geq 1 - \xi_i \text{ for } y_i = +1 \tag{2.14}$$

$$x_i \cdot w + b \leq -1 + \xi_i \text{ for } y_i = -1 \tag{2.15}$$

$$\xi_i \geq 0 \ \forall i \tag{2.16}$$

If the slack variable is equal to 0 (i.e. $\xi_i = 0$), then the instance x_i is classified correctly, if the slack variable is between 0 and 1 (i.e. $0 < \xi_i < 1$), then the instance x_i is classified correctly but lays in between F_1 and F_2 hyperplanes, if $\xi_i \geq 1$, then x_i is classified incorrectly (Alpaydın 2004). Eq. 2.14, Eq. 2.15 and Eq. 2.16 are combined into:

$$y_i \left(x_i \cdot w + b \right) - 1 + \xi_i \geq 0 \text{ where } \xi_i \geq 0 \ \forall i \tag{2.17}$$

In Figure 2.5 the point is on the wrong side of the maximal margin hyperplane.

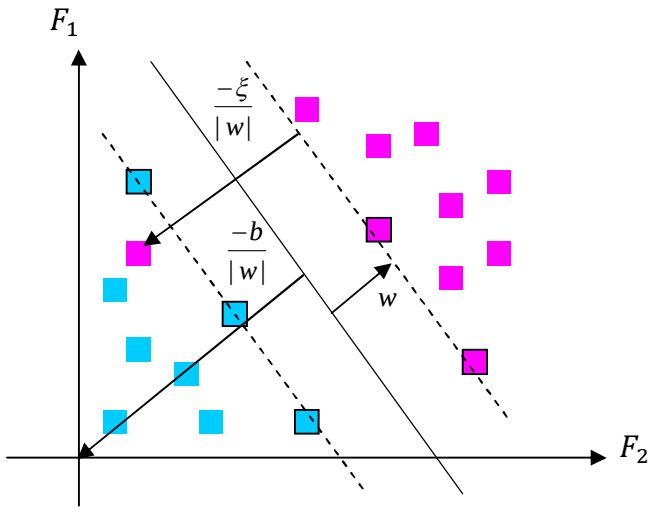


Figure 2.5 : Non-linearly separable data

In order to prevent the system to adjust the boundary for each case, a parameter C is added into the equation. This is the maximum value that the Lagrange multipliers can take. Then, the Lagrange formulation can be reformed in Eq. 2.18 as below:

$$L_p = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i - \sum_{i=1}^{l}\alpha_i[y_i(x_i w + b) - 1 + \xi_i] - \sum_{i=1}^{l}\beta_i\xi_i \tag{2.18}$$

In Eq. 2.18, β_i is the Lagrange multiplier which provides ξ_i to be positive. Since this Lagrange formulation is very complex to solve it is converted to a dual problem as it is done with the linearly separable examples. If the KKT rules are applied to this problem then the equations Eq. 2.19, Eq. 2.20 and Eq. 2.21 are obtained (Demirci 2007).

$$\frac{\partial L_p}{\partial w} = 0 \Rightarrow w - \sum_i \alpha_i y_i x_i \tag{2.19}$$

$$\frac{\partial L_p}{\partial b} = -\sum_i \alpha_i y_i = 0 \tag{2.20}$$

$$\frac{\partial L_p}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \tag{2.21}$$

If these equations are substituted in Eq. 2.18 then the following equation is obtained.

$$L_D \equiv \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha^T \mathbf{H}\alpha \text{ such that } 0 \le \alpha_i \le C\,\forall i \text{ and } \sum_{i=1}^{l}\alpha_i y_i = 0 \tag{2.22}$$

x_i values corresponding to the Lagrange multipliers, $0 < \alpha_i < C$, which are the solutions to this problem, are SVs.

2.1.1.3 Multi-Class Classification

The SVMs explained so far work if the examples are formed with only two different classes. In case of more classes the system should make multiple classifications. There are two basic approaches used to separate multiple classes in SVM. First approach is to reform the Lagrange function used directly for SVM so that it can be used for multiple classes operations. However, since errors increase as the number of classes increases, this approach is not a preferred one. Second approach is to run SVM to make dual classifications with respect to the number of classes. Some methods applied in this approach are one versus one, one versus all and directed loop free graphs (Demirci 2007).

Since the multi-class classification is not included in the scope of this thesis, it is not detailed more.

2.1.2 Support Vector Regression (SVR)

In regression, it is predicted real-valued output y_i, instead of categorical output. The training dataset form is:

$$\{x_i, y_i\} \text{ where } i = 1,\ldots,L \quad y_i \in \Re,\ x_i \in \Re^D$$
$$y_i = w \cdot x_i + b \tag{2.23}$$

In Figure 2.6 boarded squares are SVs. These SVs are on the tube boundary or in the outside of the tube. In other words, there is no SV in the tube.

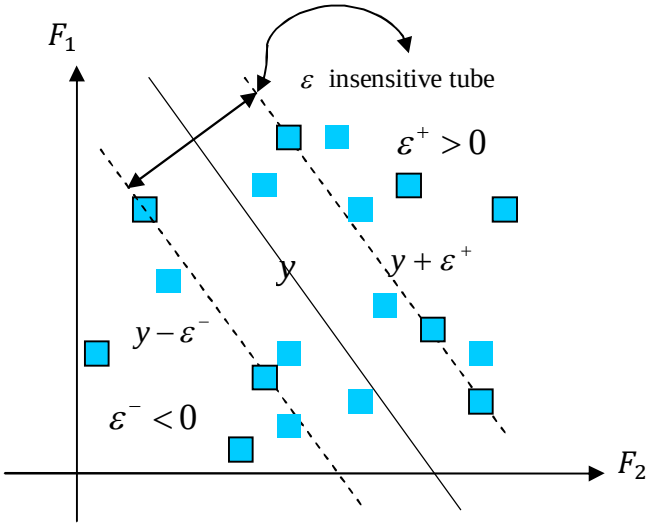


Figure 2.6 : ε-insensitive tube

Eq. 2.23 shows the linear regression hyperplane. SVM uses a penalty function to measure the error approximation between continuous output y_i and target t_i of the related instance x_i. The error/loss is equal to 0 if the measured distance between the predicted y_i and the actual value t_i is less than w, i.e. if $|t_i - y_i| < w$. This is the Vapnik's *linear loss* function. If the predicted value is within the tube then the loss is zero. For the other points outside the tube, the loss equals to the difference between the predicted value and the radius ε of the tube (Kecman 2004).

The slack variables for the points outside the tube are given below:

$$t_i \le y_i + \varepsilon + \xi^+ \tag{2.24}$$

$$t_i \ge y_i - \varepsilon - \xi^- \tag{2.25}$$

The error function can be written as:

$$C\sum_{i=1}^{l}\left(\xi_i^+ + \xi_i^-\right) + \frac{1}{2}\|w\|^2 \tag{2.26}$$

It is needed to minimize Eq. 2.26 under constraints $\xi^+ \geq 0,\ \xi^- \geq 0\ \forall i$ Lagrange multipliers are used to make this minimization process.

$$\alpha_i^+ \geq 0, \quad \alpha_i^- \geq 0, \quad \beta_i^+ \geq 0, \quad \beta_i^- \geq 0 \quad \forall_i :$$
$$L_P = C\sum_{i=1}^{l}(\xi_i^+ + \xi_i^-) + \frac{1}{2}\| w \|^2 - \sum_{i=1}^{l}(\beta_i^+\xi_i^+ + \beta_i^-\xi_i^-) - \sum_{i=1}^{l}\alpha_i^+(\varepsilon + \xi_i^+ + y_i - t_i) - \sum_{i=1}^{l}\alpha_i^+(\varepsilon + \xi_i^- - y_i + t_i) \tag{2.27}$$

Eq. 2.27 is needed to be minimized with respect to w, b, ξ_i^+, ξ_i^-, and maximized with respect to Lagrange multipliers α_i^+, α_i^-, β_i^+, β_i^-. The following equations show the derivates of L_p with respect to primal variables w, b, ξ_i^+, ξ_i^-, and setting the derivatives to 0.

$$\frac{\partial Lp}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{l}(\alpha_i^+ - \alpha_i^-)x_i \tag{2.28}$$

$$\frac{\partial Lp}{\partial b} = 0 \Rightarrow \sum_{i=1}^{l}(\alpha_i^+ - \alpha_i^-) = 0 \tag{2.29}$$

$$\frac{\partial Lp}{\partial \xi_i^+} = 0 \Rightarrow C = \alpha_i^+ + \beta_i^+ \tag{2.30}$$

$$\frac{\partial Lp}{\partial \xi_i^-} = 0 \Rightarrow C = \alpha_i^- + \beta_i^- \tag{2.31}$$

It is needed to be maximized L_D with respect to α_i^+ and α_i^- $(\alpha_i^+ \geq 0, \alpha_i^- \geq 0)$ where;

$$L_D = \sum_{i=1}^{l}(\alpha_i^+ - \alpha_i^-)t_i - \varepsilon\sum_{i=1}^{l}(\alpha_i^+ - \alpha_i^-) - \frac{1}{2}\sum_{i,j}^{l}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)x_i x_j \tag{2.32}$$

such that $0 \leq \alpha_i^+ \leq C,\ 0 \leq \alpha_i^- \leq C$ and $\sum_{i=1}^{l}(\alpha_i^+ - \alpha_i^-) = 0\ \forall_i$.

Substituting Eq. 2.26 into Eq. 2.23 the following equation is found:

$$y' = \sum_{i=1}^{l} (\alpha_i^+ - \alpha_i^-) x_i \cdot x' + b \tag{2.33}$$

As a result a set of Support Vectors x_s are found.

2.1.3 Non–Linear Support Vector Machines

In previous sections it is mentioned about linear SVM. In this section non-linear SVM is explained. If the input data aren't linearly separable in its original space, then the data are transformed into a new high dimensional space using a nonlinear mapping. In this space the input data are separated with a separating hyperplane (Figure 2.7).

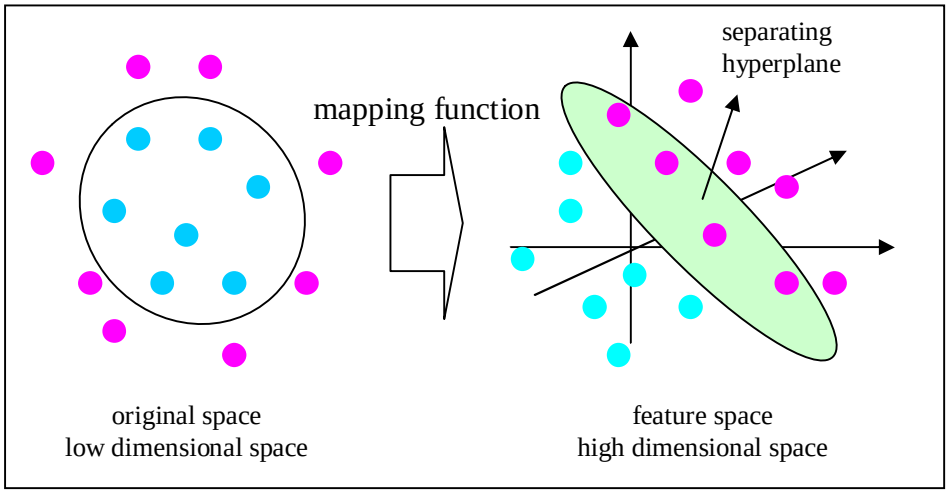


Figure 2.7 : SVM process flow

Kernel trick (Schölkopf and Smola 2002, Favorov and Kursun 2011) is used to find the maximal marginal hyperplane when the input data are taken for separating into the high dimensional feature space using the suitable mapping function, i.e. $x \rightarrow \Phi(x)$. There are many mapping functions and Table 2.1 illustrates a few special kernel functions.

Table 2.1 : Kernel functions

Kernel Type	Equations
Gaussian Radial Basis Function (RBF) Kernel	$K(x_i, x_j) = e^{-\left(\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)}$
Polynomial Kernel	$K(x_i, x_j) = (x_i \cdot x_j + a)^b$
Sigmoid Kernel	$K(x_i, x_j) = \tanh(ax_i \cdot x_j - b)$
Linear Kernel	$K(x_i, x_j) = x_i \cdot x_j$

2.2 A LIBRARY FOR SUPPORT VECTOR MACHINES (LIBSVM)

LIBSVM is a simple and efficient software for both classification and regression. There are many SVM packages like LibSVM by Chang and Lin, MySVM[3] by Stefan Rübing and SVMLigth[4] by Thorsten Joachims, etc… Because of its simple use LibSVM is selected among these packages and some modifications are applied for improving its performance for grid-search of optimal SVM parameters combined with cross-validation.

LIBSVM supports C-SVM classification (C-SVC), nu-SVM classification (nu-SVC), epsilon-SVM regression (epsilon-SVR), nu-SVM regression (nu-SVR), one-class SVM and multi-class classification. In this study, Java version of LIBSVM (Chang and Lin 2001)[5] is preferred to use. LIBSVM includes svm_train and svm_predict classes. It is prepared a separate dataset for each class. In general, it is called as train/training dataset which is used for svm_train class, test/testing dataset is used for svm_predict class. LIBSVM is implemented in accordance with reading from a file. The following format is an instance of the training and testing datasets in corresponding files.

<label> <index 1>:<feature 1> <index 2>:<feature 2> …

The label indicates the target of a class in classification, but in regression it indicates a continues/real number. Each feature has an index which starts from 1 and must be in ascending order. In testing dataset, the label is not necessary, i.e. can be any number.

[3] Software available at: http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/index.html [cited November, 2010]
[4] Software available at: http://svmlight.joachims.org/ [cited November, 2010]
[5] Software available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm/ [cited August, 2008]

The label is just used for calculating errors (in regression) and accuracy (in classification).

The general information about parts of LIBSVM is given below. The changes on LIBSVM code and used hyper-parameters in the study are explained (Table 2.2).

Table 2.2 : Hyper parameters of LibSVM

Options / Hyper Parameters of SVM	**Default Value**	**Types**
-s svm_type: set type of SVM	default 0	0 -- C-SVC 1 -- nu-SVC 2 -- one-class SVM 3 -- epsilon-SVR 4 -- nu-SVR
-t kernel type: set type of kernel function	default 2	0 -- linear: u'*v 1 -- polynomial: (gamma*u'*v + coef0)^degree 2 -- radial basis function: exp(-gamma*|u-v|^2) 3 -- sigmoid: tanh(gamma*u'*v + coef0) 4 -- pre-computed kernel (kernel values in train_file)
-g (γ) gamma: set gamma in kernel function	default 1/k	
-c (C) cost: set the parameter C of C-SVC, epsilon-SVR, and nu-SVR	default 1	

Source: http://www.csie.ntu.edu.tw/~cjlin/libsvm/ [cited November, 2009]

Cost is the penalty factor or error tolerance. If it is too large, it has a high penalty for nonseparable points and it may store many support vectors and overfit. If it is too small, it may have underfitting (Figure 2.8).

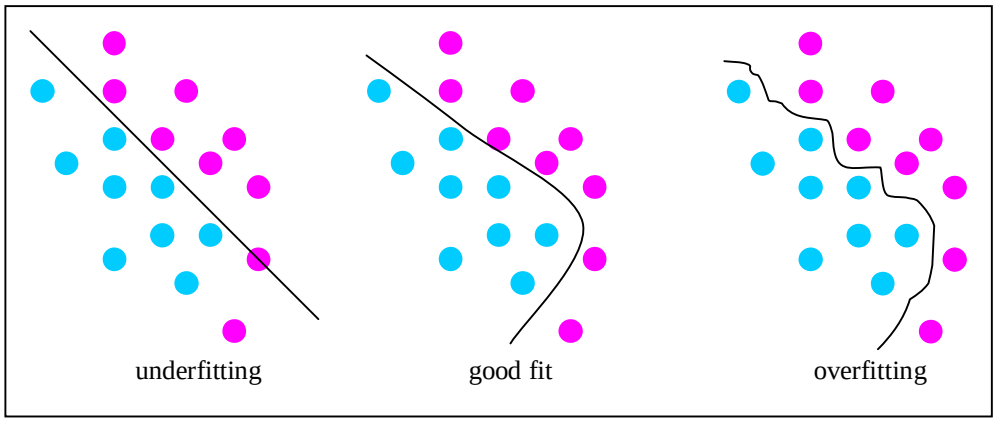


Figure 2.8 : Underfitting and Overfitting

2.2.1 How to use LIBSVM Classes

2.2.1.1 Usage of "svm-train"

Usage: svm_train [options] train_file [model_file]

train_file contains the training dataset. svm_train generates the model file. Below, the usage examples are provided:

>java svm_train -s 1 -t 0 -c 10 train_file model_file

Train a classifier with linear kernel, value of Cost is chosen 10 and the other options are chosen default.

>java svm_train -s 3 -t 2 –g 0.5 train_file model_file

Solve a SVM regression problem with RBF kernel, the value of gamma is 0.5 and the other options are chosen default.

2.2.1.2 Usage of "svm-predict"

Usage: svm-predict test_file model_file output_file

test_file includes the testing dataset which will be predicted. svm_predict produces output in the output-file.

2.2.1.3 Usage of “svm-toy”

This is a simple and useful graphical program which is programmed in Java for classification of data using the options of SVM. Change button chooses the class label (1, 2 and 3). The data is obtained by pressing points to the screen after selecting colors. Run button is used to obtain a model and separates the data. Clear button clears the screen. Save button is used to store the data seen on the screen. Load button is used to install the saved data. Each color point on the screen has a label (1, 2 and 3) and two features (x-axis and y-axis).

Below a very simple svm-toy example is provided (Figure 2.9, Figure 2.10).

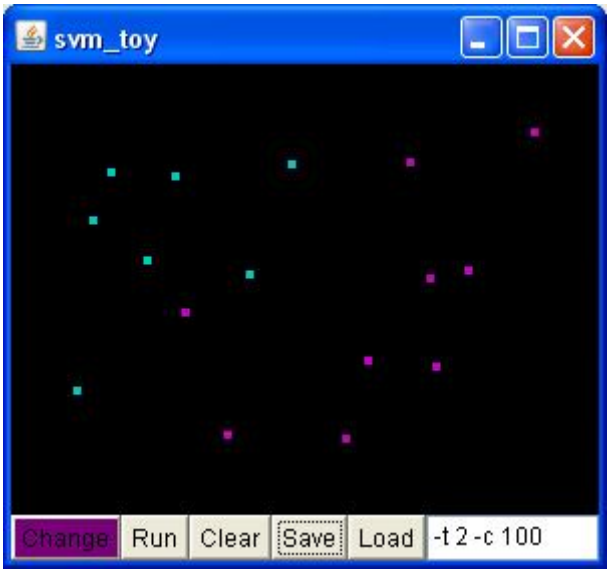


Figure 2.9 : Before running svm-toy

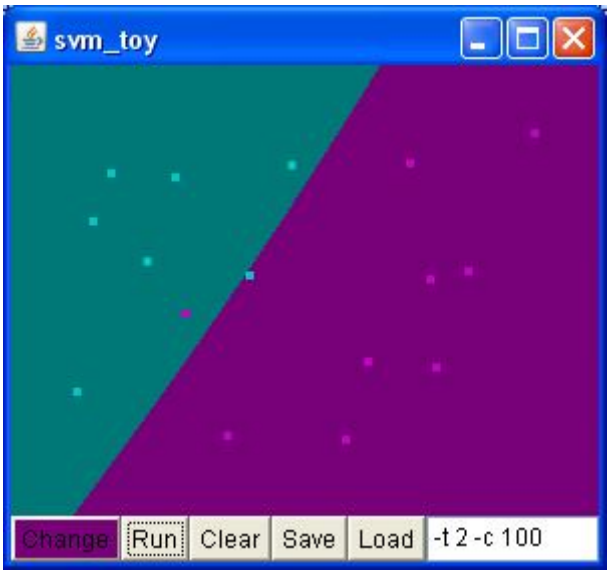


Figure 2.10 : After running svm-toy

As shown in Figure 2.9 and Figure 2.10 it is seen two classes which are separated in a plane. The options are chosen RBF kernel, Cost is chosen 100. The data is recorded under the name of toy1. Table 2.3 shows the toy1 dataset.

Table 2.3 : Toy 1 dataset

Class Label	index:feature1	index:feature2
1	1:0.096	2:0.104
1	1:0.078	2:0.152
1	1:0.160	2:0.108
1	1:0.132	2:0.192
1	1:0.276	2:0.096
1	1:0.234	2:0.206
1	1:0.062	2:0.322
3	1:0.420	2:0.298
3	1:0.452	2:0.202
3	1:0.414	2:0.210
3	1:0.518	2:0.064
3	1:0.352	2:0.292
3	1:0.212	2:0.366
3	1:0.330	2:0.370
3	1:0.394	2:0.094
3	1:0.170	2:0.244

The example of training a classifier with RBF kernel, the value of Cost is chosen 100 and the other options are chosen default. The training dataset is toy1. After training dataset the results in Figure 2.11 are obtained.

```
C:\libsvm-2.89\java>java svm_train -t 2 -c 100 toy1 model
.*
optimization finished, #iter = 18
nu = 0.3140414842826386
obj = -405.1444193732676, rho = 1.0074778087108636
nSV = 6, nBSV = 3
Total nSV = 6
```

Figure 2.11 : Training results from command prompt

nu is the hyper-parameter of nu-SVC and nu-SVR. obj is the optimal objective value, rho is the bias term. nSV and nBSV are number of support vectors and bounded support vectors, respectively.

2.2.2 Pre-computed Kernel Type

There are special kernel types for some problems. Pre-computed kernel type is one of them. Pre-computed kernel is not a new type of kernel. User can calculate the kernel-matrix first, and then the option of pre-computed kernel is selected. It is no need to calculate kernel-matrix again by SVM. This way is suitable for the dataset which has a large number of features. Pre-computed kernel type can be a perfect solution for computing time-saving way.

The following format is an instance of training dataset with the option of pre-computed kernel. It is assumed that there are L training instances. K(x, y) is the kernel value of the instances x and y.

<label> 0:i 1:K(xi,x1) ... L:K(xi,xL)

The following format is an instance of testing dataset with the option of pre-computed kernel.

<label> 0:? 1:K(x,x1) ... L:K(x,xL)

? may be any number.

In Figure 2.12, linear kernel is used. The calculation of kernel-matrix is calculated as follows:

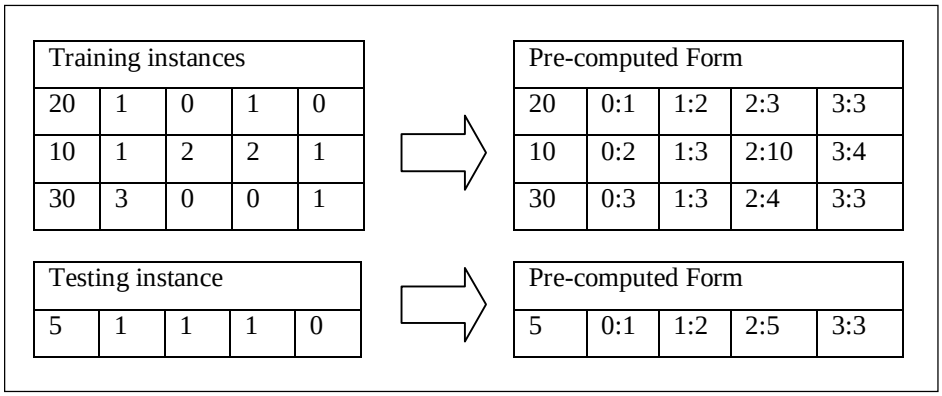


Figure 2.12 : Computation of pre-computed kernels

Computing time of the program can be decreased by using pre-computed kernel type. As mentioned in subsection 3.1.2, pre-computed kernel type considerably improves the computing time on featured datasets. If there are not many features in a dataset using this parameter could be unnecessary.

Any subset of the training dataset is usable as a new dataset. Leave-one-out cross-validation mentioned in subsection 2.3.2 is applicable for the programs with the option of pre-computed kernel. If the method leave-one-out is applied to the training dataset, the corresponding row of pre-computed form of the dataset is separated for testing as shown in Figure 2.13. In other words, for example if the second row of training dataset in Figure 2.12 is separated for testing, to separate the second row of the pre-computed form of the dataset is enough to test.

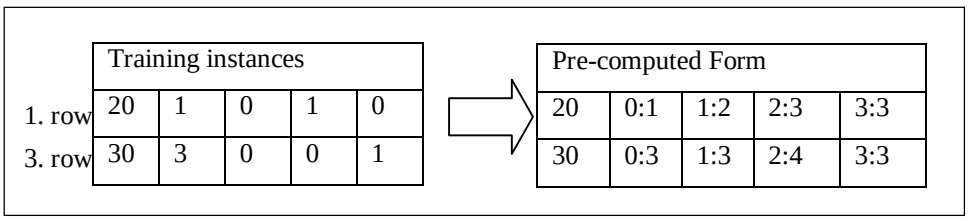


Figure 2.13 : Subset of kernel matrix

Eq. 2.34 implies that the kernel matrix in Figure 2.13.

$$\begin{bmatrix} K(1,1) & K(1,3) \\ K(3,1) & K(3,3) \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 3 & 3 \end{bmatrix} \tag{2.34}$$

2.3 OPTIMIZATION AND EVALUATION OF SVM ACCURACY

LIBSVM is received from its source and then svm_train and svm_predict classes first combined to a single class and then modified. Leave-one-out cross-validation and grid search methods are added to the program. Leave-one-out is used to show how cross-validation regression/classification is accurate. Grid search method is used to select the most appropriate parameter combination by trying all parameter combinations. Bootstrap resampling method is applied to prostate cancer data to have more accurate results.

2.3.1 *k*-fold Cross-Validation

In *k*-fold cross-validation, dataset D is divided into k subset ramdomly D_1, D_2, ..., D_k. Each subset is called as *fold*. For each of iterations one fold is separated and remaining dataset is trained.

Kohavi (1995) suggests that stratified 10-20 fold cross validation should be used for accuracy estimation, and multiple runs of 3-5 fold cross validation should be used for model selection.

2.3.2 Leave-One-Out Cross-Validation

In LOOCV, one instance is separated and the training is applied on other instances to have the model. To test the model the instance separated is used. In this method each instance is tested once. The purpose of this operation is to measure the correctness of the prediction or classification. Because of its computational cost *k*-fold cross validation applications are more advantageous then this method.

2.3.3 Grid Search Method

This method is used for finding optimal parameter values. Grid search method give better results when it is used with cross validation since the parameters (C and γ) are selected properly as a result of checking the corresponding points of the selected parameters on each instance.

In this thesis grid search method and cross-validation used together to find the best fit hyper parameter combination. The kernel type selected is RBF because of its better performance.

2.3.4 Bootstrap Resampling Method

In this method datasets generated from observations are used for having more accurate statistical predictions. Using whole data sets for predictions causes costs time and money. Therefore samples (resamples or bootstrap samples) are needed which represent data better. Here, various size and amount of data sets can be generated by resampling the observations which are randomly replaced in a dataset having any size. In this way maximum possible information can be retrieved from a given dataset. This method is developed by Bradley Efron in 1979 and named as Bootstrap (Resampling) Method.

To use the bootstrap resampling method there is no need to have a large number of samples. Besides, this method gives more accurate results compared to classical methods (Hesterberg. 2003).

2.4 DATASET DESCRIPTION

2.4.1 Synthetic Dataset

In this study, before using original dataset it is created synthetic one which are written to obtain statistical outputs. The synthetic dataset has nine variables. In the dataset, the last three columns are Glandular (G), Stromal (S) and Cancer (C) tissue percentages. The synthetic dataset consists of six metabolites. The algorithm provided in Figure 2.14, is written for G and must be applied for both S and C to complete one cycle of the synthetic dataset instance generator (each cycle generates 12 samples).

Algorithm: Generating Synthetic Dataset

Input: N, number of instance

Output: D, a set of dataset

Method:

1. **Do for** $i = 1, 2, \dots, N$
2. **For each** possible order of G (G, S, C or G, C, S etc...)
3. Choose random $G \in [0, 1]$
4. **For each** metabolites of G (Ga or Gb)
5. (if Ga is random then) Compute $\underset{Gb = \frac{G-Ga}{1-2.Ga}}{Ga \in ([0, 0.45] \| [0.55, 1])}$
6. Add Ga and Gb to D
7. Choose random $R \in [0, 1]$
8. Compute $S = R.(1 - G)$
9. Compute $C = (1 - G - S)$
10. Add for each tissue (G, S, C) to D

Figure 2.14 : Algorithm of generating synthetic dataset

In creating the synthetic dataset the following equations and the algorithm are used;

$$(G + S + C) = 1 \tag{2.35}$$

The sum of tissues is equal to 1 according to the Eq. 2.34. R is a random double. In Figure 2.14, Step 3 indicates that G is generated as a random variable in between 0 and 1. Also Step 8 indicates that S is connected to G and also S is a random number, too. Moreover, Eq. 2.34 and Step 9 shows that C is connected to both G and S together. C is calculated by subtracting the sum of the others from 1.

This algorithm is repeated for every permutation of the configuration of each tissue. In other words, each round of repetitive the arrangement of tissues changes, such as $G, S, C - G, C, S - S, G, C - S, C, G - C, G, S - C, S, G$. For example, in the fourth round of algorithm, line 3 is changed as $S \in [0, 1]$, Step 8 is changed as $C = R.(1 - S)$ and Step 9 is changed as $G = (1 - S - C)$ for creating an instance. Ga, Gb, Sa, Sb, Ca

and Cb are metabolites belong to the specific tissues; G, S and C respectively. Each tissue type has two metabolites.

$$G = Ga + Gb - 2GaGb \tag{2.36}$$

The Eq. 2.35 is repeated for G, S and C, separately. Eq. 2.34 and Eq. 2.35 are the main equations of the synthetic dataset. At every turn of algorithm in Figure 2.14, one of the synthetic metabolites of a specific tissue must be chosen randomly. The equation on Step 5 is repeated for every random metabolites of each tissue. To place the variables (tissues and metabolites) in the dataset homogeneously is aimed for creating the dataset. As a result 12 different instances are created from many different possible cases.

According to Step 5 if Ga is chosen randomly then Ga can be in between 0 and 0.45 or 0.55 and 1. The range from 0.45 to 0.55 is not used as a rule. The border values are not included. If Ga is chosen randomly then Gb can not be a random number or vice versa. In other words, for a minimal dataset Gb is selected randomly to create the minimal dataset instances. This process is repeated for every metabolites of each tissue.

The minimal dataset must have 12 instances because of the random cases. Therefore the number of instances in a dataset must be multiples of 12 e.g. 12, 24, 48… In other words, the dataset is used as the form of multiples of 12 clusters. It is produced the same type instances as requested. For example; if the number of instance is 24, then consequently each instance type is used twice in the dataset. Each cluster of the synthetic dataset is generated randomly. In other words, the same dataset cluster is never used twice.

Table 2.4 shows the real copy of the generated data which has 12 instances, the minimal dataset cluster. If Table 2.4 is examined the connectivity of the synthetic tissue types and metabolites can be seen among themselves and separately.

Table 2.4 : Generated the minimal synthetic dataset

Ga	Gb	Sa	Sb	Ca	Cb	G	S	C
0,71261	0,89902	0,39118	-0,84561	0,7183	0,58585	0,33033	0,20715	0,46252
-0,05582	0,61414	0,87477	0,74327	2,71565	0,60032	0,62689	0,31766	0,05545
0,30382	-0,54957	0,26711	0,29113	0,37002	0,53499	0,08819	0,40271	0,5091
0,58797	0,3769	-0,40192	0,23069	0,57408	0,74207	0,52166	0,01421	0,46413
0,24505	0,71541	0,34571	-1,07703	0,5993	1,12039	0,60984	0,01337	0,37679
-1,11473	0,55234	-0,86294	0,37033	-0,28131	0,29806	0,66902	0,14653	0,18445
0,42123	1,65192	0,43276	-3,1442	0,32384	-0,04331	0,68147	0,00995	0,30858
0,33696	0,3276	1,80315	0,68274	0,29844	0,58059	0,44378	0,02373	0,53249
0,35266	0,74987	0,42411	-2,01594	0,24358	0,12608	0,57363	0,11813	0,30824
4,39058	0,55432	0,35337	0,59769	0,15724	0,34535	0,07737	0,52865	0,39399
0,71662	1,57252	0,74875	0,21999	0,61205	1,27928	0,03533	0,63931	0,32536
1,57438	0,7278	-1,03319	0,39504	-0,0743	0,77105	0,01051	0,17816	0,81134

2.4.2 Real-World Datasets

2.4.2.1 Colon (Colorectal) cancer dataset

Colon cancer is a disorder which influences colon and rectum. The colon is the part of the digestive system. Tumors of the colon are growths arising from the inner wall of the large intestine which is a muscular tube from the small intestine to the rectum. Benign tumors of the large intestine are called polyps. Malignant tumors of the large intestine are called cancers. Benign polyps do not spread to other parts of the body. Benign polyps are not life-threatening but if benign polyps are not removed from the body, they can become cancerous over time. Cancer of the colon can damage adjacent tissues. Cancer cells can also break away and spread to other organs[6]. Colon cancer is the fourth leading cause of cancer in Turkish men and the second cause of cancer in Turkish females according to Turkish Statistical Institute (2010).

In this section, the colon cancer dataset is used for classification. The corresponding dataset has 33 instances and 4 features (obtained from an original 188 features). Each instance is obtained from biopsy of a human being creating the class label (0 or 1) for that instance as healthy or unhealthy.

[6] http://www.medicinenet.com/colon_cancer/article.htm [cited December, 2010]

2.4.2.2 Leukemia dataset

Leukemia is a cancer type that starts in the tissue that forms blood. To understand cancer, it is needed to know how normal blood cells form. Most blood cells develop from cells in the bone marrow called stem cells. Bone marrow is the soft material in the center of most bones. Stem cells mature into different kinds of blood cells. Each kind has a special job: White blood cells help fight infection, red blood cells carry oxygen to tissues throughout the body and platelets help form blood clots that control bleeding. White blood cells, red blood cells, and platelets are made from stem cells as the body needs them. When cells grow old or get damaged, they die, and new cells take their place. In a person with leukemia, the bone marrow makes abnormal white blood cells which are called leukemia cells. Unlike normal blood cells, leukemia cells don't die when they should. They may crowd out normal white blood cells, red blood cells, and platelets. This makes it hard for normal blood cells to do their work[7].

Leukemia dataset (Golub *et al.* 1999) [8] is especially chosen to show how pre-computed kernel option saves computing time according to the other kernel types of SVM. Leukemia dataset has 38 instances, 2 classes (1 and -1) and 7129 features.

2.4.2.3 Prostate cancer dataset

Prostate cancer is a malignant tumor that consists of cells from the prostate gland that is located at the base of the urinary bladder. Also prostate gland is an organ that surrounds the first part of the urethra. All prostate cancers do not behave similarly. Some types of prostate cancer grow slowly and produce no appreciable signs for many years. But some aggressive types grow and spread more rapidly than others. This type cancer can cause a significant shortening of life expectancy in men affected by them. As the cancer advances, however, it can spread beyond the prostate. Moreover, the cancer also can spread throughout other areas of the body, such as the bones, lungs, and liver.

The cause of prostate cancer is unknown. The risk factors for prostate cancer include age, genetics, hormones and such environmental factors. The chances of developing

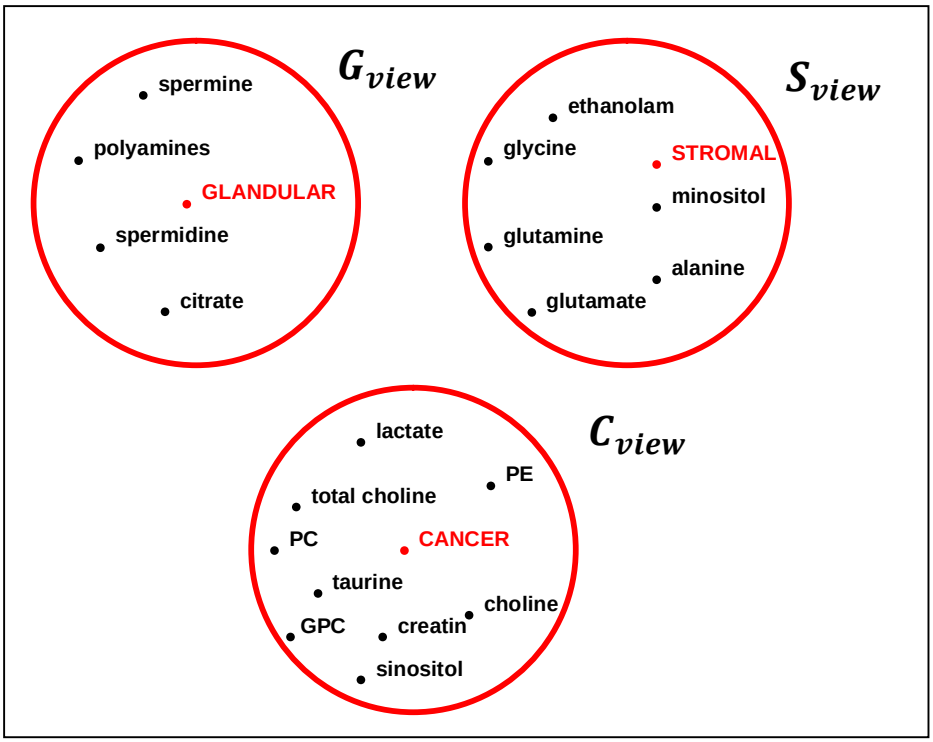
[7] Source at: http://www.medicinenet.com/leukemia/article.htm [cited December, 2010]
[8] Available at: http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/ [cited October, 2010]

prostate cancer increase with age. Prostate cancer is common over the age of 80, while rarely seen in men younger than 40.[9] Prostate cancer is the second leading cause of deaths from cancer in Turkish men according to Turkish Statistical Institute (2010).

PIMR method is reconstructed for the prostate cancer dataset. The purpose of using the prostate cancer dataset is to obtain results on a real dataset. This dataset has 130 rows and 22 columns. The first three columns of the dataset indicate Glandular, Stromal and Cancer tissue percentages, respectively. The other 19 columns are metabolites which belong to only one of the views as indicated by an array that we called the partition table. Originally, we use the partition that we are given with by the UNC biomedical engineering group (see Figure 2.15; Keshari *et al.* 2009). In the dataset, each row is an instance which is obtained from biopsy of a different healthy or unhealthy person. As the testset we use the whole database. The training dataset is generated using bootstrap resampling method out of the whole dataset. In other words, the training dataset is a random subset of the testing dataset (not all test samples are used during training and this process is repeated a number of times for statistical significance).

As mentioned above the total number of metabolites is 19. Glandular tissue type has 4 metabolites, Stromal has 6, and cancer has 9 metabolites. Figure 2.15 shows the cluster of tissues with their members.

[9] Source at: http://www.medicinenet.com/prostate_cancer/article.htm [cited September, 2010]



Source: (Adapted from Keshari *et al.* 2009)

Figure 2.15 : Spectral clustering of views

2.5 PARALLEL INTERACTING MULTI-VIEW REGRESSION (PIMR) ON SYNTHETIC DATA

In this section, to motivate it, PIMR is first tried on a synthetic dataset. The generated training and test datasets have instances with 3 target variables and six features, the features are organized into 3 views (2 features per target variable), as shown in Figure 2.16.

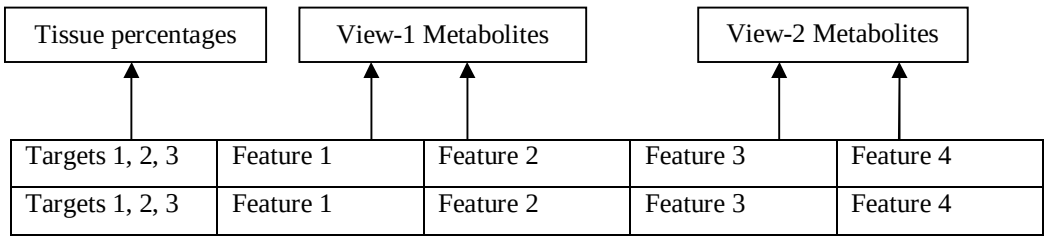


Figure 2.16 : Structure of instances (two instances and two views are shown)

2.5.1 Training and Testing Phases of PIMR on Synthetic Data

SVM involves training and testing phases. In other words, PIMR contains these phases in itself. In the training phase, the training dataset is used, but in the testing phase both the training and testing datasets are used. The main purpose of using the training dataset is to obtain a regressor which is used to obtain a predicted output/prediction in testing phase. By establishing a relationship between target and features of a view, SVM creates a function called regressor in regression. A separate regressor is produced for each view. The target is estimated using features and predictions of other views (Figure 2.17).

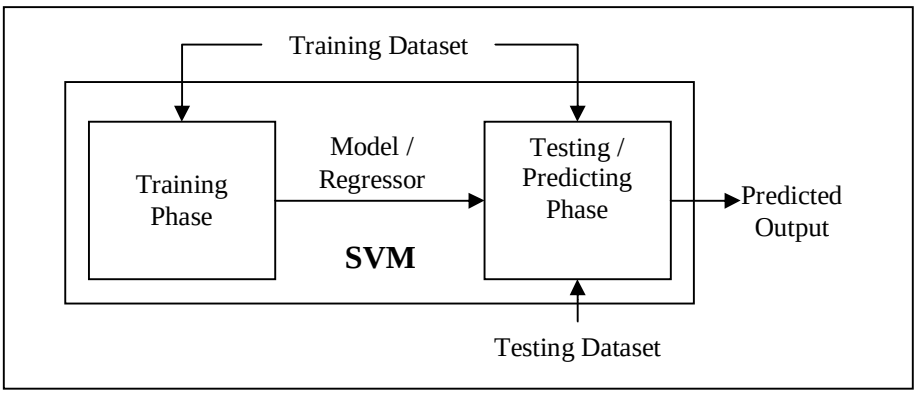


Figure 2.17 : Training and testing phases of PIMR

PIMR algorithm is divided into two parts as training and testing processes. In Figure 2.18 the training processes of PIMR algorithm on synthetic data is provided.

Algorithm: PIMR Training Algorithm

Input: K: sequence #

Method:

1. **Do for** $k = 1, 2 \ldots K$:
2. Randomly generate TD^k: Training dataset //$12 \times k$ instances are generated // k is chosen 25
3. $G_0, S_0, C_0 = 0$;
4. **Do for** $i = 0, 1 \ldots I$: { //iteration
5. $[model: M_{G,i},\ predicted\ output: G_{i+1}]^k \leftarrow SVM\_learn(input: [Ga,\ Gb,\ S_i,\ C_i],\ T_G)^k$
   $[model: M_{S,i},\ predicted\ output: S_{i+1}]^k \leftarrow SVM\_learn(input: [Sa,\ Sb,\ G_i,\ C_i],\ T_S)^k$
   $[model: M_{C,i},\ predicted\ output: C_{i+1}]^k \leftarrow SVM\_learn(input: [Ca,\ Cb,\ G_i,\ S_i],\ T_C)^k$
6. **Output** $RMSE(G_i),\ RMSE(S_i),\ RMSE(C_i), r_{G,i}^2, r_{S,i}^2\ r_{C,i}^2, \sigma(RMSE(G_i)), \sigma(RMSE(S_i)), \sigma(RMSE(C_i)), \sigma(r_{G,i}^2), \sigma(r_{S,i}^2), \sigma(r_{C,i}^2)$
7. $[G_{i+1} \leftarrow G_i\ in\ G_{view}]^k$
   $[S_{i+1} \leftarrow S_i\ in\ S_{view}]^k$
   $[C_{i+1} \leftarrow C_i\ in\ C_{view}]^k$ } //end for

Figure 2.18 : Training algorithm of PIMR on synthetic data

The training phase of PIMR algorithm the training dataset which involves generated $12 \times k$ instances is trained and a regressor/model is generated to use for testing phase of SVM. This process is repeated for every views of training dataset and for three iterations of PIMR. k is also used for the sequence number of training and testing datasets. In other words, the number k shows the current processes and instance quantity of the datasets. G_0, S_0 and C_0 variables in testing dataset initialize to 0 (step 3). The iterations start with step 4 in Figure 2.18. The next part of the algorithm is described in more detail in conjunction with the testing algorithm. In Figure 2.19 the testing processes of PIMR algorithm on synthetic data is provided.

Algorithm: PIMR Testing Algorithm

Input: K: sequence #

Randomly generate PD: Predicting dataset // instance number is chosen 2400

Method:

1. **Do for** $k = 1, 2 \ldots K$:
2. $G_0, S_0, C_0 = 0$;
3. **Do for** $i = 0, 1 \ldots I$: { //iteration
4. $[\,predicted\ output: G_{i+1}\,]^k \leftarrow SVM\_learn\big(input: [Ga, Gb, S_i,\ C_i\ ],\ T_G,\ \ M_{G,i}\big)^k$

   $[\,predicted\ output: S_{i+1}\,]^k \leftarrow SVM\_learn\big(input: [Sa, Sb, G_i,\ C_i\ ],\ T_S,\ \ M_{S,i}\big)^k$

   $[\,predicted\ output: C_{i+1}\,]^k \leftarrow SVM\_learn\big(input: [Ca, Cb, G_i,\ S_i\ ],\ T_C,\ \ M_{C,i}\big)^k$
5. **Output** $RMSE(G_i),\ RMSE(S_i),\ RMSE(C_i), r^2_{G,i}, r^2_{S,i}\ r^2_{C,i}, \sigma\big(RMSE(G_i)\big), \sigma\big(RMSE(S_i)\big),$ $\sigma\big(RMSE(C_i)\big), \sigma\big(r^2_{G,i}\big), \sigma\big(r^2_{S,i}\big), \sigma\big(r^2_{C,i}\big)$
6. $[G_{i+1} \leftarrow G_i\ in\ G_{view}]^k$

   $[S_{i+1} \leftarrow S_i\ in\ S_{view}]^k$

   $[C_{i+1} \leftarrow C_i\ in\ C_{view}]^k$
7. Compute $E_i = \sqrt{\left[\frac{\sum_{i=1}^{K}\left(1-(G_i+S_i+C_i)\right)^2}{K}\right]}$ // RMSE of total error for $G + S + C$
8. **Output** $RMSE(E_i), \sigma\big(RMSE(E_i)\big)$ **}** //end for

Figure 2.19 : Testing algorithm of PIMR on synthetic data

A testset with 2400 instances is generated. In other words, in Figure 2.19, $k = 0$ means minimal synthetic dataset with 12 samples, $k = 1$ means the dataset which consist 24 instances. K indicates the instance quantity of the last dataset. G_0, S_0, C_0 variables in testing dataset initialize to 0 (step 2). In step 4 iterations start.

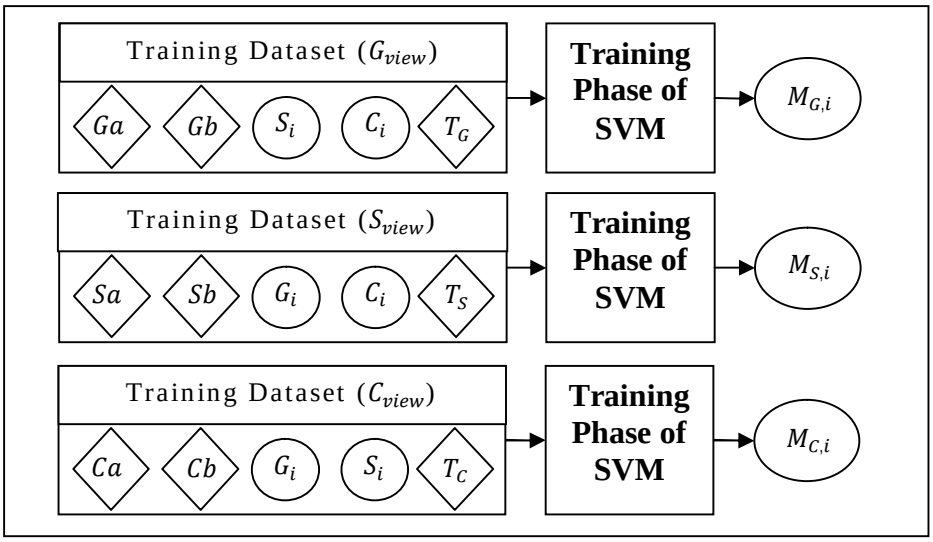


Figure 2.20 : Training phase of views on synthetic data

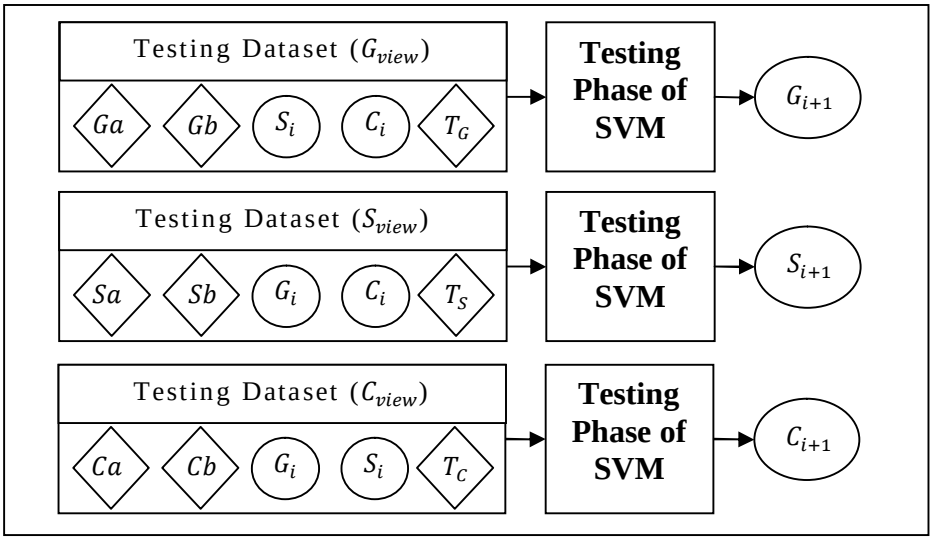


Figure 2.21 : Testing phase of views on synthetic data

In Figure 2.20 and Figure 2.21 G_{view}, S_{view} and C_{view} in SVM learning (training ve testing phases) processes are shown. The spiral structure of PIMR is expressed by the iterations. PIMR has three iterations and the mutual interaction of views is assumed to be in a stable condition at the end of these iterations. In general, the purpose of the iterations can be summarized as follows; the output of the first iteration is the input of the second one and the output of the second iteration is the input of the third one. Thus, it can be declared that the method has affected items from each other. It is assumed the

mutual interaction is stable after the third iteration. The views run in parallel. Based on Figure 2.20 and Figure 2.21 above the following results can be summarized.

- In the 1st iteration, the training dataset (training $G_{view,1}$) (i.e. target T_G and the features $Ga, Gb,\ S_0,\ C_0$) is trained in the training phase and a regressor ($M_{G,0}$) is generated. The same training dataset is tested using the regressor $M_{G,0}$ in the testing phase and training G_1 of training dataset is generated as predicted output (Figure 2.20). In the second side of the 1st iteration, the testing dataset (testing $G_{view,1}$) i.e. the target T_G and the features $Ga, Gb,\ S_0,\ C_0$ are tested with the regressor that generated previously and then G_1 of testing dataset is obtained as predicted output in testing phase (Figure 2.21). The outputs are stored for use in the 2nd iteration.
- In the 1st iteration, the training dataset (training $S_{view,1}$) (i.e. target T_S and the features $Sa, Sb,\ G_0,\ C_0$) is trained in the training phase and a regressor ($M_{S,0}$) is generated. The same training dataset is tested using the regressor $M_{S,0}$ in the testing phase and S_1 of training dataset is generated as predicted output (Figure 2.20). In the second side of the 1st iteration, the testing dataset (testing $S_{view,1}$) i.e. the target T_S and the features $Sa, Sb,\ G_0,\ C_0$ are tested with the regressor that generated previously and then S_1 of testing dataset is obtained as predicted output in testing phase (Figure 2.21). The outputs are stored for use in the 2nd iteration.
- In the 1st iteration, the training dataset (training $C_{view,1}$) (i.e. target T_C and the features $Ca, Cb,\ G_0,\ S_0$) is trained in the training phase and a regressor ($M_{C,0}$) is generated. The same training dataset is tested using the regressor $M_{C,0}$ in the testing phase and C_1 of training dataset is generated as predicted output (Figure 2.20). In the second side of the 1st iteration, the testing dataset (testing $C_{view,1}$) i.e. the target T_C and the features $Ca, Cb,\ G_0,\ S_0$ are tested with the regressor that generated previously and then C_1 of testing dataset is obtained as predicted output in testing phase (Figure 2.21). The outputs are stored for use in the 2nd iteration.

The training and predicting processes are actualized in SVM. The 2nd iteration serves as a link between the 1st and the 3rd iteration. The specific iteration uses the predicted outputs of the 1st iteration as inputs and generates outputs which are inputs for the 3rd iteration. Based on Figure 2.20 and Figure 2.21 the following results can be summarized.

- In the 2nd iteration, the features G_0, S_0 are removed and the generated targets S_1, C_1 are added to training $G_{view,2}$ and the constant features Ga, Gb with the target T_G are trained in the training phase and a regressor ($M_{G,1}$) is generated. The same training $G_{view,2}$ is tested using the regressor $M_{G,1}$ in the testing phase and G_2 of training dataset is generated as predicted output (Figure 2.20). In the second side of the 2nd iteration, the features G_0, S_0 are removed and the targets S_1, C_1 incoming from the previous iteration are added to training $G_{view,2}$ and the features Ga, Gb with the target T_G using the regressor $M_{G,1}$ are tested all together and then testing G_2 is obtained as predicted output in testing phase (Figure 2.21). The outputs are stored for use in the 3rd iteration.
- In the 2nd iteration, the features G_0, C_0 are removed and the generated targets G_1, C_1 are added to training $S_{view,2}$ and the constant features Sa, Sb with the target T_S are trained in the training phase and a regressor ($M_{S,1}$) is generated. The same training $S_{view,2}$ is tested using the regressor $M_{S,1}$ in the testing phase and S_2 of training dataset is generated as predicted output (Figure 2.20). In the second side of the 2nd iteration, the features G_0, C_0 are removed and the targets G_1, C_1 incoming from the previous iteration are added to training $S_{view,2}$ and the features Sa, Sb with the target T_S using the regressor $M_{S,1}$ are tested all together and then testing S_2 is obtained as predicted output in testing phase (Figure 2.21). The outputs are stored for use in the 3rd iteration.
- The 2nd iteration processes of C_{view} are similar to G_{view} and S_{view}.

The same processes of the 2nd iteration are acceptable for the 3rd and the other iterations of PIMR. Figure 2.14 shows a complete structure of PIMR on synthetic dataset.

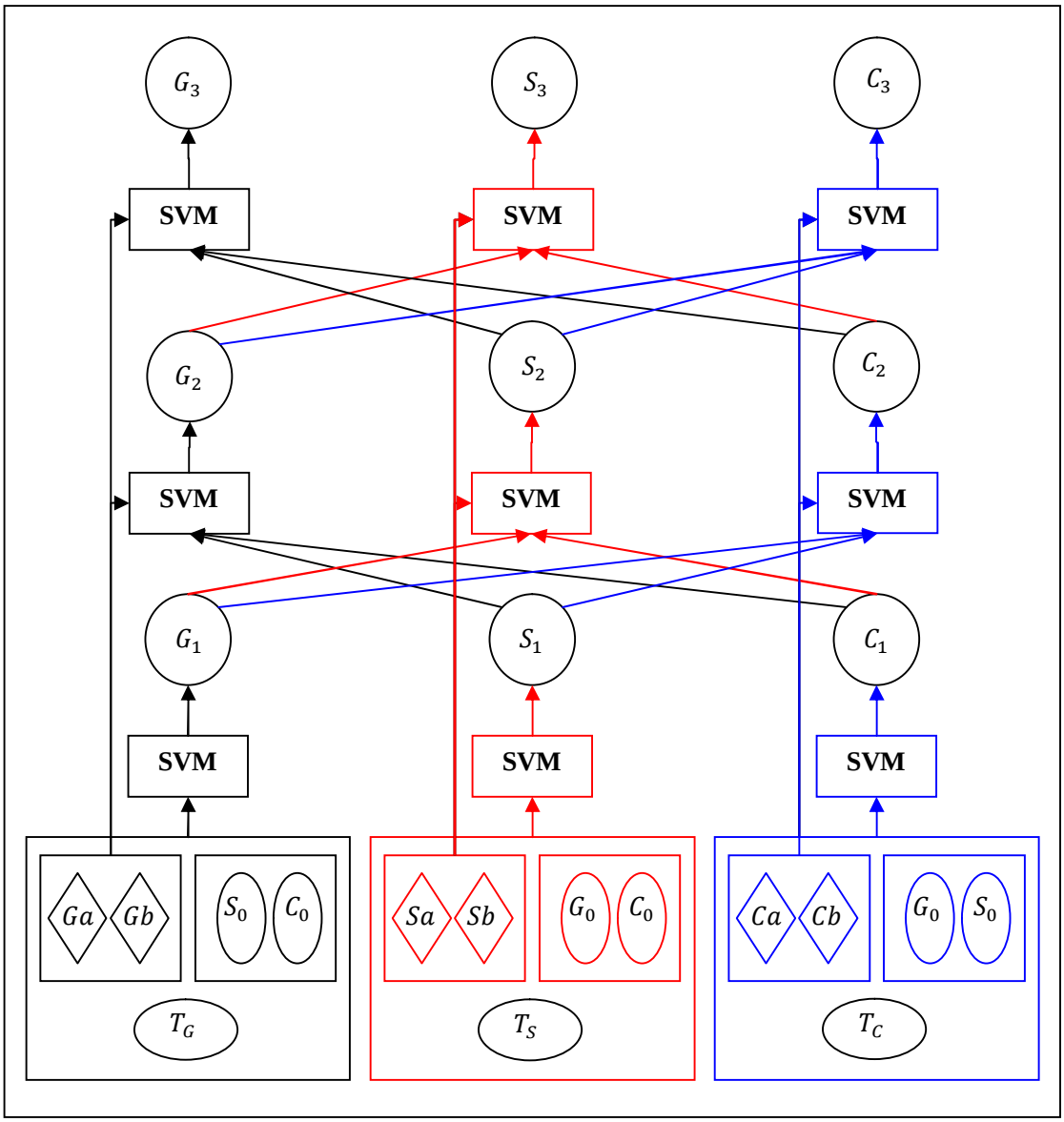


Figure 2.22 : Complete picture of PIMR

2.5.2 Classical Single-View Regression on Synthetic Data

In Figure 2.23 the training and testing processes of CSR algorithm on synthetic data is provided.

Algorithm: CSR algorithm

Input: K: sequence #, TD^k, PD

Method:

1. **Do for** $k = 1, 2 \ldots K$:
2. $[\, model\colon M_G,\ predicted\ output\colon G\,]^k \leftarrow SVM\_learn(input\colon [Ga, Gb, Sa, Sb, Ca, Cb\,], T_G\ )^k$
   $[\, model\colon M_S,\ predicted\ output\colon S\,]^k \leftarrow SVM\_learn(input\colon [Ga, Gb, Sa, Sb, Ca, Cb\,], T_S\ )^k$
   $[\, model\colon M_C,\ predicted\ output\colon C\,]^k \leftarrow SVM\_learn(input\colon [Ga, Gb, Sa, Sb, Ca, Cb\,], T_C\ )^k$
3. **Output** $RMSE(G),\ RMSE(S),\ RMSE(C), r_G^2, r_S^2\ r_C^2, \sigma\big(RMSE(G)\big), \sigma\big(RMSE(S)\big),$
   $\sigma\big(RMSE(C)\big), \sigma(r_G^2), \sigma(r_S^2), \sigma(r_C^2)$**}** //end for

Figure 2.23 : Algorithm of classical single-view regression on synthetic data

In Figure 2.23, below the processes are applied k times. The outputs obtained are replaced with the previous target and used as input to the next. In other words, G_1 replaces G_0 at the second time of loop. Values and places of other variables in G_{view} do not change.

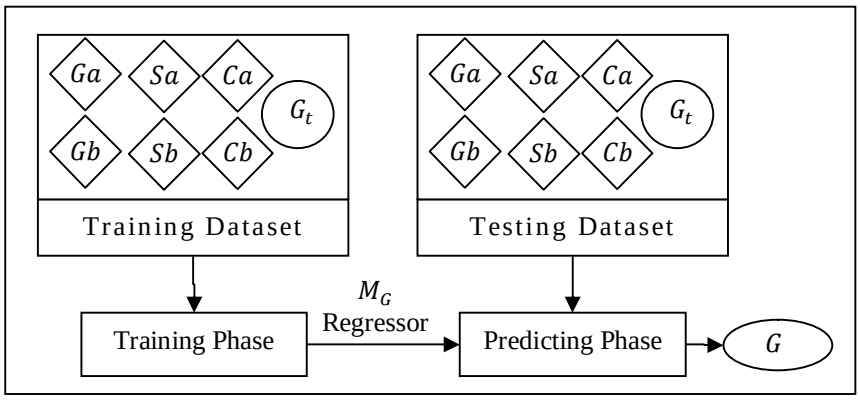


Figure 2.24 : G_{view} on synthetic data for CSR

In Figure 2.24, classical single view regression is shown for G_{view}. The same figure is acceptable for a few changes to S_{view}, and C_{view}. Classical single-view regression can be summarized as below.

- Training G_{view} (i.e. Ga, Gb, Sa, Sb, Ca, Cb and T_G) is trained in the training phase and the regressor M_G is generated. Testing G_{view} is tested using M_G in testing phase and G_1 is generated.
- Training S_{view} (i.e. Ga, Gb, Sa, Sb, Ca, Cb and T_S) is trained in the training phase and the regressor M_S is generated. Testing S_{view} is tested using M_S in testing phase and S_1 is generated.
- Training C_{view} (i.e. Ga, Gb, Sa, Sb, Ca, Cb and T_C) is trained in the training phase and the regressor M_G is generated. Testing C_{view} is tested using M_C in testing phase and C_1 is generated.

The targets (T_G, T_S and T_C) are also delivered to the testing phase and compared with produced G_1, S_1 and C_1 values and statistical results are obtained. Then root mean squared error and correlation coefficient values are obtained. The results will be avaluated in the section experimental results.

2.6 PIMR ON SYNTHETIC PROSTATE CANCER DATASET

In Figure 2.25 the training processes of PIMR algorithm on the synthetic prostate cancer data is provided. The pseudocode of PIMR algorithm is included in Appendix. In Figure 2.25, the random set of integers is generated for specifying which instances belong to the training dataset. k indicates the sequence number of the processes and the instance quantity of the datasets. These random integers are generated for every random training dataset TD^k, which contains k units instance in itself. The metabolites are partitioned into views corresponding to the partition table PT. We do these experiments to show that when the natural organization of the variables into views are not preserved, the multi-view approach does not have the advantage over CSR.

Algorithm: PIMR Training Algorithm

Input: K: sequence #, D: database, PT: partition table

Method:

1. Generate randomly RS: Random set of integers // each of these numbers indicates the number of a row in D.
2. **Do for** $k = 1, 2 \ldots K$:
3. Generate random training dataset TD^k subset of D using RS, denominate columns using PT
4. $G_0, S_0, C_0 = 0$;
5. **Do for** $i = 0, 1 \ldots I$: { //iteration
6. $[model: M_{G,i},\ predicted\ output: G_{i+1}]^k \leftarrow SVM\_learn(input: [G_{metabolites},\ S_i,\ C_i],\ T_G)^k$

   $[model: M_{S,i},\ predicted\ output: S_{i+1}]^k \leftarrow SVM\_learn(input: [S_{metabolites},\ G_i,\ C_i],\ T_S)^k$

   $[model: M_{C,i},\ predicted\ output: C_{i+1}]^k \leftarrow SVM\_learn(input: [C_{metabolites},\ G_i,\ S],\ T_C)^k$
7. **Output** $RMSE(G_i),\ RMSE(S_i),\ RMSE(C_i), r^2_{G,i}, r^2_{S,i}\ r^2_{C,i}, \sigma\big(RMSE(G_i)\big), \sigma\big(RMSE(S_i)\big), \sigma\big(RMSE(C_i)\big), \sigma\big(r^2_{G,i}\big), \sigma\big(r^2_{S,i}\big), \sigma\big(r^2_{C,i}\big)$
8. $[G_{i+1} \leftarrow G_i\ in\ G_{view}]^k$

   $[S_{i+1} \leftarrow S_i\ in\ S_{view}]^k$

   $[C_{i+1} \leftarrow C_i\ in\ C_{view}]^k$ } //end for

Figure 2.25 : Training algorithm of PIMR on prostate cancer dataset

In Figure 2.26 the testing processes of PIMR algorithm on prostate cancer data is provided.

Algorithm: PIMR Testing Algorithm

Input: K: sequence #, D: database, PT: partition table

Method:

4. Generate predicting PD using the whole D, denominate columns using PT
5. **Do for** $k = 1, 2 \dots K$:
6. $G_0, S_0, C_0 = 0$;
7. **Do for** $i = 0, 1 \dots I$: { //iteration
8. $[\,predicted\ output: G_{i+1}\,]^k \leftarrow SVM\_learn\big(input: [G_{metabolites},\ S_i,\ C_i\ ],\ T_G,\ M_{G,i}\big)^k$

   $[\,predicted\ output: S_{i+1}\,]^k \leftarrow SVM\_learn\big(input: [S_{metabolites},\ G_i,\ C_i\ ],\ T_S,\ M_{S,i}\big)^k$

   $[\,predicted\ output: C_{i+1}\,]^k \leftarrow SVM\_learn\big(input: [C_{metabolites},\ G_i,\ S_i\ ],\ T_C,\ M_{C,i}\big)^k$
9. **Output** $RMSE(G_i),\ RMSE(S_i),\ RMSE(C_i), r_{G,i}^2, r_{S,i}^2\ r_{C,i}^2, \sigma\big(RMSE(G_i)\big), \sigma\big(RMSE(S_i)\big), \sigma\big(RMSE(C_i)\big), \sigma\big(r_{G,i}^2\big), \sigma\big(r_{S,i}^2\big), \sigma\big(r_{C,i}^2\big)$
10. $[G_{i+1} \leftarrow G_i\ in\ G_{view}]^k$

    $[S_{i+1} \leftarrow S_i\ in\ S_{view}]^k$

    $[C_{i+1} \leftarrow C_i\ in\ C_{view}]^k$ } // end for
11. Compute $E_i = \sqrt{\left[\frac{\sum_{i=1}^{K}(1-(G_i+S_i+C_i))^2}{K}\right]}$ // RMSE of total error for $G + S + C$
12. **Output** $RMSE(E_i), \sigma\big(RMSE(E_i)\big)$ **}** //end for

Figure 2.26 : Testing algorithm of PIMR on prostate cancer data

The iteration processes of training and testing algorithms of PIMR on prostate cancer are visualized in Figure 2.27 (G_{view} is chosen for illustration).

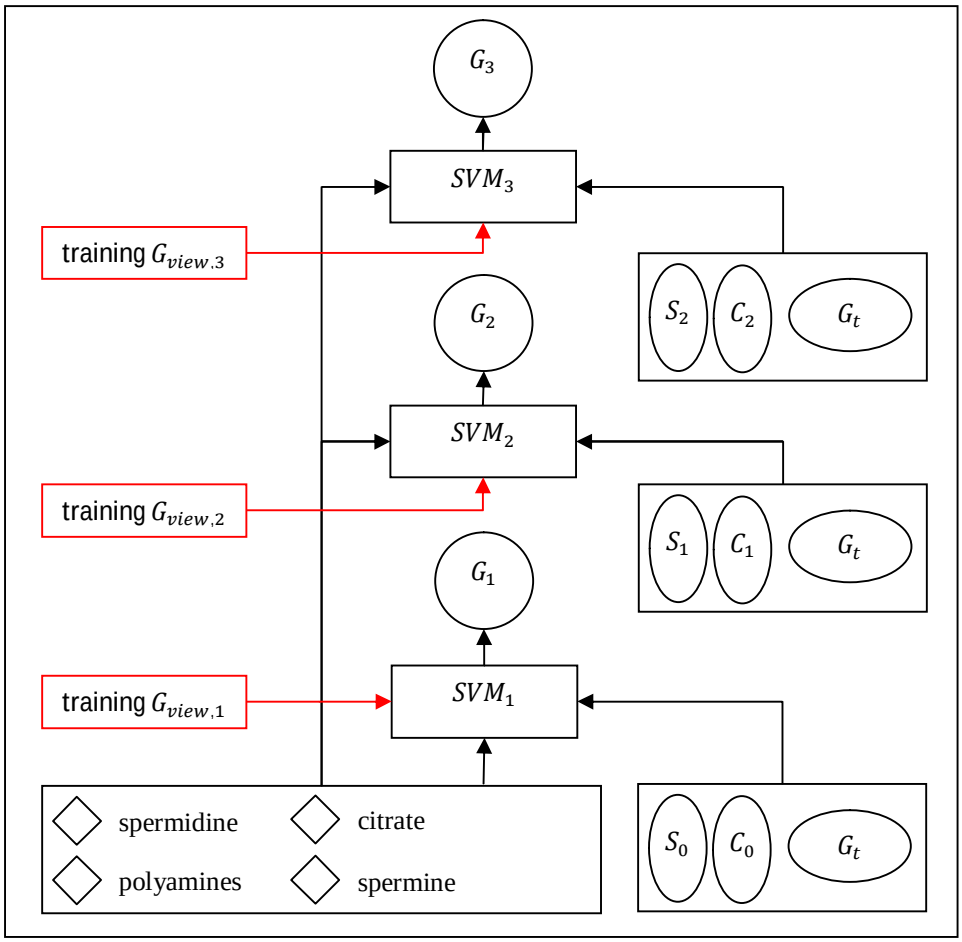


Figure 2.27 : Iterations of G_{view} on prostate cancer data

Figure 2.27 is summarized as below.

- In the 1st iteration, training $G_{view,1}$ (i.e. the target T_G and the features S_0, C_0, $spermidine$, $citrate$, $spermine$ and $polyamines$) is trained in the training phase of SVM_1 and $M_{G,1}$ (regressor/model of Glandular) is generated. In the other side of the 1st iteration, G_1 is obtained as output from testing $G_{view,1}$ in the testing phase using$M_{G,1}$. Testing $G_{view,1}$ is just used for testing.
- In the 2nd iteration, training $G_{view,2}$ (i.e. the target T_G and the features S_1, C_1, $sermidine$, $citrate$, $spermine$ and $polyamines$) is trained in the training phase of SVM_2 and $M_{G,2}$ (regressor/model of Glandular) is generated. In the other side of the 2nd iteration, G_2 is obtained as output from testing $G_{view,2}$ in the testing phase using $M_{G,2}$. Testing $G_{view,2}$ is just used for testing.

- In the 3rd iteration, training $G_{view,3}$ (i.e. the target T_G and the features S_2, C_2, $spermidine$, $citrate$, $spermine$ and $polyamines$) is trained in the training phase of SVM_3 and $M_{G,3}$ (regressor/model of Glandular) is generated. In the other side of the 3rd iteration, G_3 is obtained as output from testing $G_{view,3}$ in the testing phase using $M_{G,3}$. Testing $G_{view,3}$ is just used for testing.

Each of training dataset mentioned above is tested using the corresponding regressor and then the generated output is stored to feed the other views. The target and metabolites have constant values for all iterations.

G_{view}, S_{view} and C_{view} run in parallel, but there is not a connection between the views, so it is called single view. The metabolites are the same for each of the views but the targets are different so the generated regressors are different, too.

3. EXPERIMENTAL RESULTS

3.1 PRELIMINARY RESULTS

3.1.1 Analysis of Accuracy Results on Colon Cancer Dataset

Leave-one-out cross-validation (LOOCV) method is used on colon cancer dataset. In classification mode, each time 32 of 33 samples are trained, remaining one is tested according to LOOCV. Grid search method is applied to classification program for increasing the accuracy of SVM. In other words, grid search method is applied changing both gamma and cost values. Also, the option pre-computed kernel is selected to apply the program. The Java version of LIBSVM does not include the grid search method and LOOCV. It is included cross validation method but the method is not useful for the aim of the program. Grid search yields to select the optimal parameter for obtaining the better results. LIBSVM reads the data from file. It causes waste of time. So it is preferred to transfer the data to an array first. The data is transmitted from the array. Table 3.1 provides the results of grid search method.

Table 3.1 : Accuracy scores of grid search method on colon cancer data

Cost \ Gamma	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0.01	13	14	17	17	17	17	17	17	17	17
0.02	13	14	17	17	17	17	17	17	17	17
0.04	13	14	17	17	17	17	17	17	17	17
0.08	13	14	17	17	17	17	17	17	17	17
0.16	20	23	26	26	24	24	22	21	20	20
0.32	27	28	29	29	29	28	28	27	26	26
0.64	28	29	29	29	29	29	30	30	30	30
1.28	27	28	28	27	28	29	29	29	29	29
2.56	28	25	26	28	29	29	29	29	29	29
5.12	26	25	28	28	30	30	29	29	29	29
10.24	25	27	29	29	30	30	29	29	29	29
20.48	26	28	29	29	30	30	29	29	29	29
40.96	27	29	29	29	30	30	29	29	29	29
81.92	28	29	29	29	30	30	29	29	29	29
163.8	28	29	29	29	30	30	29	29	29	29

Table 3.1 has 10 columns and 15 rows which belong to γ and C parameters, respectively. The table shows the accuracy of LOOCV on classification.

The type of SVM is C-SVC and kernel type is pre-computed, the other options are chosen default. Figure 3.1 shows the results are better if the C is selected higher than 5.12 and γ is selected in the range of 0.5 and 0.6.

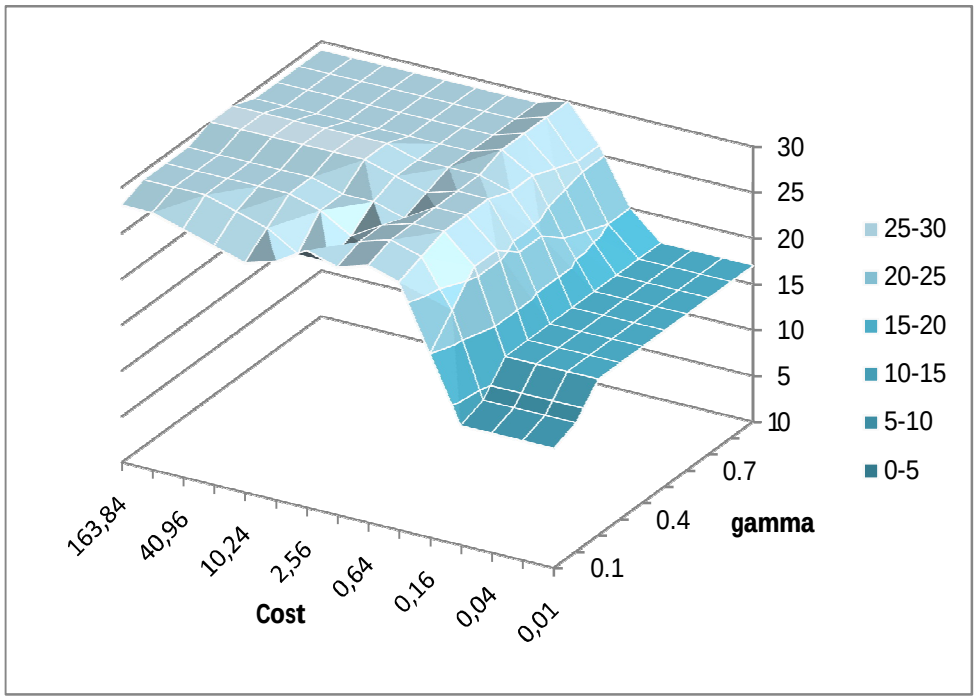


Figure 3.1 : Accuracy scores of grid search method on colon cancer data

3.1.2 Saving Computing Time with Pre-computed Kernel Type

Pre-computed kernel option is useful for saving of computing time in a dataset which has many features. Studies show that classification accuracies are achieved by using grid search method where pre-computed kernel type is selected. LOOCV method is used for both programs. One of the 62 instances is separated and 61 instances are trained. Then the instance separated is used for testing. This process is repeated until all instances are trained. Svm-type, C-SVC and other parameters are selected as default.

Table 3.2 : Comparison of the methods

Methods	**Grid search and leave one out**	**Precomputed kernel type, grid search and leave one out**	**Improvement**
Time	441444 ms (7.36 min.)	6574 ms (0.11 min.)	434870 ms (7.25 min.)

As shown in Table 3.2 selecting pre-computed kernel type parameter improves execution time of the program, considerably. Pre-computed kernel form of the program runs 67 times faster and the improvement is 7.25 minutes according to the program not including the option.

3.2 RESULTS ON SYNTHETIC DATA

In this section, the performance of PIMR is compared to classical single-view regression (CSR). Below the set of instances are obtained by taking the average of 30 loops. This process is incurred to obtain more robust and realistic results.

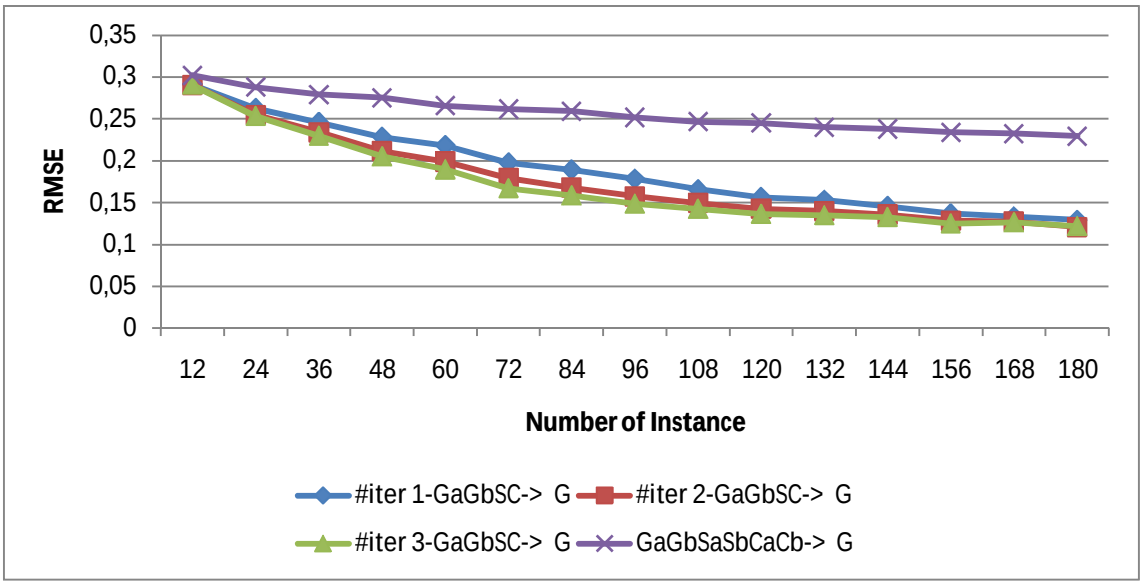


Figure 3.2 : RMSE of G_{view} on synthetic data

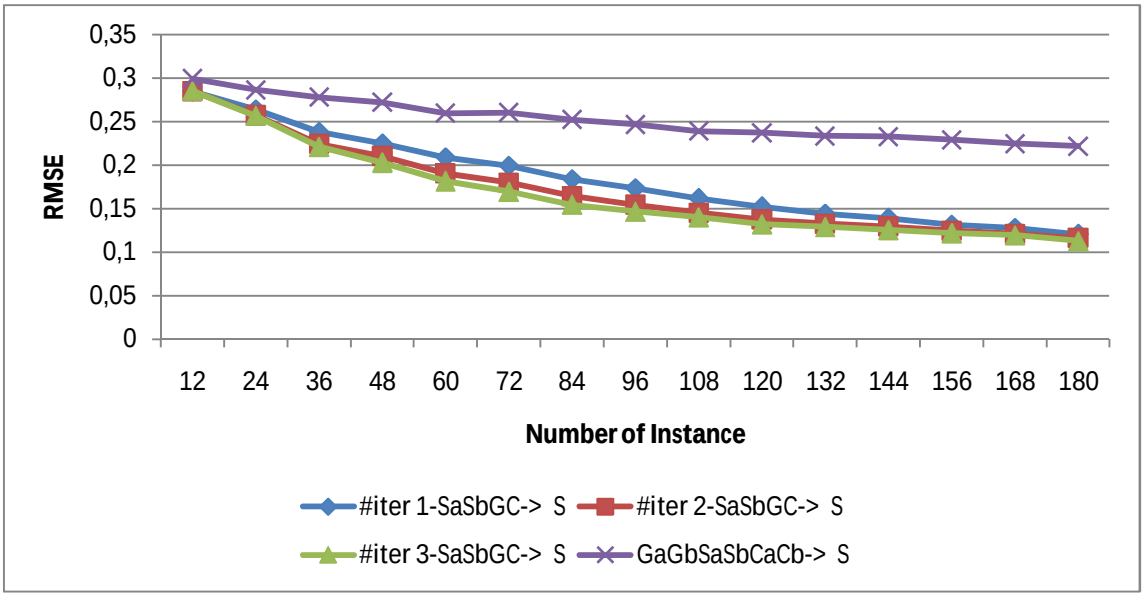


Figure 3.3 : RMSE of S_{view} on synthetic data

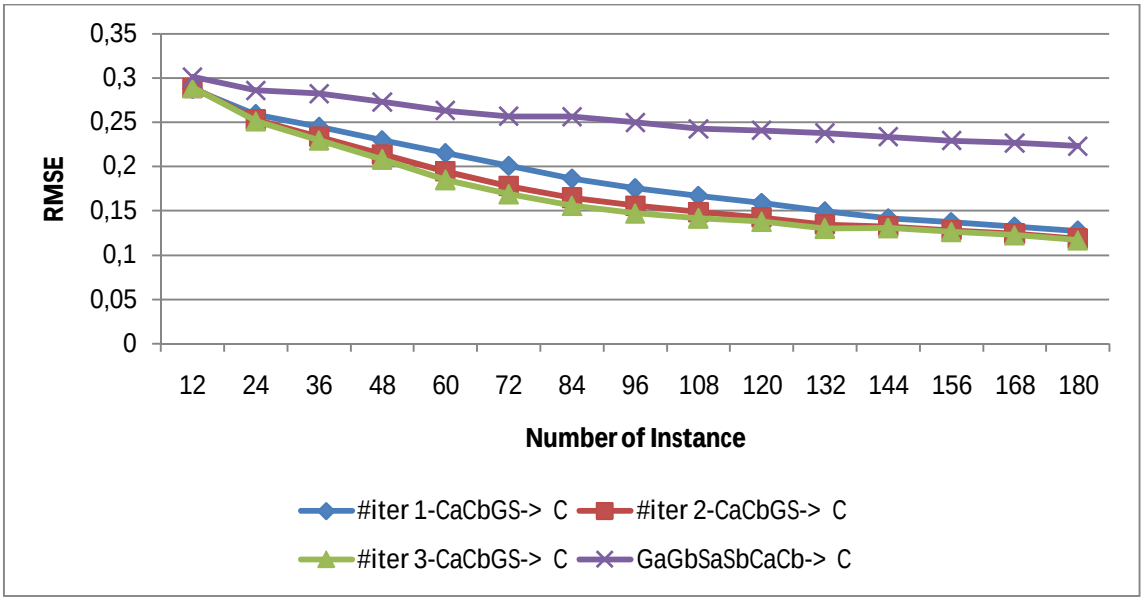


Figure 3.4 : RMSE of C_{view} on synthetic data

Figure 3.2, Figure 3.3 and Figure 3.4 show Root Mean Squared Error (RMSE) of the prediction of Glandular, Stromal and Cancer tissue scores. The row denotes RMSE; the column denotes the number of instance. Each instance group is obtained by taking the average of 30 loops. In other words, each of the datasets which has 12, 24… 180 instances, respectively, is generated 30 times. Glandular, Stromal and Cancer scores are obtained by using the mean of RMSE for each dataset group. The iterations are the units

of PIMR. Here, when the results of PIMR method is compared to that of CSR method, it can be seen that as the number of instance increases, the error rate decreases faster in PIMR method as expected. Among the iterations of PIMR, iteration 2 has the least error. As a result; PIMR has less error in its output compared to the CSR method. Iteration 1 has the same error despite the use of less data compared with the other iterations.

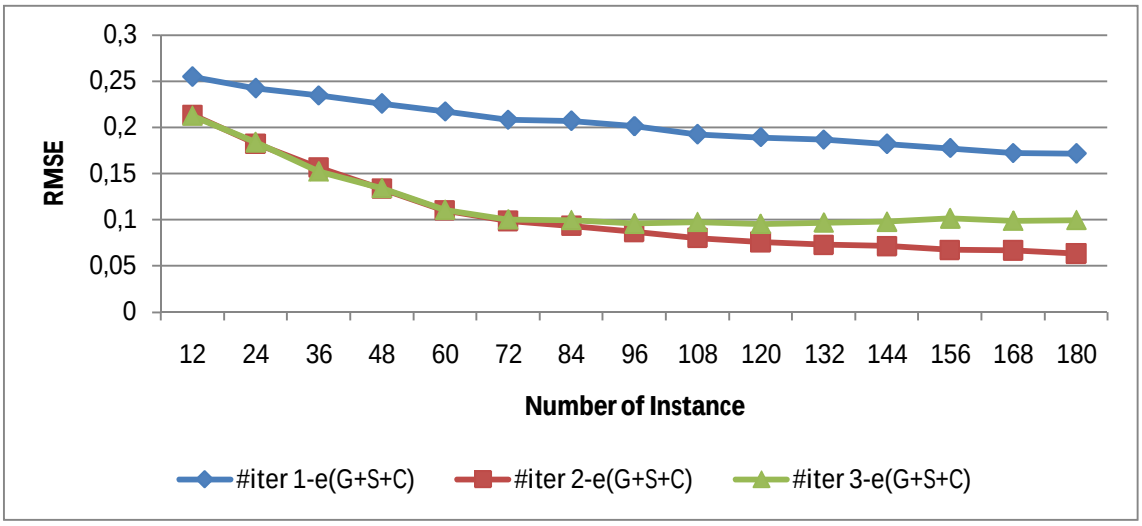


Figure 3.5 : RMSE of views on synthetic data

In Figure 3.5 the total RMSE results of the Glandular, Stromal and Cancer views are provided. For the iterations of the PIMR, total error rate is decreasing as the number of instances increases as expected. For the PIMR iteration1, the error rate is worse than the other iterations. From this point PIMR iteration 2 becomes performing better as the number of instances increases.

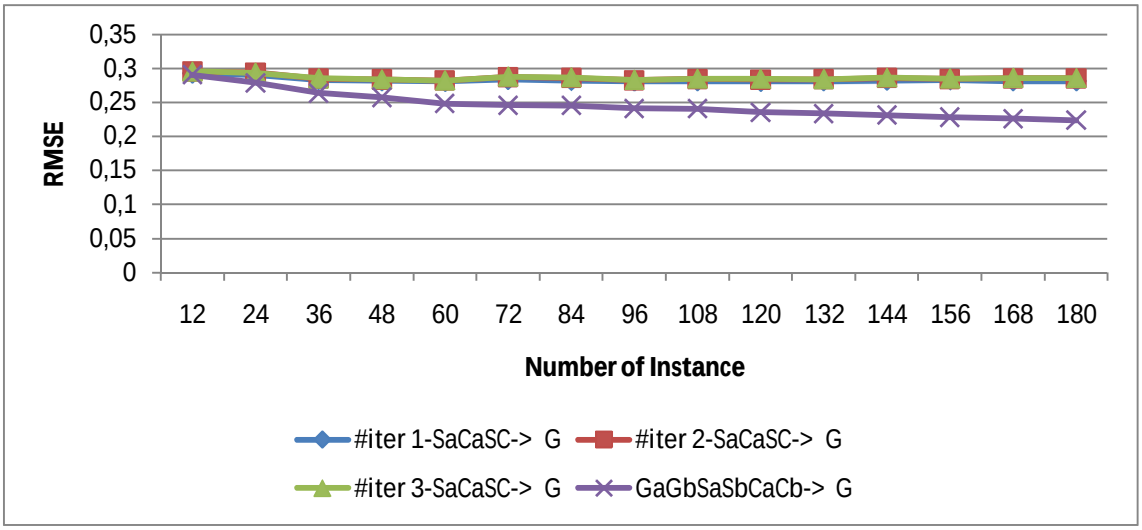


Figure 3.6 : RMSE of G_{view} with a different pattern

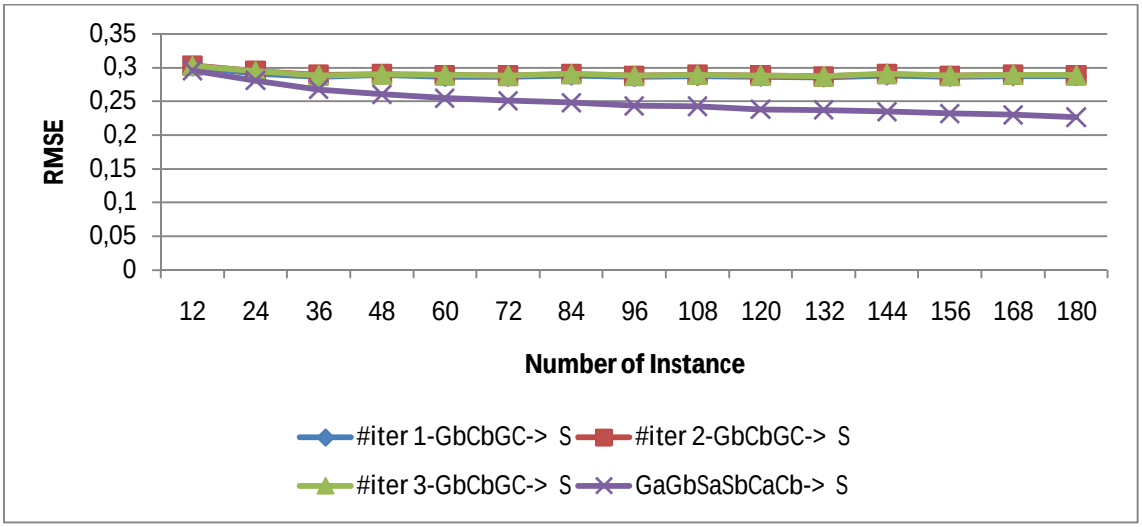


Figure 3.7 : RMSE of S_{view} with a different pattern

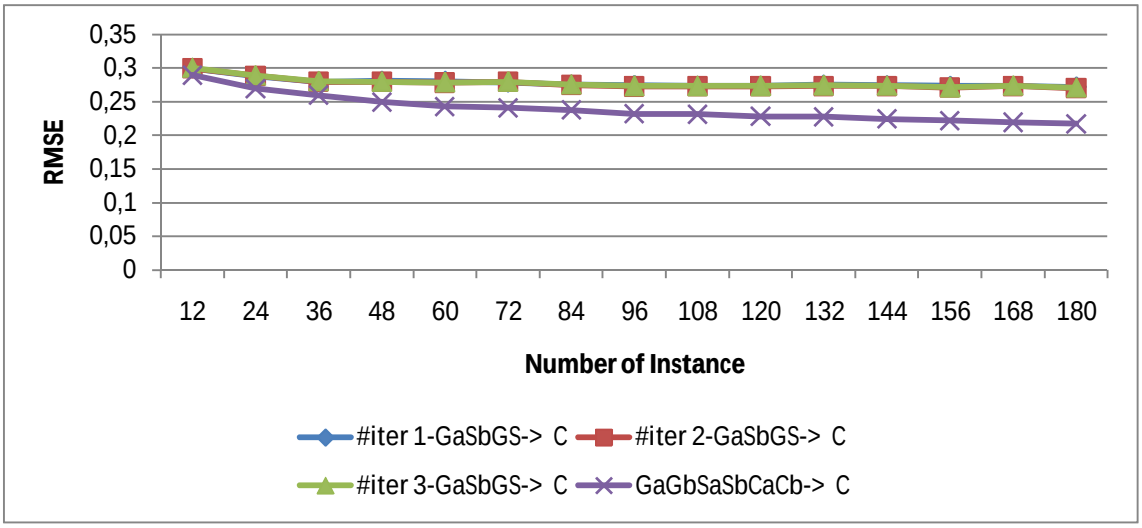


Figure 3.8 : RMSE of C_{view} with a different pattern

In Figure 3.6, Figure 3.7 and Figure 3.8 G_{view}, S_{view} and C_{view} are available with a different pattern. In other words, it is embed metabolites in views, randomly. As a result, RMSE rates of each view progress in a stable way as the number of instances increases as expected.

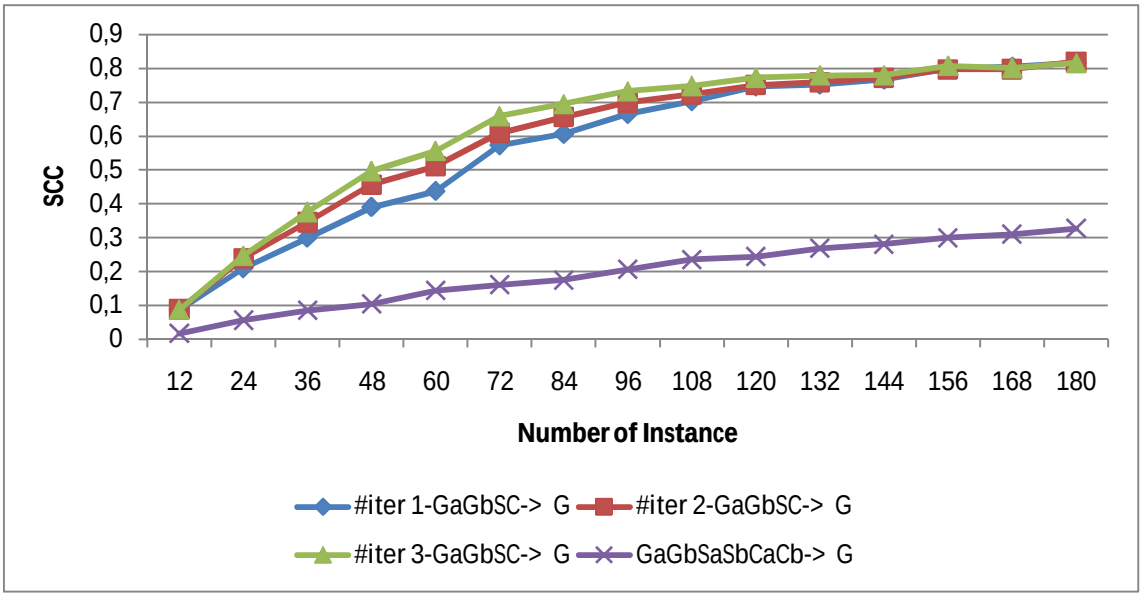


Figure 3.9 : SCC of G_{view} on synthetic data

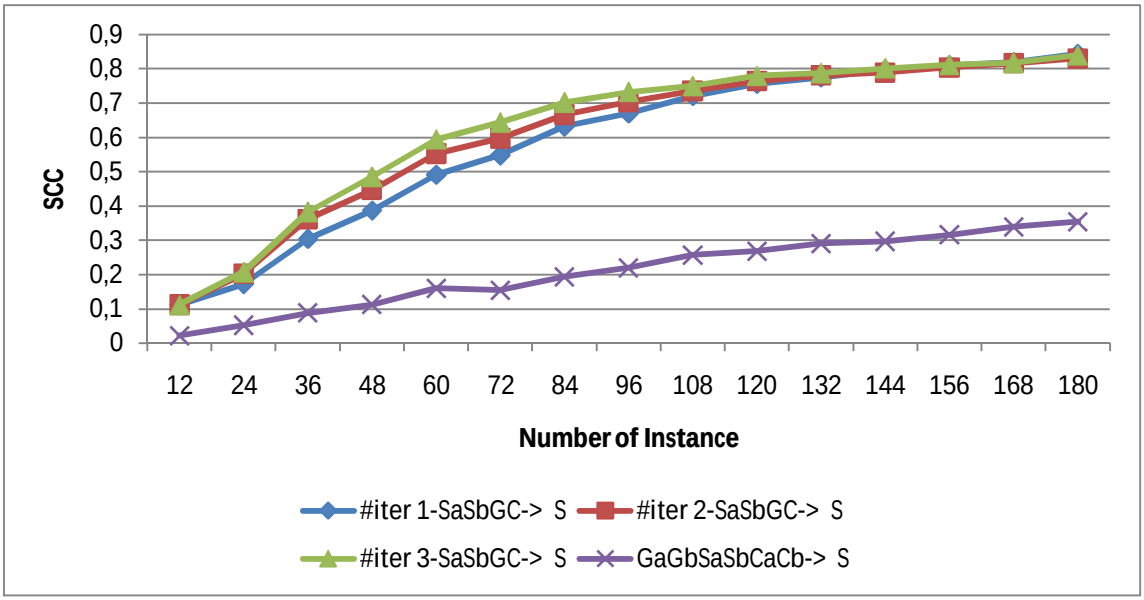


Figure 3.10 : SCC of S_{view} on synthetic data

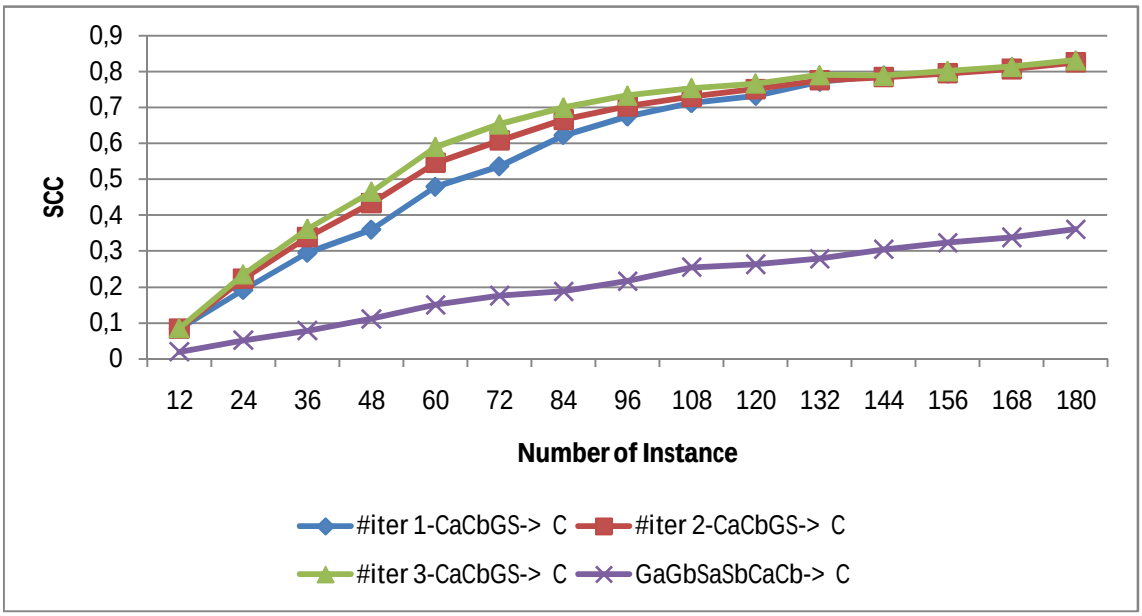


Figure 3.11 : SCC of C_{view} on synthetic data

In Figure 3.9, Figure 3.10 and Figure 3.11 rows indicate the values of squared correlation coefficient (SCC) and columns indicate number of instance. Each instance group is generated 30 times and SCC average is taken from the numbered set of data. It can be seen from these figures that PIMR always provides better correlations and as the number of instances increases correlation performances of the PIMR iterations increase

faster than that of CSR method as expected. Iteration 1 has the same correlation performance compared with the other iterations of PIMR.

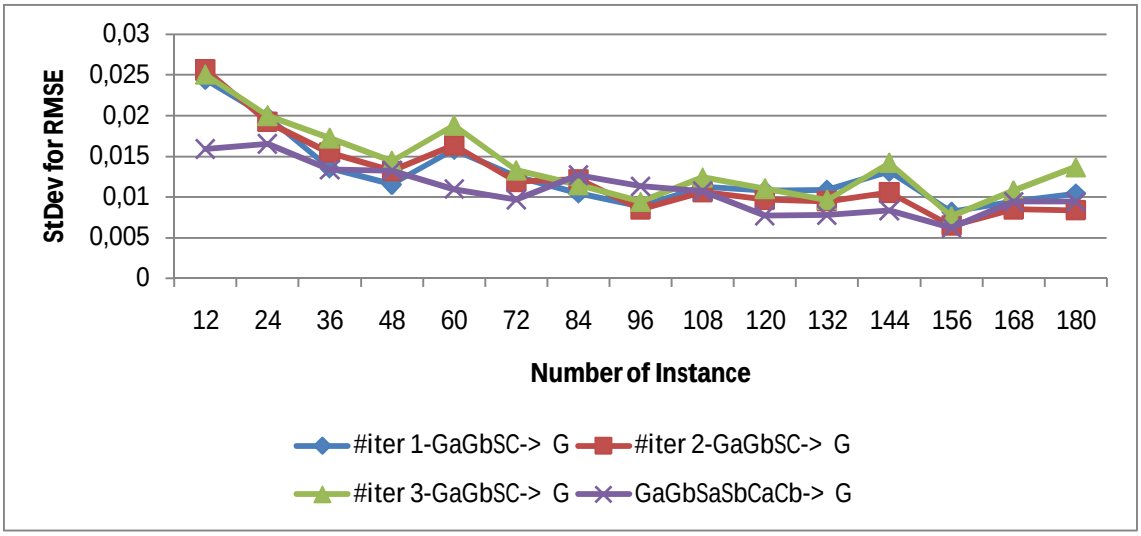


Figure 3.12 : Standard deviation for RMSE of G_{view} on synthetic data

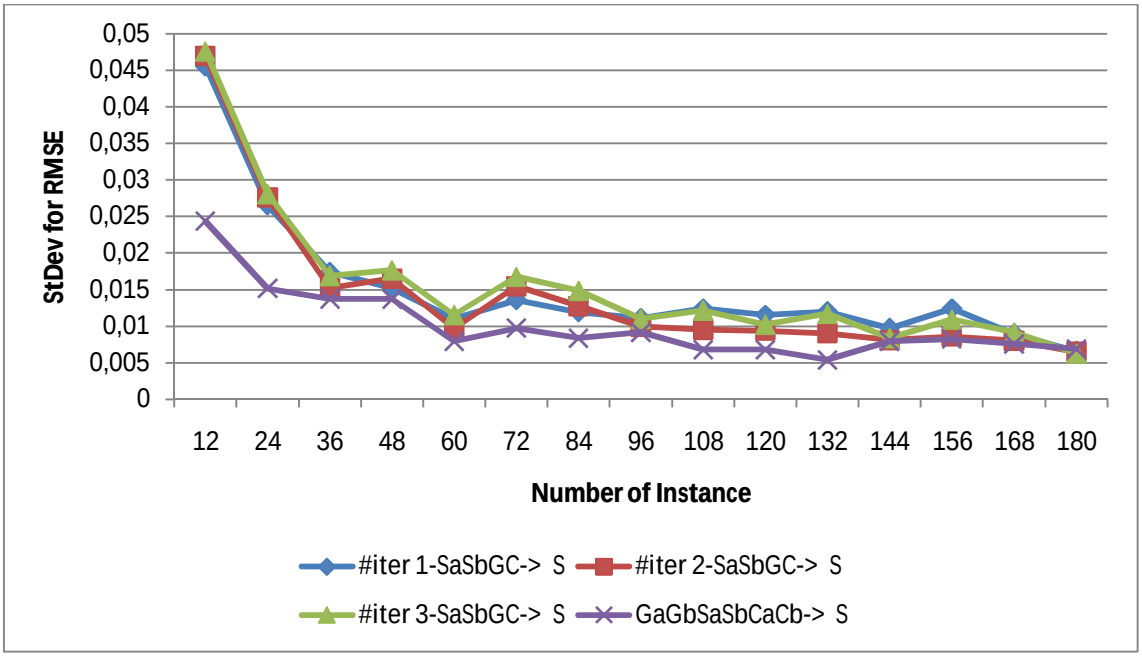


Figure 3.13 : Standard deviation for RMSE of S_{view} on synthetic data

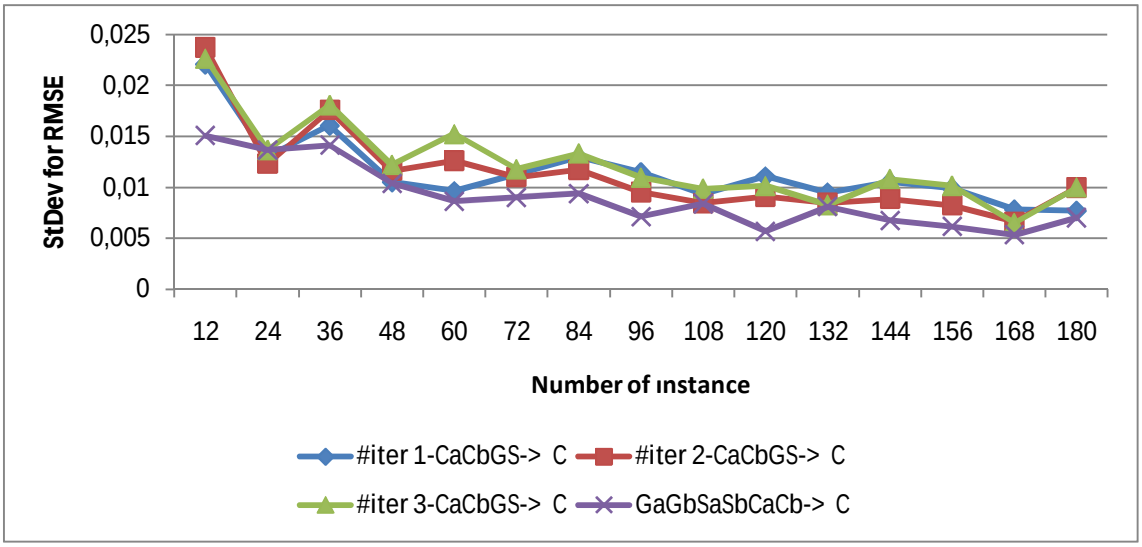


Figure 3.14 : Standard deviation for RMSE of C_{view} on synthetic data

In Figure 3.12, Figure 3.13 and Figure 3.14, rows indicate the standard deviation of root mean squared error and columns indicate the number of instance. Here, each of the used datasets is calculated by taking the average of 30 loops of the specific sets of data groups as previously mentioned. Standard deviation shows how much dispersion there is from its mean. In the charts, standard deviation decreases as the number of instances increases as expected. The decreasing of standard deviation shows the results are closer to the average. PIMR for G_{view} and S_{view} shows the same performance with CSR. But C_{view} of CSR shows better performance. Moreover, PIMR is counted to have better improvement because of its generation with fewer features.

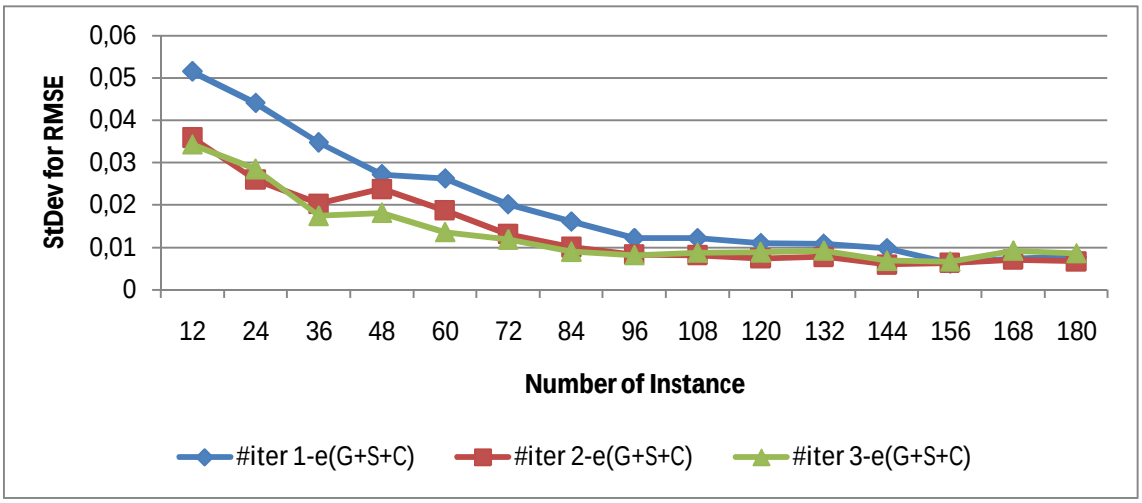


Figure 3.15 : Standard deviation for RMSE of views on synthetic data

Figure 3.15 shows standard deviation of RMSE of total views. Standard deviation of views for all iterations is balanced and performs better as the number of instances increases. After the number of total instances is 156, all iterations have the same and stable standard deviation.

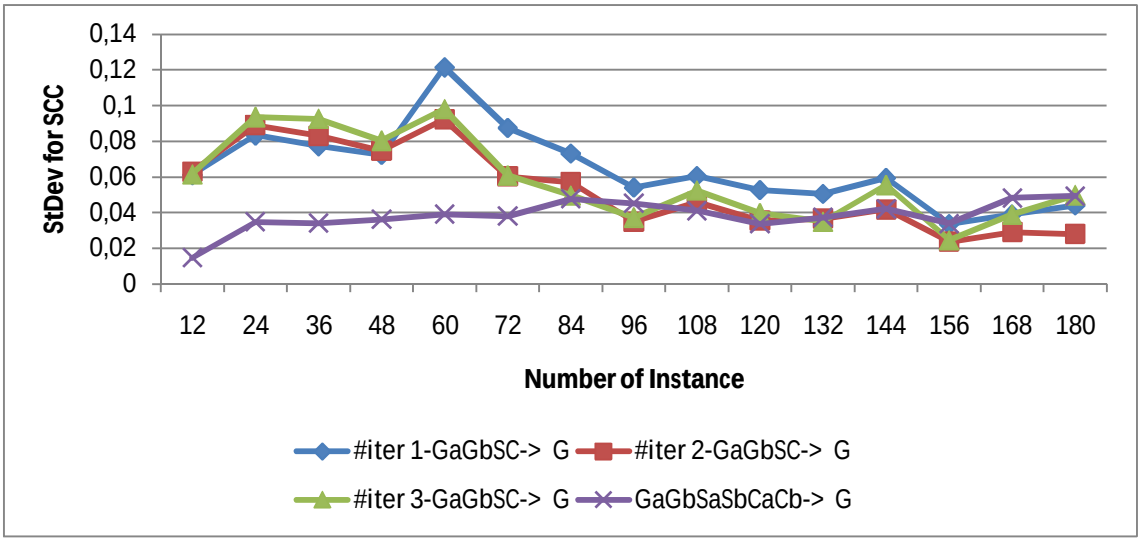


Figure 3.16 : Standard deviation for SCC of G_{view} on synthetic data

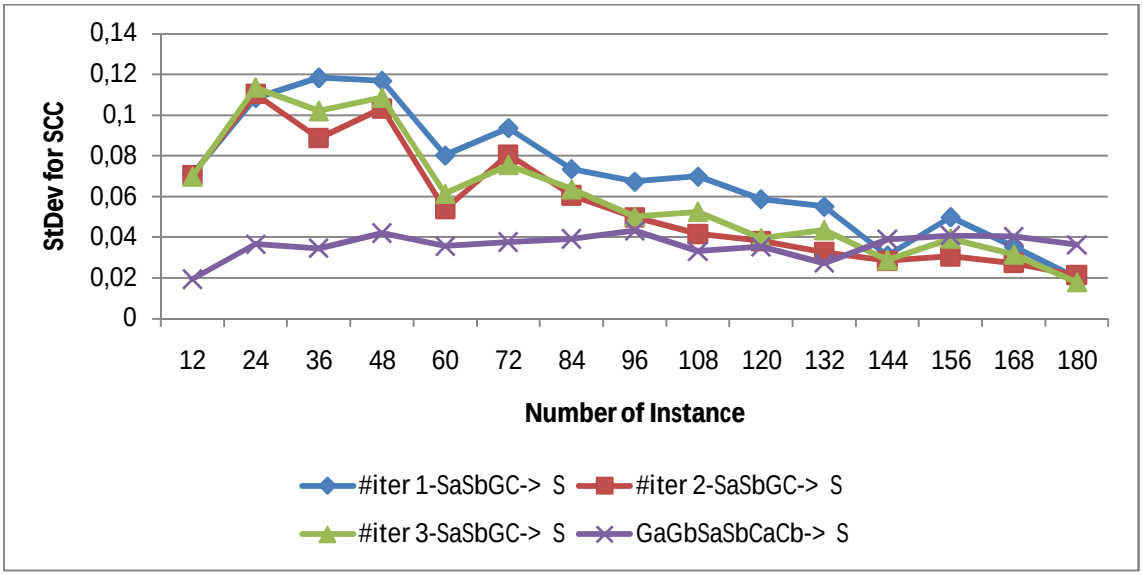


Figure 3.17 : Standard deviation for SCC of S_{view} on synthetic data

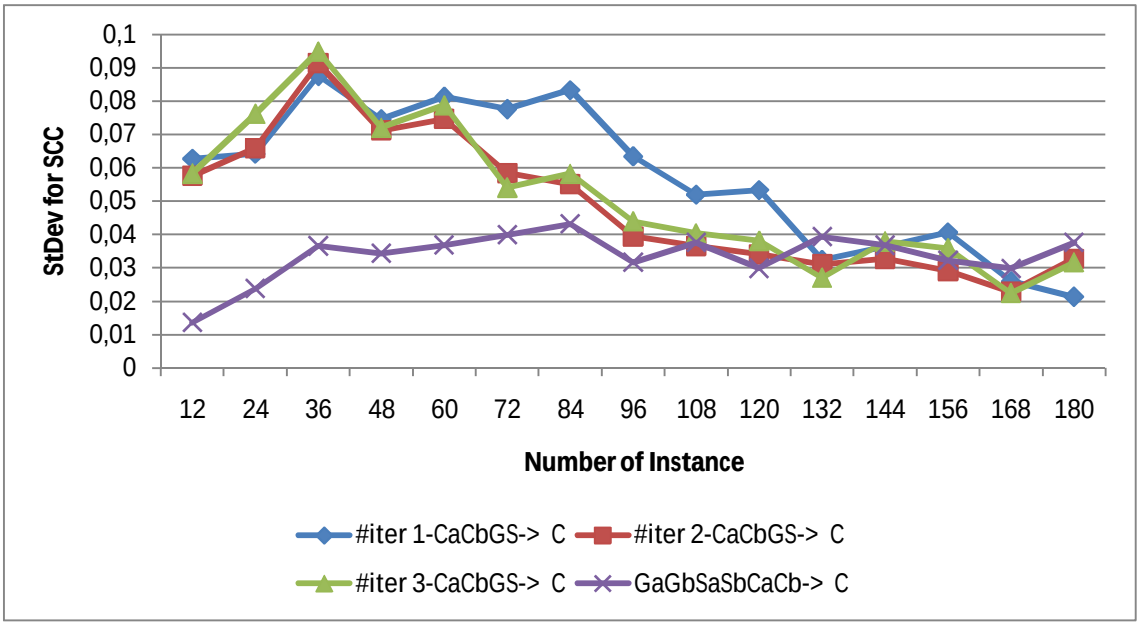


Figure 3.18 : Standard deviation for SCC of C_{view} on synthetic data

In Figure 3.16, Figure 3.17 and Figure 3.18 the charts show the standard deviation for squared correlation coefficient. Standard deviation of PIMR iterations is higher than CSR as the small number of instance. However, the performance success of standard deviation for SCC is the same as the number of instances increases. Iteration 1 is counted to have better improvement despite it has higher deviation compared with the others.

3.3 RESULTS ON PROSTATE CANCER DATASET

Prostate cancer dataset has 19 metabolites hereinbefore. The metabolites are distributed according to the partition table. In the partition table each row represents a tissue. The tissue G's members are selected 7, 14, 15, and 18 columns, the tissue S's members are selected 2, 5, 8, 11, 12 and 13 columns, the last tissue C's members are selected 0, 1, 3, 4, 6, 9, 10, 16 and 17 columns of prostate cancer dataset. The selection process is random; therefore, the different layouts may give the same result.

The training dataset is randomly selected 10 instances at a time from the whole dataset in the first turn of the system and this process is repeated 30 times. The actual results are obtained after the repeats. In other words, bootstrap resampling method is applied to

prostate cancer dataset for obtaining more efficient results. In the second turn the training dataset is randomly selected 20 instances and this process is repeated until the number of instances is 130, i.e. 13 times and each time the process is repeated 30 times as mentioned above. The following results are obtained.

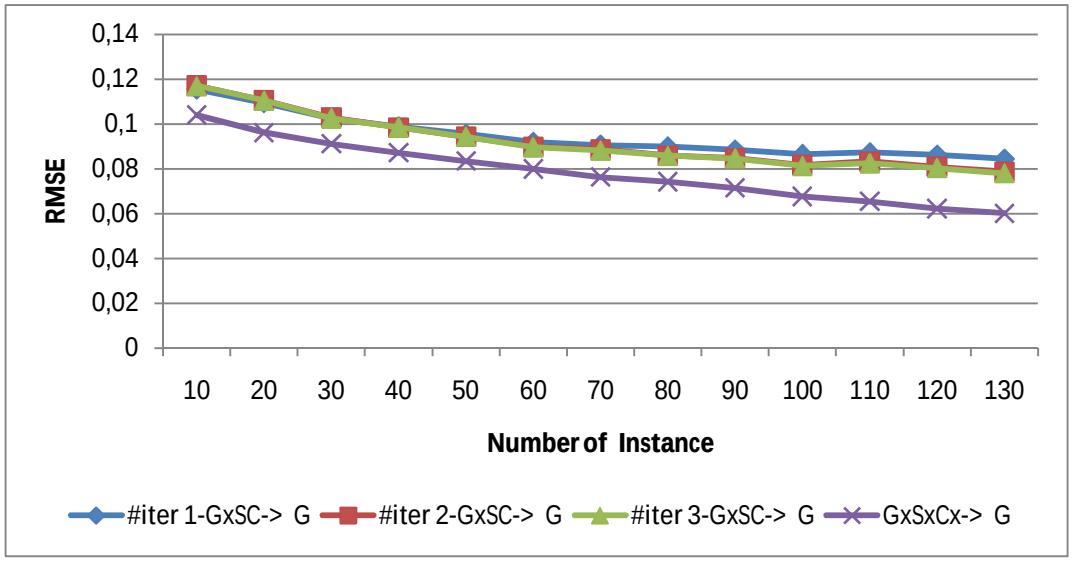


Figure 3.19 : RMSE of G_{view} on prostate cancer data

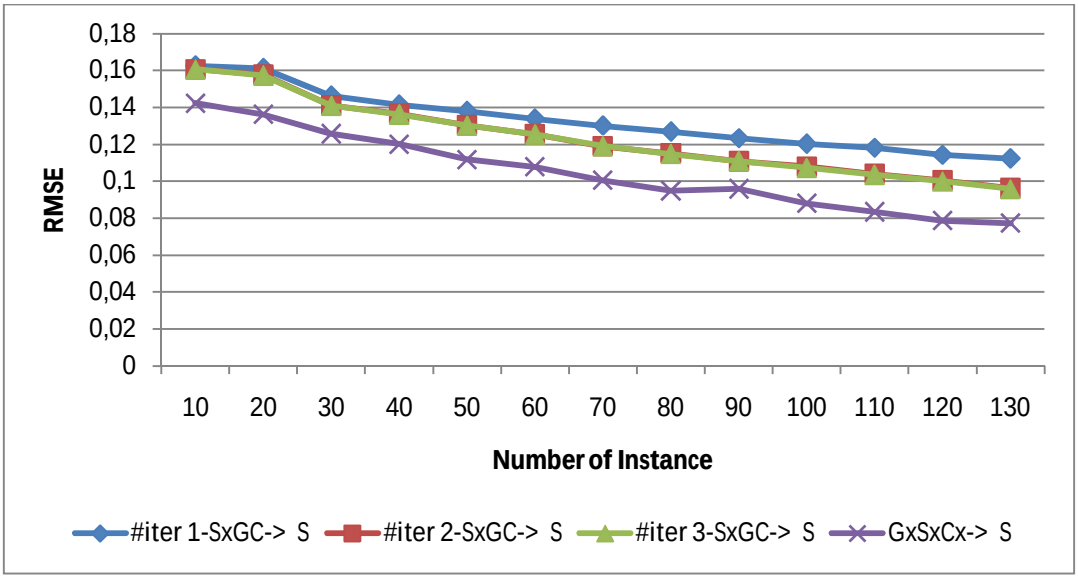


Figure 3.20 : RMSE of S_{view} on prostate cancer data

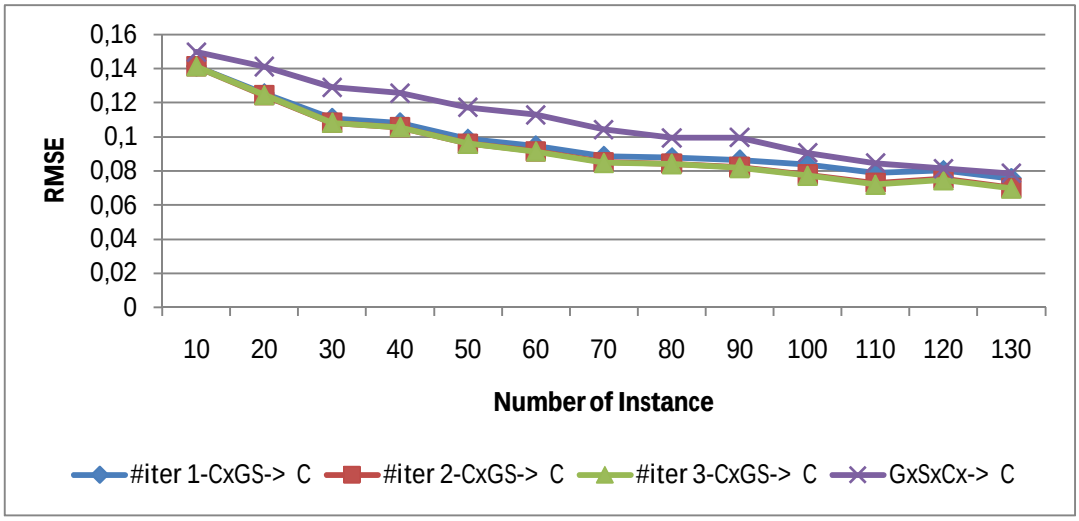


Figure 3.21 : RMSE of C_{view} on prostate cancer data

In Figure 3.19, Figure 3.20 and Figure 3.21 above the RMSE characteristics of PIMR and CSR methods are provided for each Glandular, Stromal and Cancer views. According to these results, for the views Glandular and Stromal CSR always perform a better error performance compared to the PIMR method. However, for C_{view}, PIMR has a better error characteristic if the number of instances is lower than about 110. Actually, it can be considered that the iterations of PIMR show better performance compared to CSR due to the fact that iterations are trained with less features.

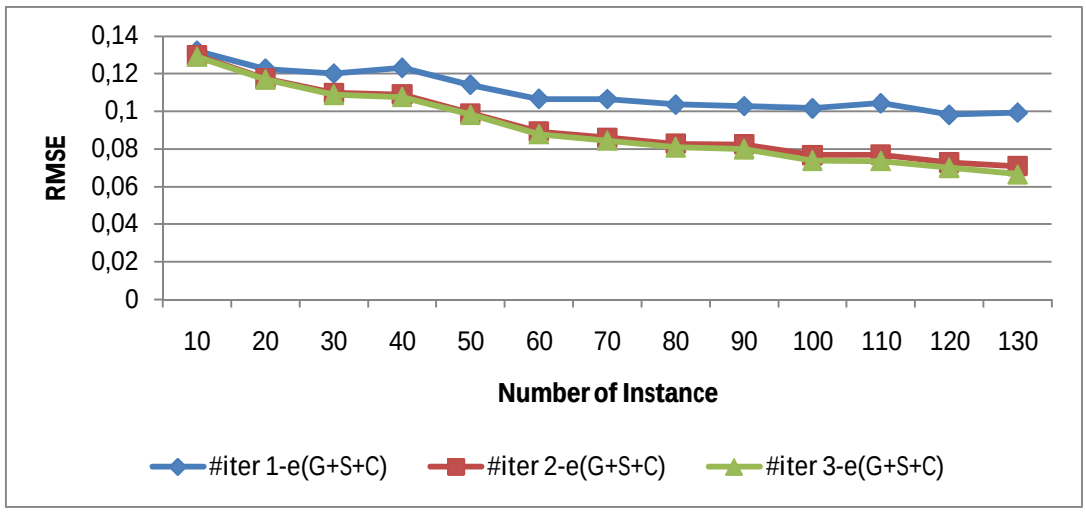


Figure 3.22 : RMSE of views on prostate cancer data

In Figure 3.22 the total RMSE characteristics of both methods are provided. In PIMR as the number of instance increases the characteristic decreases however it is for the small number of instances. For the PIMR iteration1, the error rate is worse than the other iterations. From this point PIMR iteration 2 becomes performing better as the number of instances increases.

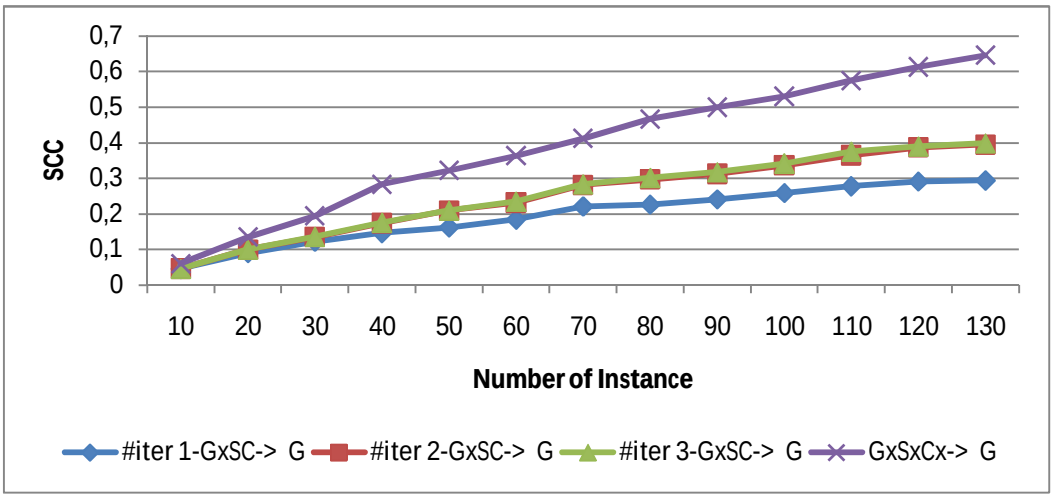


Figure 3.23 : SCC of G_{view} on prostate cancer data

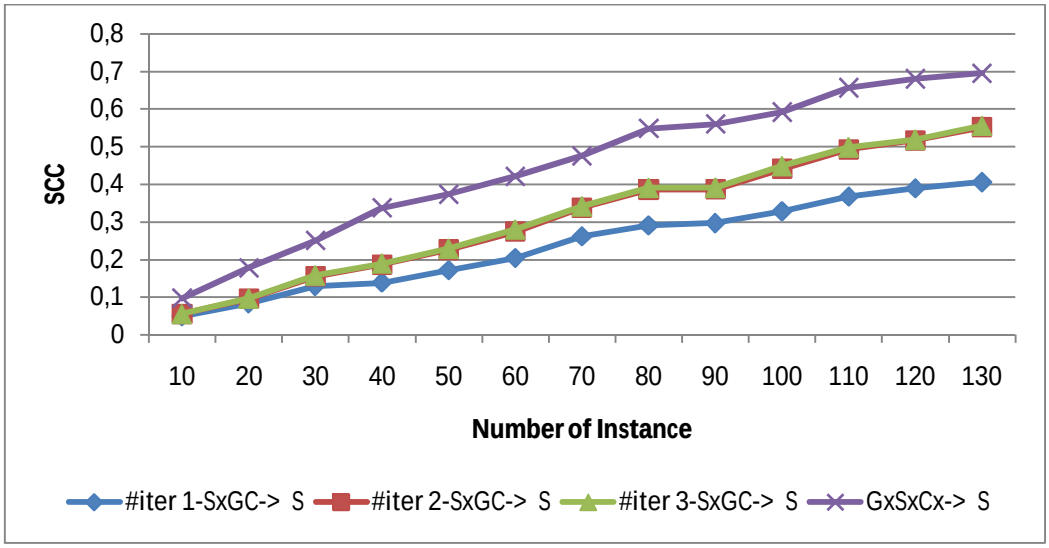


Figure 3.24 : SCC of S_{view} on prostate cancer data

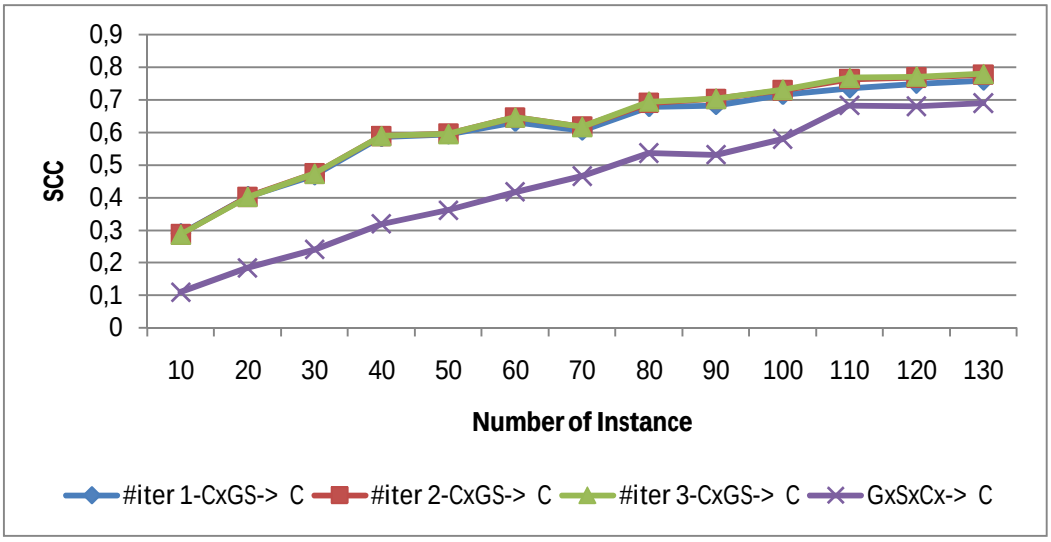


Figure 3.25 : SCC of C_{view} on prostate cancer data

In Figure 3.19, 3.20, 3.21, correlation performances of the PIMR and CSR methods are provided for each view. According to these results, the correlation of C_{view} is always higher than that of CSR. For the views G and S, correlation achieved by the PIMR is lower than CSR. Iteration 1 is lower for G_{view} and S_{view} as expected.

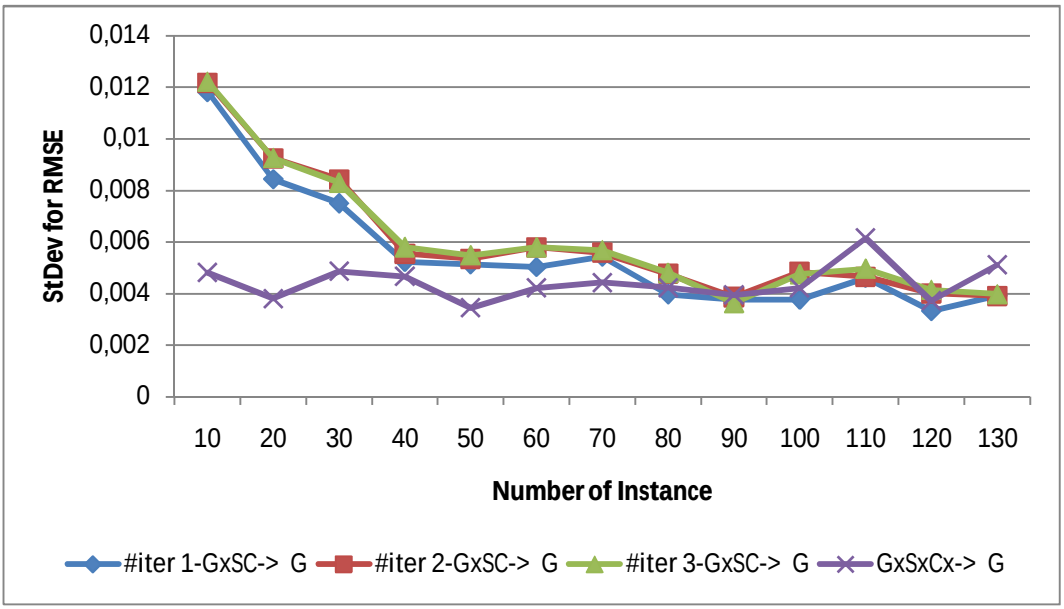


Figure 3.26 : Standard deviation for RMSE of G_{view} on prostate cancer data

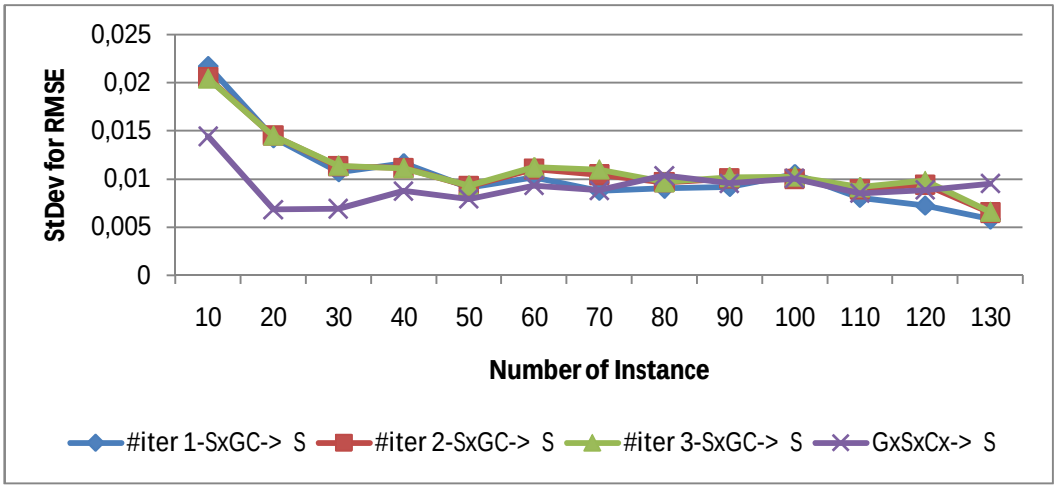


Figure 3.27 : Standard deviation for RMSE of S_{view} on prostate cancer data

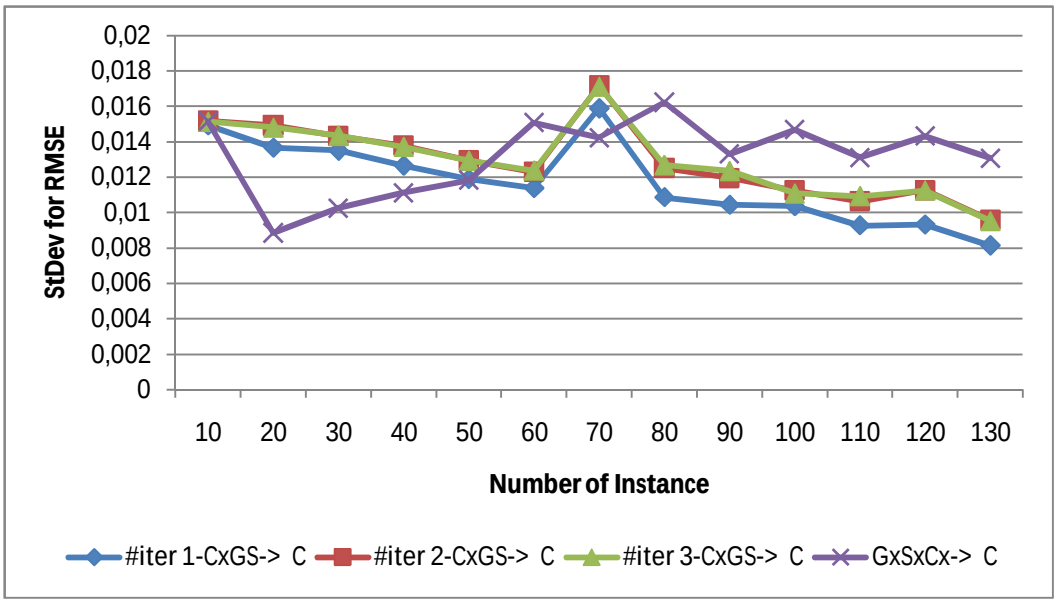


Figure 3.28 : Standard deviation for RMSE of C_{view} on prostate cancer data

In Figure 3.26, Figure 3.27 Standard deviation of G_{view}, S_{view} shows a stable characteristic as the number of instances increases. The best deviation belongs to iteration 1 of C_{view}. In the overview, the iterations of PIMR show better performance than CSR.

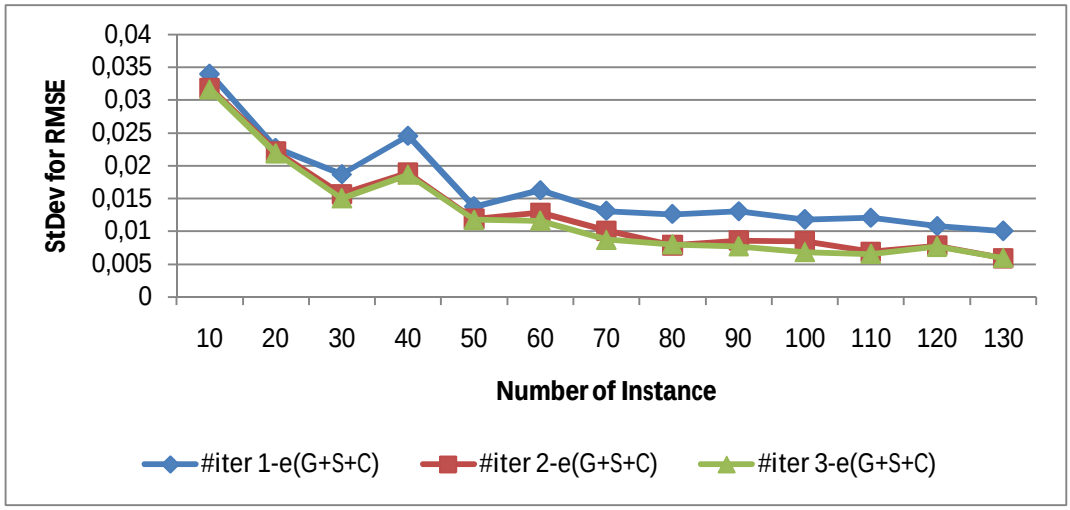


Figure 3.29 : Standard deviation for RMSE of views on prostate cancer data

Figure 3.29 shows that that standard deviation of error rate is decreasing as the number of instances increases.

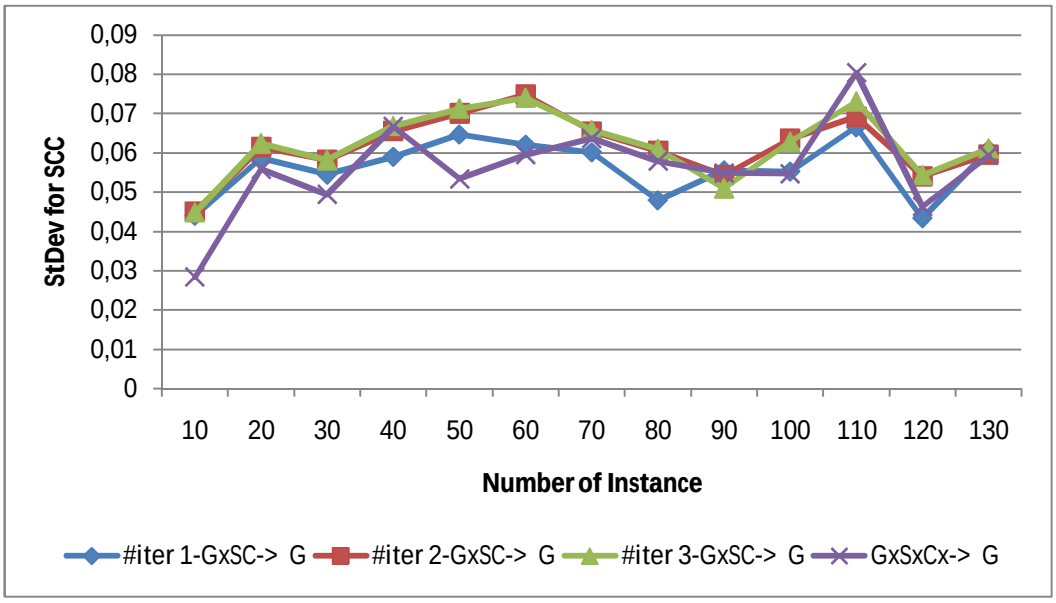


Figure 3.30 : Standard deviation for SCC of G_{view} on prostate cancer data

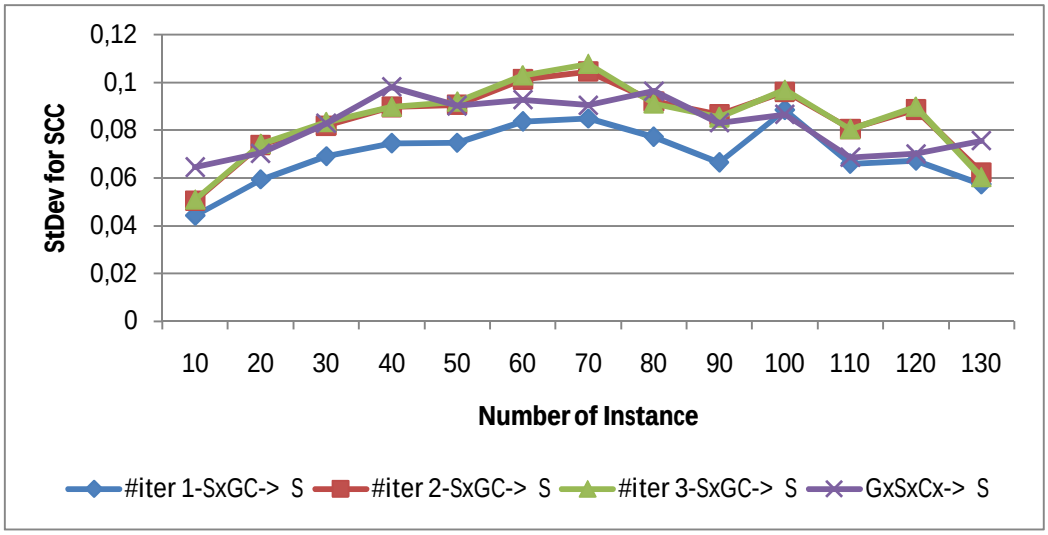


Figure 3.31 : Standard deviation for SCC of S_{view} on prostate cancer data

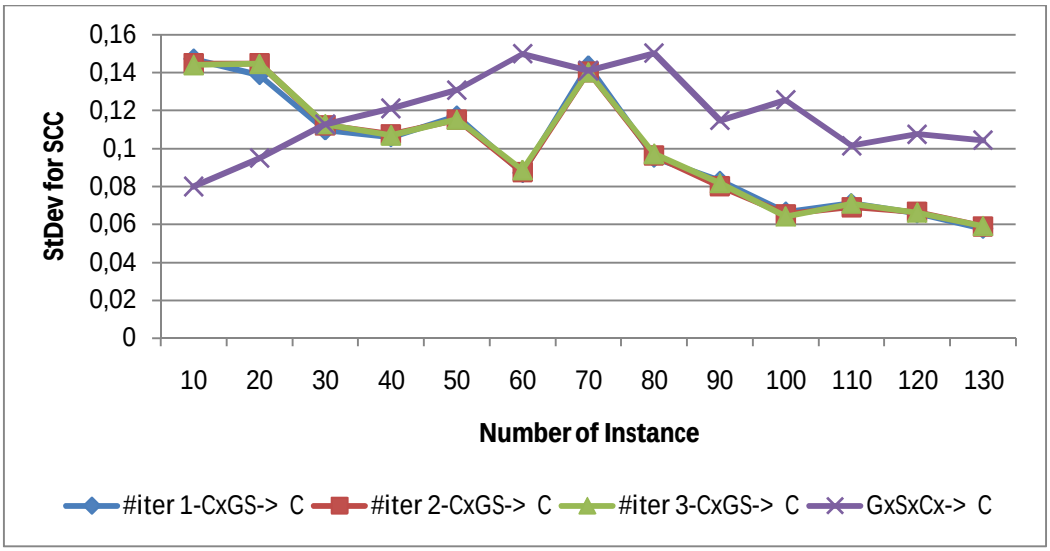


Figure 3.32 : Standard deviation for SCC of C_{view} on prostate cancer data

In Figure 3.30, Figure 3.31 and Figure 3.32, the distribution of standard deviation for correlation is unbalanced. The charts show the sudden fluctuation for many set of instances. It is observed that the iteration 1 of S_{view} shows lower deviation performance compared to CSR and the other iterations. The correlation deviation for G_{view} and S_{view} is higher as the number of instances increases. The iterations of PIMR for C_{view} have lower deviation as the number of instances increases. The standard deviation of CSR for C_{view} is generally high but balanced after 110. PIMR shows a better performance for C_{view} as the number of instances increases.

4. DISCUSSION AND CONCLUSION

In this thesis, an SVM-based multi-view multiregression approach was tried on synthetic and real-world datasets. Given a dataset divided into views, in this study, it was our aim to analyze the effect of the interactions between views, each one of which is primarily responsible for predicting a separate target variable. Firstly, a synthetic toy dataset was generated for analyzing this effect as a simulation of the real metabolomics prostate cancer dataset. The accuracy, the root mean squared error and squared correlation coefficient rates, of PIMR on synthetic and prostate cancer dataset were better than the classical single-view regression approach. Such good results were not seen when using different (shuffled) metabolite views instead of the true ones. As a future direction, the metabolites of prostate cancer dataset (or within the views) may be reduced using feature selection methods such as mRMR (Peng *et al.* 2005, Sakar and Kursun 2010).

REFERENCES

Books

Alpaydın E., 2010. 2nd Ed. *Introduction to machine learning*. Cambridge, MA: MIT Press.

Epstein, R., Clipson, A., Hesterberg, T. C., Monaghan, S., & Moore, D. S., 2003. *Bootstrap methods and permutation tests*. New York: W. H. Freeman and Company.
Chapter available at: http://bcs.whfreeman.com/pbs/cat_140/chap18.pdf

Han, J., & Kamber M., 2006. *Data mining: concept and techniques*. 2nd Ed. San Francisco: Morgan Kaufmann Publishers.

Schölkopf, B. & Smola, A.J., 2002. *Learning with Kernels*. Cambridge, MA: MIT Press.

Vapnik, V., 1995. *The nature of statistical learning theory*, New York: Springer.

Periodical Publications

Bellman, R., 1961. Adaptive control processes: A Guided Tour. Princeton: Princeton University Press.

Boser, B. E., Guyon, I. M., & Vapnik, V. N., 1992. A training algorithm for optimal margin classifiers. *In Proceedings of the Fifth Annual Workshop on Computational Learning theory (COLT '92). Pittsburgh, Pennsylvania, United States July 27 - 29. ACM*, New York, NY, pp. 144-152.

Campbell, W. M., Campbell, J. P., Reynolds, D. A., Singer, E. & Torres-Carrasquillo, P. A., 2006. Support vector machines for speaker and language recognition. *Computer Speech & Language*, **20** (2-3), pp. 210-229.

Christoudias C. M., Urtasun R., & Darrell T., 2008. Multi-View Learning in the Presence of View Disagreement, *In Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Cortes C., & Vapnik V., 1995. "Support-vector networks," Machine Learning, **20** (3), pp. 273-297.

Culp, M., Michailidis, G., & Johnson, K., 2009. On multi-view learning with additive models. *The Annals of Applied Statistics*. **3** (1), pp. 292-318.

Ding, S., Hu, T., Shen, Y., Lin, C., & Huang, Y., 2009. Detection of Lung Cancer with Breath Biomarkers Based on SVM Regression. *2009 Fifth International Conference on Natural Computation icnc*, **2**, pp.131-138.

Efron, B., 1979. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics* **7** (1), pp. 1–26.

Evgeniou, T., Micchelli, C. A. & Pontil, M., 2005. Learning Multiple Tasks with Kernel Methods. *Journal of Machine Learning Research*, **6**(Apr), pp. 615-637.

Favorov, O.V. & Kursun, O., 2011. Neocortical layer 4 as a pluripotent function linearizer. *Journal of Neurophysiology*, **105**(3), pp. 1342-1360.

Furey T.S., N. Christianini, N. Duffy, D.W. Bednarski, M. Schummer, & D.Haussler, 2000. Support vector machine classification and validation of cancer tissue samples using microarray expression data. Bioinformatics, **16** (10), pp.906–914.

Gonen, M. & Alpaydin, E., 2010. Cost-conscious multiple kernel learning. *Pattern Recognition Letters*, **31**(9), pp. 959-965.

Griffin, JL, & Kauppinen, RA, 2007. A metabolomics perspective of human brain tumors. *FEBS J 2007*; **274** (11), pp.32-9.

Hardoon, D., Szedmak, S., & Shawe-Taylor, J., 2004. Canonical correlation analysis: an overview with application to learning methods. *Neural Computation*, **16**, pp. 2639-2664.

Heisele, B. , Ho, P., & Poggio, T., 2001. Face Recognition with Support Vector Machines: Global versus Component-based Approach, *Proc. of the Eighth IEEE International Conference on Computer Vision, ICCV 2001*, **2**, Vancouver, Canada, pp. 688-694.

Joachims, T., 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *European Conference on Machine Learning (ECML)*.

Kohavi, R., 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* **2** (12), pp. 1137–1143. (Morgan Kaufmann, San Mateo) Available at: http://robotics.stanford.edu/users/ronnyk/

Kursun, O., Alpaydin, E. & Favorov, O.V., 2011. Canonical Correlation Analysis Using Within-class Coupling. *Pattern Recognition Letters,* **32**(2), pp. 134-144. DOI: 10.1016/j.patrec.2010.09.025

Kursun, O. & Favorov, O.V., 2010. Feature Selection and Extraction Using an Unsupervised Biologically-Suggested Approximation to Gebelein's Maximal Correlation. *International Journal of Pattern Recognition and Artificial Intelligence,* **24**(3), pp. 337-358.

Liu, H. X., Zhang, R. S., Luan, F., Yao, X. J., Liu, M. C., Hu, Z. D., & Fan, B. T., 2003. Diagnosing breast cancer based on support vector machines. Journal of Chemical Information and Computer Sciences. **43** (3). pp. 900-907.

Peng, H., Long, F. & Ding, C., 2005. Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**(8), pp. 1226-1238.

Sakar, C.O. & Kursun, O., 2010. Telediagnosis of Parkinson's Disease Using Measurements of Dysphonia. *Journal of Medical Systems*, **34**(4), pp. 591-599. DOI: 10.1007/s10916-009-9272-y

Sakar, C.O., Kursun, O., Seker, H., Gurgen, F., Aydin N., & Favorov, O.V., 2009. Parallel Interacting Multiview Learning: An Application to Prediction of Protein Sub-Nuclear Location. *9th International Conference on Information Technology and Applications in Biomedicine (IEEE ITAB 2009)*, Larnaca, Cyprus. pp.1-4.

Venkatesh, J., & Sureshkumar, C., 2009. Handwritten Tamil Character Recognition Using SVM. *(IJCNS) International Journal of Computer and Network Security.* **1** (3), *pp. 29-33.*

Wang, Z., & Chen, S., 2009. Multi-view kernel machine on single-view data. Elsevier Science Publisher B. V. Amsterdam, The Netherlands, Neurocomputing, **72** (10-12), pp. 2444-2449.

Other Publications

Chang, C.-C., & Lin, C.-J., 2001. LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm [cited August, 2008].

Demirci, D. A., 2007. Destek vektör makineleri ile karakter tanıma. *Thesis for the M.A. Degree.* İstanbul: Yıldız Teknik University NatSci.

Fletcher, T., 2009. Support Vector Machines Explained. www.cs.ucl.ac.uk/sta_/T.Fletcher/ [cited July, 2010].

Golub, T. R., Slonim, D. K., Tamayo, Huard, P. C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., & Lander, E. S. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531, 1999.

Kecman, V., 2004. Support Vector Machines Basics. *School of Engineering Report 616***.** The University of Auckland, Auckland, NZ, pp. 58.

Keshari, K. R., Kursun, O., Iman, R., Favorov, O., Tabatabai, Z.L., Simko, J., Shinohara, K., Macdonald, J., & Kurhanewickz, J., 2009. Mutual Information and HRMAS NMR Spectroscopy of biopsies to study Prostate Cancer. *Poster*. Department of Radiology, Pathology, Urology. San Francisco: University of California.

Kim, D-H., Mayer, D., Hunjan, S., Xing, L., & Spielman, D., 2004. Citrate Magnetic Resonance Spectroscopy at 3 T**.** Proc. Intl. Soc. Mag. Reson. Med. 11.

Özkara, H. A., 2008. Gelişen klinik biyokimya: Metabolomik. *Hacettepe Tıp Dergisi* 2008; 39:4-8.

Turkish Statistical Institute, 2010. *Turkey's Statistical Yearbook 2009*. Ankara: Turkish Statistical Institute, Printing Division, pp. 100.
Report available at: http://www.tuik.gov.tr/yillik/yillik.pdf

APPENDIX A

Algorithm: Pseudocode of the PIMR method on prostate cancer dataset

Input: D: Database, P: Partition table, n_lp, n_l, c: user defined number,
Output: results tables of rmse, correlation, standard deviations of rmse and correlation
Method:

```
1.  PROCEDURE main()
2.      READ D:database
3.      READ P:partition table
4.      RETURN testing_data <- CALL procedure get_testing_data();
5.      COPY G, S and C members of testing_data as new members to .
        testing_data
6.      CLEAR the G, S and C of testing_data
7.      FOR is less than n_lp
8.          FOR data_id is less than c
9.              RETURN training_data <- CALL procedure .
                get_training_data();
10.             SET temp_testing_data <- CLONE testing_data
11.             COPY G, S and C members of training_data as new .
                members to training_data
12.             CLEAR the G, S and C of training_data
13.             FOR j is less than 3 //it runs 3 iterations
14.                 CALL procedure compute_gsc(P, training_data, .
                    temp_testing_data);
15.                 CHANGE G, S and C values of temp_testing_data
                    with outputs
16.                 CHANGE G, S and C values of training_data with
                    outputs
17.                 REMOVE outputs of temp_testing_data
18.                 REMOVE outputs of training_data
```

```
19.                         CALL procedure .
                            compute_error_of_gsc(temp_testing_data);
20.                 END for
21.                 CALL procedure compute_gsc(P, training_data, .
                    temp_testing_data);
22.                 CALL procedure .
                    compute_error_of_gsc(temp_testing_data );
23.             END for
24.         END for
25.         PRINT Root Mean Squared Error
26.         PRINT Squared correlation coefficient
27.         COMPUTE Standard Deviation for Root Mean Squared Error
28.         PRINT Standard Deviation for Root Mean Squared Error
29.         COMPUTE Standard Deviation for Squared correlation coefficient
30.         PRINT Standard Deviation for Squared correlation coefficient
31.     END procedure

32.     PROCEDURE get_training_data()
33.         ADD random lines of D to training_data
34.         RETURN training_data
35.     END procedure

36.     PROCEDURE get_testing_data()
37.         ADD whole data in D to testing_data
38.         RETURN testing_data
39.     END procedure

40.     PROCEDURE compute_gsc(P, training_data, temp_testing_data)
41.         FOR each pattern // G_SCGx, S_GCSx, C_GSCx,  G_GxSxCx, .
            S_GxSxCx, C_GxSxCx
42.             SET training_array <- set array for related pattern according to P
43.             CALL procedure train(training_array);
```

```
44.             SET output <- CALL procedure predict(model, training_array);
45.             COPY outputs as new members to training_data
46.             SET temp_testing_array <- set array for related pattern
47.             SET output <- CALL procedure predict(model, .
                temp_testing_array);
48.             COPY outputs as new members to temp_testing_data
49.         END for
50.     END procedure

51.     PROCEDURE compute_error_of_gsc(temp_testing_data){
52.         COMPUTE root mean squared error of G, S and C(G+S+C) members of
            testing_data
53.         COMPUTE squared correlation coefficient of G, S and C(G+S+C) .
            members of testing_data
54.     END procedure

55.     PROCEDURE train(training_array)
56.         SET SVM parameters to user defined value
57.         TRAIN training_array
58.         DETERMINE model
59.     END procedure

60.     PROCEDURE predict(testing_array)
61.         TEST testing_array
62.         RETURN output
63.     END procedure
```

CURRICULUM VITAE

Name / Last Name	**:**	Safiye YAYLAOĞLU
Place / Date of Birth	**:**	İstanbul / 1980
Marital Status	**:**	Married
Driving License	**:**	B Class
E-Mail	**:**	ysafiye@hotmail.com
Address	**:**	Atakent Mah. Mesa B4 Blok G: 4 D: 34303 Küçükçekmece/İSTANBUL
Languages	**:**	English
High School	**:**	Private Safiye Sultan Girl College - 1999
B.Sc.	**:**	Çanakkale Onsekiz Mart University, Computer and Instructional Technologies Education - 2003
M.Sc.	**:**	Bahçeşehir University - 2011
Name of Institute	**:**	The Graduate School of Natural and Applied Sciences
Name of Program	**:**	Computer Engineering Graduate Program
Work Experience	**:**	Halkalı Cumhuriyet Elementary School, Information Technologies Teacher and Supervisor – (September 2003 – February 2011) Halkalı Güneş Elementary School, Information Technologies Teacher and Supervisor – (February 2011 – Today)