Formulation of Multiple Kernel Learning Using Composite Architectures

A thesis submitted

in partial fulfillment for the award of the degree of

Doctor of Philosophy

by

Shiju S. S.

Department of Mathematics
Indian Institute of Space Science and Technology
Thiruvananthapuram, India

February 2019

Certificate

This is to certify that the thesis titled ***Formulation of Multiple Kernel Learning Using Composite Architectures*** submitted by **Shiju S. S.**, to the Indian Institute of Space Science and Technology, Thiruvananthapuram, in partial fulfillment for the award of the degree of **Doctor of Philosophy**, is a bona fide record of the original work carried out by him/her under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Sumitra S.
Associate Professor
Department of Mathematics

Dr. N. Sabu
Professor & Head
Department of Mathematics

Place: Thiruvananthapuram
Date: February 2019

Declaration

I declare that this thesis titled ***Formulation of Multiple Kernel Learning Using Composite Architectures*** submitted in partial fulfillment for the award of the degree of **Doctor of Philosophy** is a record of original work carried out by me under the supervision of **Dr. Sumitra S.**, and has not formed the basis for the award of any degree, diploma, associateship, fellowship, or other titles in this or any other Institution or University of higher learning. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

Place: Thiruvananthapuram
Date: February 2019

Shiju S. S.
(SC13D008)

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Dr. Sumitra S. for the continuous support of my Ph.D. study and related research, for her patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank the rest of my doctoral committee: Prof. Chandra B.(IIT Delhi), Prof. Subrahamanian Moosath K.S.(Professor, Department of Mathematics, IIST), Prof. N. Sabu.(Professor & Head, Department of Mathematics, IIST), Prof. Raju K. George (Dean R&D, IIST), Prof. Rama Krishna Sai Gorthi(IIT Tirupati) and Dr. Deepak TG (IIST), for their insightful comments and encouragement, for the hard question which incented me to widen my research from various perspectives.

I thank my fellow labmates especially Mr. Asif Salim in for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last four years. I thank my friends in the Indian Institute of Space Science and Technology. Also I am grateful to the departmental office staff for making my life easy in terms of paperworks.

I would also like to thank my family: my parents, my wife and my daughter for supporting me spiritually throughout writing this thesis and my life in general.

Last but not the least, I would like to thank Almighty God who has given me good health, knowledge, strength and skills needed to complete the thesis.

Shiju S. S.

Abstract

Multiple Kernel Learning (MKL) algorithms deal with learning the optimal kernel from training data along with learning the function that generates the data. Generally in MKL, the optimal kernel is defined as a combination of kernels under consideration (which are usually termed as base kernels). There are different approaches to learning the optimal combination of kernels. The main objective of the thesis is to develop models for finding the optimal combination of kernels suitable for different types of problems.

This thesis describes the formulation of optimal kernel using four different approaches. The first approach is based on the selection of kernel suitable for the features (attributes) of the data. The features having similar characteristics are clustered together and a suitable kernel is found for each cluster. The optimal kernel is defined as a linear combination of the kernels defined over the cluster subspaces. We formulated a methodology for clustering the features and applying a separate kernel over each cluster.

The second approach is based on Kumar et. al. (2012) [1] in which the problem of learning the kernel for a binary classification is designed as another binary classification problem and hence the data modeling problem involves the computation of two decision boundaries of which one is related with that of MKL and the other with that of input data. They used two different cost functions for finding the optimal function related with kernel learning and the classification task. We modified this work in such a way that in our approach, the optimal functions are found with the aid of a single cost function by constructing a global reproducing kernel Hilbert space (RKHS). This global RKHS is defined as the direct sum of the RKHSs corresponding to the decision boundaries of MKL and that of input data. Hence the optimal function can be represented as the direct sum of the decision boundaries under consideration.

Kumar et. al. (2012) framework has been extended to regression problems also. We also developed a nonlinear formulation, in which the optimal kernel is represented as a nonlinear combination of kernels. Such a combination of kernels results in an indefinite symmetric matrix and hence makes use of the concepts of Krein Space for finding the optimal functions.

Finally, we formulated the MKL using composite kernel functions (MKLCKF). In this MKLCKF the optimal kernel is represented as a linear combination of composite kernel

functions. Corresponding to each data point a composite kernel function is designed whose domain is constructed as the direct product of the range space of base kernels. In this way, the composite kernels make use of the information of all the base kernels for finding their image. Thus MKLCKF has three layers in which the first layer consists of base kernels, the second layer consists of composite kernels and the third layer is the optimal kernel which is a linear combination of the composite kernels. For making the algorithm more computationally effective, we formulated one more variation of the algorithm in which the coefficients of the linear combination are replaced with a similarity function that captures the local properties of the input data. With the aid of data compression techniques, the models have been applied on large data. In the case of large scale classification and regression, dictionary learning and pre-clustering approaches have been used respectively.

The efficiency of all the developed approaches was verified by applying them on real world problems and the results were found to be promising. The comparative study of all the models we developed had also been conducted.

Contents

List of Tables

List of Figures

List of Algorithms

Abbreviations

ARIMA	Autoregressive integrated moving average
DMKL	Discriminative Multiple Kernel Learning
FMKL	Feature wise Multiple Kernel Learning
GL-MKL	Group Lasso regularized Multiple Kernel Learning
GMKL	Generalized Multiple Kernel Learning
KRR	Kernel Ridge Regression
LpMKL	Lp Norm regurlarized Multiple Kernel Learning
LSH	Locally Sensitive Hashing
MKL	Multiple Kernel Learning
MOC	Model-based Overlapping Clustering
OCC	Overlapping Corelation Clustering
OC	Overlapping Clustering
RKHS	Reproducing Kernel Hilbert Space
RMSE	Root Mean Squared Error
SMO	Sequential Minimal Optimization
SPG-GMKL	Spectral Projected Gradient - Generalized Multiple Kernel Learning
SVM	Support Vector Machine
SVR	Support Vector Regression

Chapter 1

Introduction

The kernel algorithms are the class of algorithms that make use of the concept of reproducing kernel Hilbert space (RKHS) in data modeling [2, 3, 4]. The concept of reproducing kernel and RKHS [5] became popular in machine learning after the development of support vector machine (SVM) [6]. Later the kernel theory concepts were adopted in various areas of machine learning such as dimensionality reduction[7], regression[8], clustering [9], one class classification [10] etc. The performance of a kernel algorithm depends on the reproducing kernel used. Hence the development of efficient methods for finding the optimal kernel is very much essential for kernel methods .

The current available tools for kernel selection include techniques like cross validation and multiple kernel learning (MKL). The advantage of MKL is that it automatically finds the best combination of kernels from a pool of available kernels (base kernels).

The kernel methods represent the solution f of a learning problem in the form

$$f(x) = \sum_{i=1}^{N} \alpha_i \hat{k}(x, x_i) \tag{1.1}$$

where $x_i, i = 1, 2, \ldots N$ are the given inputs, $\hat{k}$ the reproducing kernel corresponding to the RKHS in which f lies and $\alpha_i, i = 1, 2, \ldots N$ are the set of parameters to be determined from the given data. Generally, in MKL algorithms, the reproducing kernel is defined as a linear combination of a set of kernels. Using this concept, (1.1) can be written as

$$f(x) = \sum_{i=1}^{N} \alpha_i \sum_{l=1}^{P} d_l k_l(x_i, x), d_l \geq 0 \tag{1.2}$$

where k_l belongs to the set of base kernels under consideration.

1.1 Multiple Kernel Learning Approaches

Various methodologies are adopted for learning the kernel from the data in MKL algorithms. The main approaches in MKL are fixed rule, heuristic, optimization, Bayesian and boosting approaches [11]. The fixed precomputed values are used as kernel weights in the fixed rule approach [12]. The heuristic approach is also widely used in approximating the kernel weights. In that methodology, the kernel weights are calculated based on heuristic approach. For example in the work [13], the conditional class probabilities of data labels are used for fixing the kernel values while that of the work [14] uses predictive performance of each kernel for approximating the kernel weights.

In optimization of parameters for the MKL algorithms, there are two ways of optimizing the kernel parameters. They are one-step method [15, 16, 17, 18], where both kernel parameters(d_l) and function parameters(α_i) are updated simultaneously till convergence and two-step method [19, 20, 21], where in the first step, one set of parameters are updated until convergence while the other is kept constant and in the second step, the updated values are used to find the optimum values of the set of parameters that are kept constant in the first step.

The MKL techniques belonging to optimization approach can be classified as similarity and structural based risk methods. The similarity based algorithms use the concept of kernel alignment [22], which deals with the measurement of correlation or similarity between two kernels. The work of [16] is an extension of target alignment work, in which the alignment between ideal kernel and combination of kernels is maximized by solving a quadratic optimization problem. The work of [17, 18] also belongs to this category and the problem is solved using the techniques like semi-definite programming and advanced gradient based methods .

The structural risk-based MKL algorithm uses the concerned learning algorithm's cost function for the risk minimization. The work of [15] is based on this approach, in which the optimal kernel is represented as a linear combination of kernels such that both the kernel parameters and function parameters are updated simultaneously till convergence on the basis of global structural risk minimization problem. In Bayesian approach, the kernel parameters are modeled with priors. These prior distributions are then solved using approaches such as multinomial probit [23]. In the boosting approach to MKL [24], a new kernel is added to the set of base kernels until a stopping criterion is reached.

The MKL techniques have been formulated using semi-supervised [25] and unsupervised approaches also [26]. The Absent MKL is another MKL variant which deals with

absent features(channel) [27]. The works of [1] describes a MKL model, in which the problem of finding the optimum kernel is modeled as a classification problem by using two separate cost functions with one for finding the parameters of the function that corresponds to the optimum kernel and the one that has to be learned from the data. The faster optimization of kernel parameters for adapting to large scale data set is detailed in [28, 29]. The nonlinear combination of kernels is discussed in the works of [30, 20, 31].

In the works of [32, 33], the MKL is formulated using feature-wise kernel selection, that is, a kernel is selected for each feature and the reproducing kernel is formulated as a linear combination of such kernels. Such a MKL formulation also helps to incorporate the feature selection in the learning problem, as the prominent features can be selected on the basis of the weights associated with them. The MKL framework has been explored in the feature fusion domain also [32].

1.2 Organization of Thesis

The thesis is organized to have the following chapters in addition to the chapters that describe the introductory concepts and conclusion. The gist of each of those chapters is given below.

1.2.1 Theoretical Foundations

The theoretical concepts used for formulating the methods we propose in this thesis are given in this chapter.

1.2.2 Localization of Multiple Kernel Learning using Feature Clustering

A recent approach to MKL is associating each feature with a kernel. The main drawback of this approach is the computational cost associated with the selection of kernel in the case of high dimensional data. In this chapter, we propose clustering of features in MKL to overcome the computational burden in feature wise localization of kernels. The features are first clustered in this technique and then an appropriate kernel is assigned to each cluster. Hence each of the base kernels is defined on the subspace generated by the features belonging to the cluster to which it is assigned. The optimal reproducing kernel is then represented as a linear combination of such kernels. This method helps to impose dimensionality reduction

of the data in an implicit manner, which in turn increases the performance of the kernel learning algorithm.

1.2.3 Classification Approach for Multiple Kernel Learning

In this chapter, the MKL is formulated as a supervised classification problem. We dealt with binary classification data and hence the data modeling problem involves the computation of two decision boundaries of which one related with that of kernel learning and the other with that of input data. The authors of [1] used two separate cost functions for finding the parameters corresponding to those two decision boundaries. In the approach we developed they are found with the aid of a single cost function and for that we formulated the problem of finding the unknown parameters as a single classification task. This is achieved by representing the problem of finding the optimum kernel as a supervised classification model, such that the the corresponding decision boundary is assumed to lie in a RKHS $\mathcal{F}^*$ and the decision boundary that separates the data into two classes lies in a RKHS $\mathcal{F}$. A global RKHS, $\mathcal{F}^\dagger = \mathcal{F} \oplus \mathcal{F}^*$ is constructed for searching the unknown functions with the aid of a single cost function.

We incorporated the proposed techniques in SVM equipped with sequential minimal optimization (SMO) and used two step optimization method for finding the unknown parameters. Since functional representation of MKL results in an exponential increase in the training points associated with model development, the empirical kernel mean maps of two classes under study is used for effective data compression.

We developed batch as well as online versions of the proposed framework. The efficiency of the models were being verified by comparing their performance with that of existing techniques using simulated and real world data sets. On the basis of performance measures like accuracy and F measure, our models performance were found to be better than other existing algorithms.

1.2.4 Regression Approach for Multiple Kernel Learning

In this chapter, the work of [1] is extended to regression problems. The formulation of MKL as a regression problem requires the generation of outputs by ideal kernel. We proved that the ideal kernel function for regression is same as that of classification. As MKL formulation demands more space requirements, supervised pre-clustering technique have been used for selecting the vital data points. The linear as well as non linear combination of base kernels are considered for developing the model. The function represented using

non linear combination of base kernels may not be positive semi definite and hence Krein space concepts are used for formulation. The procedure for finding the parameters of the models using rigde regression is also discussed. For that we derived the indefinite kernel ridge regression in Krein space. The efficiency of the proposed models were verified using real world data sets and the results were found to be promising.

1.2.5 Learning Kernel using Composite Kernel Functions

This chapter discusses the formulation of MKL using composite kernel functions for finding the best combination of kernels from a given P base kernels for machine learning problems such as classification and regression. With reference to each data point a composite kernel function is constructed such that it makes use of the information of all the given P base kernels for finding the image at each of the points in its domain. The two variants of this formulation have been derived. In the first variant, the optimal kernel is represented as a linear combination of newly designed kernels. As each composite kernel function is built upon a data point, we introduced a second variant in which the coefficients of the linear combination are replaced with a neighborhood function of the reference data point. This representation makes the algorithm more computationally efficient. We verified the efficiency of the proposed models using real world data sets and compared their performance with existing techniques. The proposed methods showed excellent performance. Of the two variants of the approach, the performance of the second variant was found to be better.

As the data increases, the number of training points as well as the number of terms in the proposed kernel increases. Thus the overall complexity of the problem is increased. In order to tackle this problem, subset selection approach [34] is followed for regression and dictionary learning approach ([35]) for classification. We did experimental analysis by incorporating these two in the proposed methods and the results were found to be better than the other existing techniques we used for comparison.

1.2.6 Real World Applications

The MKL models proposed in this thesis as well as the state-of-the- art MKl models were applied on two real world applications and made a comparative study of their performance, whose description is given in this chapter.

1.3 Major Contributions of the Thesis

The important contributions of the thesis are listed below:

1. The MKL problem is formulated by representing the optimal kernel as a linear combination of kernels that are defined over the clusters of features. For that a feature clustering algorithm has been formulated. Also for making the algorithm more computationally effective, different techniques have been applied to reduce the length of feature vector.

2. For analysing the binary classification data, we formulated the problem of finding the unknown parameters of the optimum kernel and the function that has to be learned from the data (binary classifier) as a classification task. A single cost function is used to find the parameters associated with optimum kernel as well as the binary classifier.

3. We designed MKL as a regression problem for analyzing the regression data. For that it is proved that the ideal kernel for this formulation is same that of MKL's classification model. The model is developed using linear as well as nonlinear combination of kernels. The reproducing kernel Krein space concepts are used for formulating the non linear version. The indefinite kernel ridge regression is derived in Krein space and it is used for finding the parameters of non linear model.

4. The optimal kernel is represented as the linear combination of composite kernel functions. For that, corresponding to each data point a composite kernel function is designed whose domain is constructed as the direct product of the range space of base kernels, so that the composite kernels make use of the information of all the base kernels for finding their image. Thus this model has three layers in which the first layer consists of base kernels, the second layer consists of composite kernels and third layer is the optimal kernel which is a linear combination of the composite kernels. For making the algorithm more computationally effective, one more variation of the algorithm is formulated in which the coefficients of the linear combination are replaced with a similarity function that captures the local properties of the input data.

Chapter 2

Theoretical Foundations

The basic methods we used for building the work described in this thesis are given in this chapter. A brief description of kernel theory and algorithms is given. A couple of relevant MKL algorithms which are necessary for the main work in this thesis are also given. The feature clustering algorithms and big data approaches in the multiple kernel learning paradigm are also discussed.

2.1 Kernel Methods

Let $\{(x_1, y_1), (x_2, y_2) \dots (x_N, y_N)\}$ be the given data, where $x_i \in \mathcal{X} \subseteq \mathbb{R}^n$ and $y_i \in \mathbb{R}, i = 1, 2, \dots N$.

The kernel methods make use of the concepts of Reproducible Kernel Hilbert Space (RKHS)[2, 36, 4] for data modeling. Let f be the function that generates the data and let it belong to RKHS $\mathcal{F}$. Therefore by kernel theory, $f(x)$ can be written as

$$f(x) = \sum_i \alpha_i k(x_i, x) \tag{2.1}$$

where $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is the reproducing kernel of $\mathcal{F}$ and $\alpha_i \in \mathbb{R}$. As kernel methods use Tikhonov regularization concepts, using the Representer theorem [37], (2.1) becomes

$$f(x) = \sum_{i=1}^{N} \alpha_i k(x_i, x) \tag{2.2}$$

2.1.1 Support Vector Machine (SVM)

The solution of the data modeling problem in SVM can be represented as

$$f'(x) = f(x) + b$$

where $f(x)$ is as given in (2.2) and $b \in \mathbb{R}$ is the bias. The cost function of SVM classification can be written as

$$\begin{aligned} &\min_{f \in \mathcal{F}, b \in \mathbb{R}} \frac{1}{2}\|f\|^2 + C\sum_i \xi_i \\ &subj.\ to \\ &\quad y_i(f'(x_i) - 1 + \xi_i) \geq 0, \quad i = 1, 2 \ldots N \end{aligned}$$

where $\xi_i = \max(0, 1 - y_i f'(x_i))$ and $C > 0$ is the regularization parameter.

The dual function of SVM with the reproducing kernel k and dual parameter α_i can be written as

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ &\quad subj.\ to. \\ &\qquad \sum_{i=1}^{N} \alpha_i y_i = 0 \\ &\qquad 0 \leq \alpha_i \leq C, \quad i = 1, 2 \ldots N \end{aligned} \tag{2.3}$$

The selection of kernel plays an important role in the performance of the algorithm. The MKL techniques help in selecting the optimal reproducing kernel for a given data.

2.1.2 Support Vector Regression (SVR)

For regression problems, it searches for the optimal hyperplane that passes through the data points in RKHS with a marginal tolerance of error. The cost function of SVM regression

can be written as

$$\min_{f\in\mathcal{F},b\in\mathbb{R}} \frac{1}{2}\|f\|^2 + C\sum_i (\xi_i + \xi_i^*)$$

$$\begin{aligned} subj.\ to \\ & y_i - f(x_i) - b \le \epsilon + \xi_i, \quad i = 1, 2 \dots N \\ & f(x_i) + b - y_i \le \epsilon + \xi_i^*, \quad i = 1, 2 \dots N \\ & \xi_i, \xi_i^* \ge 0, \quad i = 1, 2 \dots N \end{aligned}$$

where and $C > 0$ is the regularization parameter.

The dual function of SVR with the reproducing kernel k and dual parameter $\alpha_i \& \alpha_i^*$ can be written as

$$\begin{aligned} \max_{\alpha,\alpha^*} \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i^*) - \epsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) \\ subj.\ to. \\ \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \\ 0 \le \alpha_i, \alpha_i^* \le C, \quad i = 1, 2 \dots N \end{aligned} \tag{2.4}$$

2.2 Kernel Ridge Regression

The risk minimization problem for the kernel ridge regression can be written as below.

$$\min_{f\in\mathcal{F}} \quad \frac{1}{2}\sum_{i=1}^{N}(f(x_i) - y_i)^2 + \frac{\lambda}{2}\|f\|^2$$

where y_i is the output label and $\lambda > 0$ is the regularization parameter. By substituting $f(x) = \sum_{i=1}^{N} \alpha_i k(x_i, x)$, the above cost function becomes

$$\min_{\alpha\in\mathbb{R}^N} \quad \frac{1}{2}\|K\alpha - y\|^2 + \frac{\lambda}{2}\alpha^T K\alpha$$

where K is the kernel matrix and y is the output vector. The optimal value of α is given

by

$$\alpha = (K + \lambda I)^{-1} y \tag{2.5}$$

2.3 Multiple Kernel Learning Algorithms

In multiple kernel learning algorithms, there are many ways of learning the kernel k. One of the techniques is to represent it as a linear combination of base kernels under consideration.That is

$$\mathbf{k}(x, z) = \sum_{l=1}^{p} d_l k_l(x, z) \tag{2.6}$$

where kernel weights $d_l \geq 0, l = 1, 2 \ldots p$, p is the number of base kernels and $x, z \in \mathcal{X}$. By substituting the above in SVM classification dual function,

$$\begin{aligned} J = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \sum_{l=1}^{p} d_l k_l(x_i, x_j) \\ subj.\ to \\ \sum_i \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \\ d_l \geq 0 \end{aligned} \tag{2.7}$$

SimpleMKL [19] is one of the prominent works in multiple kernel learning algorithms. In SimpleMKL, the optimization of α and d_l are carried out alternatively as two steps so that in first step the α's are updated using conventional algorithms such as SVM-SMO [38] and in second step the d_l's are updated using gradient descent algorithm. These two steps of optimization are iterated till convergence. In order to make the kernel weights sparse, additional constraints have also been incorporated. Therefore in this case the dual function

is

$$
\begin{aligned}
J(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \sum_{l=1}^{p} d_l k_l(x_i, x_j) \\
subj.\ to \\
\sum_i \alpha_i y_i = 0 \\
0 <= \alpha_i <= C \\
\sum_l d_l = 1 \\
d_l >= 0
\end{aligned}
\tag{2.8}
$$

The regularization concepts are incorporated in the above formulation in Generalized MKL(GMKL) [21] and hence in that case the problem formulation becomes

$$
\begin{aligned}
\min_{f \in \mathcal{F}, b, d \in \mathbb{R}} \frac{1}{2}\|f\|^2 + \sum_i l(y_i, f(x_i)) + r(d) \\
subject\ \ to \quad d \geq 0
\end{aligned}
$$

where r is the regularizer and l is the loss function.

2.4 Large Data Approaches in Kernel Methods

The main disadvantage of kernel methods is their computational complexity which scales as $O(N^3)$ where N is the number of training points. Various approaches like boosting ([39]), decomposition techniques such as SVMTorch ([40]), sequential minimization optimization ([38]) have been developed for scaling the kernel algorithms to large data sets. The work of [41] improves computational performance of kernel machines by using a fixed Gaussian kernel whereas in [29], the authors used used semi infinite linear programming based optimization algorithm for decreasing the computation complexity.

The data compression is an efficient tool for dealing with large scale data. The work described in the papers[42, 43, 44] use minimum enclosing ball concepts to select representative points. The drawback of this approach is described in [45]. The work [46] used squashing for selecting a working set for training. The authors of the work [34] developed

a supervised preclustering approach for scaling kernel based regression by making use of the concepts of uniform continuity and compactness. In our work, we use this compression technique for large scale regression problems and dictionary based learning for classification. The description of these compression techniques is given below.

2.4.1 Supervised Preclustering

In the pre-clustering approach developed by [34], the function f to be learned is uniformly continuous, by assuming that it lies in a continuous RKHS $\mathcal{F}$, having the domain of its members a compact set $\mathcal{X}$. The idea of uniform continuity is used to define a similarity measure on the function to be estimated, As the function, f, is uniformly continuous, corresponding to similarity measure $\epsilon > 0$, there exists a radius, δ, independent of $x \in \mathcal{X}$, such that

$$\hat{d}(f(x), f(x')) < \epsilon \; \forall \; x' \in B(x, \delta) \tag{2.9}$$

where $B(x, \delta)$, is an open ball of radius δ in input space and $\hat{d}$ is a suitable metric on $\mathbb{R}$. An open ball $B(x, \delta)$ in the input space is called a *cluster* if all points associated with it satisfies (2.9). The basic idea of pre-clustering is that any data points which satisfy (2.9) can be considered to be "similar" and therefore form pre-clusters. The centers of the clusters are then used as a sparse data set for the function estimation. The output information has also been used to form clusters and hence it is a supervised clustering.

The working procedure of the algorithm is as follows. Corresponding to the given similarity measure ϵ, the algorithm finds the radius δ in an iterative manner. In an iteration, an open ball $B(x, \delta)$ is formed in a greedy manner and all those non-center points that satisfy (2.9) get eliminated. Thus in each iteration, the set of training points consists of the centers of the open balls and those points that do not satisfy (2.9). The algorithm gets terminated if all the training points under consideration satisfy (2.9). Otherwise δ gets updated using the formula $\delta := \delta - h$, where $h > 0$ is the step length and moves to the next iteration.

2.4.2 Data Compression using Dictionary Learning

In dictionary learning, the dictionary consists of subset of data points such that all the input data can be represented as a linear combination of them. Let X be the input data and D be the dictionary atoms and W be the sparse weights. Then the problem of dictionary learning

can be formulated as

$$\min_{D,W} \|X - DW\|^2 + \lambda \|W\|_0 \tag{2.10}$$

where $\lambda > 0$ is the regularization parameter. SVDD [47] is one of the state-of-the-art dictionary learning algorithms which produces sparse representation of input data. It is an iterative method that updates sparse weights and dictionary atoms in a simultaneous manner. [48] uses least square formulation with recursive approach in solving the dictionary learning problem. [35] discusses a label consistent dictionary learning algorithm, where it uses label information in the cost function for learning the dictionary. We use the same algorithm in our model for applying over large data in the case of classification.

Chapter 3

Localization of Multiple Kernel Learning using Feature Clustering

The multiple kernel learning generally formulates the optimal kernel as a combination of kernels, in which all of them are defined on the same input space. In order to understand the relevance of each feature, there have been studies in which a separate kernel is assigned for each of them. In [32], MKL is formulated using feature-wise kernel selection, that is, a kernel is selected for each feature and the reproducing kernel is formulated as the linear combination of such kernels. Such a MKL formulation also helps to incorporate the feature selection in the learning problem, as the prominent features can be selected on the basis of the weights associated with them. The main drawback of this method is the computational cost associated with it in the case of high dimensional data.

This chapter describes the MKL frame work we propose using feature-wise kernel selection, which is named as feature wise multiple kernel learning (FMKL). A clustering phase is introduced in the FMKL model for overcoming the computational burden associated with the feature wise kernel selection. That is, the features are first clustered and then a suitable kernel is selected for each cluster. We also propose a feature clustering algorithm which is named as Overlapping Clustering(OC) algorithm, whose description is given in this chapter. We incorporate the developed MKL model with SVM classification for experimental analysis.

The next section describes the feature clustering approach for MKL while the section 3.2 describes the proposed model. The experimental setup and results of empirical study conducted on various datasets are described in section 3.3. The concluding remarks are given in section 3.4.

3.1 Feature Clustering Approach for MKL

The selection of kernels on the basis of the characteristics of features is an important approach in MKL. A new reproducing kernel can be defined as a linear combination of kernels where each kernel corresponds to a feature whose consistency is proved in [49]. The concept of feature wise kernel combination for heterogeneous features has been applied in areas like object recognition. One such work is [32] in which a separate kernel is selected for every feature and linear combination of those kernels is used as the reproducing kernel. This procedure is formulated as Group Lasso MKL [50] for finding the kernel weights. In feature wise combination of kernels, for some features the weight may be close to zero and hence feature selection is implicitly performed in this method.

The procedure of finding the kernel for each feature is computationally expensive. More than one feature may perform well with same kernel. Hence introducing a clustering phase for grouping the similar features in such MKL models makes them more computationally effective.

3.1.1 Feature Grouping Algorithms

The feature clustering or feature grouping is usually applied for very large dimensional dataset for reducing the dimension and thereby improving the computation time. [51] is a popular feature clustering algorithm in the domain of text classification and it uses information theory, which is an effective tool for natural language processing. [52, 53] depicts feature clustering algorithms for text categorization in which first one uses distributional clustering for finding similar words and the latter uses information theory for clustering the features. The works of [54] and [55] are co-relation based methods whereas the work in the paper [56] is an overlapping clustering algorithm. In FMKL model, we introduce feature clustering for finding the similar features, which helps to associate a kernel to each cluster.

3.2 Feature wise Localized Multiple Kernel Learning

The development of model constitutes three modules as shown in Fig. 3.1.

1. **Feature Vector Construction** - A vector corresponding to each feature termed as *Feature Vector* is created and clusters are formed using them. Four types of feature vectors are used in this approach.

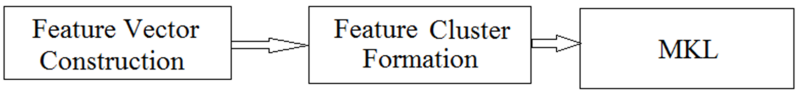

Figure 3.1: Feature wise Localized MKL

2. **Feature Cluster Formation** - A clustering algorithm is applied on the feature vectors in this stage.

3. **MKL** - After forming the clusters, each cluster is assigned with a suitable kernel, which can be found by techniques like cross validation. Such kernels are then linearly combined to form the reproducing kernel and it is then plugged into the kernel algorithm under consideration. We chose the SVM algorithm in the experiments we did.

Given below is a description of these three modules.

3.2.1 Feature Vector Construction

Consider the input data matrix

$$M = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots\vdots\vdots & \vdots \\ x_{N,1} & x_{N,2} & \dots & x_{N,n} \end{pmatrix}$$

where $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,n})^T$. The feature vector corresponding to each feature will be the input for clustering which can be represented as follows:

$$F = \begin{pmatrix} f_1^T \\ f_2^T \\ \vdots \\ f_n^T \end{pmatrix} \tag{3.1}$$

where f_i corresponds to feature vector for the feature i. The feature vector is constructed in four different ways, whose explanation is given below.

3.2.1.1 Actual Values

In this case, all the feature values are taken into consideration, that is, the feature vector for k^{th} feature is

$$f_k = [x_{1,k}, x_{2,k}, \ldots x_{N,k}]^T \tag{3.2}$$

This approach is not suitable for high dimensional data, since the computational complexity increases with the increase in dimension.

3.2.1.2 Mean

The clustering data for this approach is the mean of the values of features, belonging to different classes under consideration. The feature vector for the k^{th} feature in this case is

$$f_k = [\mu_k^+, \mu_k^-]^T \tag{3.3}$$

where μ_k^+ is the mean of k^{th} feature of positive class and μ_k^- is the mean of k^{th} feature of negative class. The mean can be calculated as:

$$\begin{aligned} \mu_k^+ &= \frac{\sum_{i=1}^N x_{ik}\{y_i = +\}}{\sum_{i=1}^N 1\{y_i = +\}} \\ \mu_k^- &= \frac{\sum_{i=1}^N x_{ik}\{y_i = -\}}{\sum_{i=1}^N 1\{y_i = -\}} \end{aligned} \tag{3.4}$$

3.2.1.3 Conditional Expectation

If we know the individual distributions of the features, the concept of conditional expectation can be applied for finding the feature vector. Using maximum likelihood techniques we can find the parameters of the distribution. Hence in this case the feature vector for the k^{th} feature consists of the conditional expectation of corresponding classes, that is,

$$f_k = [E_k^+, E_k^-] \tag{3.5}$$

where E_k^+ and E_k^- are the conditional expectation of the positive and negative class respectively. For example, if a feature follows normal distribution, the conditional distribution is

determined as follows:

$$\begin{aligned} E(x^k/y_i = +) &= \sum_{i=1}^{N} x_i^k P(x_i^k/y_i = +) \\ E(x^k/y_i = -) &= \sum_{i=1}^{N} x_i^k P(x_i^k/y_i = -) \end{aligned} \tag{3.6}$$

where

$$\begin{aligned} P(x_i^k/y_i = +) &= \frac{1}{\sqrt{2\pi}\sigma_k^+} \exp\left(\frac{-\|x_i^k - \mu_k^+\|^2}{2{\sigma_k^+}^2}\right) \\ P(x_i^k/y_i = -) &= \frac{1}{\sqrt{2\pi}\sigma_k^-} \exp\left(\frac{-\|x_i^k - \mu_k^-\|^2}{2{\sigma_k^-}^2}\right) \end{aligned} \tag{3.7}$$

Here μ_k^+ and μ_k^- can be calculated using (3.4) and

$$\begin{aligned} \sigma_k^+ &= \frac{\sum_{i=1}^{N}(x_i^k - \mu_k^+)^2\{y_i = +\}}{\left(\sum_{i=1}^{N} 1\right) - 1\{y_i = +\}} \\ \sigma_k^- &= \frac{\sum_{i=1}^{N}(x_i^k - \mu_k^-)^2\{y_i = +\}}{\left(\sum_{i=1}^{N} 1\right) - 1\{y_i = +\}} \end{aligned} \tag{3.8}$$

3.2.2 Clustering Techniques

For clustering the features we use three different clustering techniques, namely, k-means, spectral and the proposed clustering algorithm OC. In k-means as well as spectral clustering, the number of clusters has to be mentioned where as in OC, it uses a similarity threshold which determines the number of clusters. The OC uses fuzzy measures of similarity and hence a data point can be a member of more than one cluster. It also makes use of hierarchical clustering concepts.

3.2.2.1 Overlapping Clustering (OC)

The clustering of similar features is studied for various aspects in machine learning. The paper [57] describes a feature clustering algorithm based on a similarity measure. In that work the features are words in text processing and clustering algorithm is used for the

purpose of dimensionality reduction. In that algorithm, all the features are sequentially taken and added to a cluster if the similarity measure is greater than a threshold. When the similarity is less than the threshold, a new cluster is formed. The feature selection is greedy in manner and hence the cluster formation depends on the order of the features. The overlapping clustering algorithm (OC) is an extended version of this algorithm. In this method, cluster formation is independent of the order of features. The description of OC is given below.

The OC can be considered as a two stage algorithm in which the centroids are calculated in the first stage and data points are added to clusters in the second stage. This algorithm uses the measure of similarity between a data point and a centroid in the scale range from 0 to 1. We used Gaussian metric to calculate the similarity measure in which the similarity between a cluster centroid c and data point x can be defined as

$$\partial_c(x) = \exp(-\frac{\|x - c\|^2}{\beta}) \tag{3.9}$$

where $\beta > 0$.

The algorithm is shown in 1 where OC is main method which calls both phases of the algorithm in sequential order. The first phase of the algorithm selects the centroid for each cluster. It uses Gaussian similarity metric defined in (3.9) for calculating the similarity between a feature and a centroid.

The algorithm is similar to hierarchical clustering where each data point (feature vector) is being initially considered as a cluster centroid as shown in Fig. 3.2. Then a level for threshold t is set for merging as in hierarchical clustering. It is initialized with a high value (line 7) and will be decremented in each iteration for hierarchically combining the cluster centroids (line 11). In each iteration, the pairs of centroids whose similarities is greater than t are identified (line 9). The resulting pairs are then sorted in descending order on the basis of similarity so that the most similar features is grouped together. From this sorted list, each pair is added to another list for merging if both centroids of that pair have not been selected earlier for merging. Even if one or two pairs got missed out for merging in an iteration, it is selected for merging in the next iteration. Then the clusters associated with these identified pairs are merged and the new cluster centroid is calculated (line 10). Suppose c_i is the centroid of cluster i having m elements and c_j is the centroid of cluster j

Algorithm 1 Centroid Calculation Algorithm

procedure OC(datapoints)
    Calculate centroids using $CentroidCalculation$
return Do clustering using $doCluster$
end procedure
procedure CENTROIDCALCULATION(datapoints) ▷ The first phase of Overlapped Hierarchical Clustering Algorithm
    Initialize the threshold t with a high value;
    Initialize centroids of each cluster as the feature point itself.;
    while $t >= T$ **do**
        calculate pairs ▷ Find all the pairs of cluster centroids with similarity between is greater than t
        $mergepairs$ ▷ Merge the clusters corresponding to each pairs
        Decrement the t for next step
    end while
end procedure

procedure DOCLUSTER(centroids) ▷ Clustering of data points based on centroids
    for $i = 1 \rightarrow n$ **do**
        for $j = 1 \rightarrow p$ **do**
            $d \leftarrow \partial_{c_j}(f_i)$ ▷ Distance between j^{th} cluster centroid and i^{th} point using (3.9)
            if $d > T$ **then**
                Add point i to the cluster j
            end if
        end for
    end for
end procedure

having n elements. If these two clusters are merged, the centroid of the resulting cluster is

$$new\_centroid = \frac{(m * c_i) + (n * c_j)}{(m + n)} \tag{3.10}$$

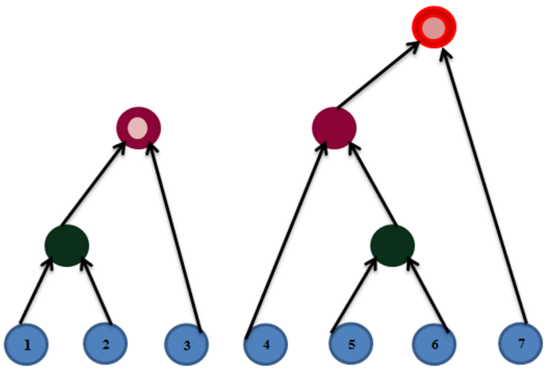

Figure 3.2: OC Algorithm Phase 1

This process continues until the step threshold $t < T$, for some threshold T, which determines the number of clusters. T can be determined using cross validation techniques.

In phase two of OC, the membership of each data point with a cluster is determined. A data point is associated with a cluster if its similarity metric is greater than the threshold T (Fig. 3.3). Hence a data point might be associated with more than one cluster. After the formation of clusters, each cluster is assigned with a suitable kernel using cross validation methods.

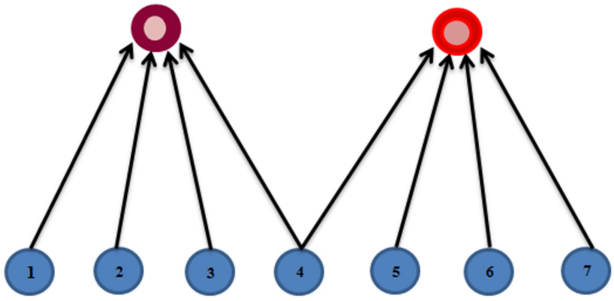

Figure 3.3: OC Algorithm Phase 2

The OC algorithm differs from the state-of-the-art techniques in overlapping clustering such as Model-based Overlapping Clustering (MOC)[56] and Overlapping Corelation Clustering (OCC)[55]. In [56] the clustering algorithm is formulated as a linear equation where the linear operator is a membership matrix which maps the data to an arbitrary space.

In [55] the cost function is defined based on co-relation between the data points. The OC is formulated using the principles of hierarchical clustering.

3.2.2.2 Time Complexity Analysis

The average height of the tree formed as part of centroid calculation algorithm is $\log n$, where n is the number of features. The maximum number of iterations required for finding similarity between all the pairs from a set of n features is n^2 and therefore time complexity is $\Theta(n^2)$. The merging of centroid uses (3.10) for calculating a new centroid. The complexity for merging is $\Theta(1)$ since no loops are involved. The overall time complexity of first procedure can be approximated as $\Theta(\log n) * \{\Theta(n^2) + \Theta(1)\}$ which is $\Theta(n^2 \log n)$. The complexity of the second procedure is $\Theta(pn)$ where n is the number of features and p is the number of centroids calculated by phase 1 procedure.

The total time complexity of OC algorithm is $\Theta(n^2 \log n) + \Theta(pn)$ which can be approximated as $\Theta(n^2 \log n)$, since $p < n$.

3.2.3 Multiple Kernel Learning using Clusters of Features

The validity of feature cluster wise combination of kernels can be proved by using the kernel construction theory from [3, 4], whose description is given below. Consider a scalar $\alpha \geq 0 \in \mathbb{R}$. Then

$$\begin{aligned} k'(x, z) &= \alpha k(x, z) \\ k''(x, z) &= k_1(x_a, z_a) + k_2(x_b, z_b) \end{aligned} \tag{3.11}$$

where, $x = x^a \cup x^b \in \mathcal{X}$ and $z = z^a \cup z^b \in \mathcal{X}$ are also valid kernels defined on $\mathcal{X} \times \mathcal{X}$. Here k_1 and k_2 are kernels on their respective spaces.

For a given data set, let p clusters be formed. Let x^{C_l} consists of features of x, that falls in the l^{th} cluster such that $x = \bigcup_{l=1}^{p} x^{C_l}$.

Let $k_l, l = 1, 2, \ldots p$ be the kernels defined on the subspace generated by the features in l^{th} cluster, $C_l, l = 1, 2, \ldots p$. Then from (3.11),

$$k_{c_l} = d_l k_l(x^{C_l}, z^{C_l}), d_l \geq 0, l = 1, 2, \ldots p \tag{3.12}$$

and

$$\mathbf{k}(x, z) = \sum_{l=1}^{p} k_{c_l} \tag{3.13}$$

are well-defined kernels.

Hence,

$$\begin{aligned} f(x) &= \sum_{i=1}^{N} \alpha_i k(x_i, x) \\ &= \sum_{i=1}^{N} \alpha_i \sum_{l=1}^{p} d_l k_l(x_i^{C_l}, x^{C_l}) \end{aligned} \tag{3.14}$$

On the basis of the above theory, the dual function of the SVM can be written as

$$\begin{aligned} J = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \sum_{l=1}^{p} d_l k_l(x_i^{C_l}, x^{C_l}) \\ subj.\ to \\ \sum_i \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \\ \sum_l d_l = 1 \\ d_l \geq 0 \end{aligned} \tag{3.15}$$

The constraint $d_l \geq 0$ keeps the new reproducing kernel to be a positive semi-definite one. The constraint $\sum_l d_l = 1$ makes the solution of kernel weights more sparse since this is an l_1-norm constraint[19].

Two variations of the above equations are analyzed. In one formulation, constraint $\sum_l |d_l| = 1$ is removed and in second d_l is fixed as $\frac{1}{p},\ p \in \mathbb{R} - \{0\}$ for all the clusters. The model did not perform well in these two cases.

The optimization of the α and d_l is implemented in the same way as SimpleMKL[19] by Rakotomamonjy et. al. As explained earlier, the selection of optimum kernel k_l for each feature cluster can be done using cross validation techniques.

3.3 Experiments

The proposed method was implemented in Java over the JKernelMachine framework [58, 59] and tested in a machine of intel i5 with 6 GB RAM. The setup, experiment methodology and results are explained below.

3.3.1 Experimental Setup

A layer of code is implemented over the jKernelMachine framework with proper abstraction. The datasets used are detailed in Table 3.1 which are binary class classification problems taken from UCI repository [60] and IDA benchmark repository [61]. The heart disease dataset is actually 5 class classification dataset, which is converted to a binary classification where the classes are mapped to two classes, namely, with heart disease or no heart disease.

Table 3.1: Datasets used for FMKL experiments

Dataset	**Repository**	**Dimension**	**Data Points**
Heart	UCI	13	303
Ionosphere	UCI	33	351
Sonar	UCI	60	208
Parkinsons	UCI	22	195
Pima	UCI	8	768
Musk 2	UCI	166	476
Arrythmia	UCI	276	452
WDBC	UCI	30	569
Twonorm	IDA	20	7400
Ringnorm	IDA	20	7400

The six different variations of proposed model are listed in Table 3.2.

Table 3.2: Proposed FMKL Models

Model Name	**Feature Vector**	**Clustering**
FMKL Normal	Normal	Overlapping
FMKL Spectral	μ	Spectral
FMKL KMeans	μ	K-Means
FMKL P_KMeans	Conditional Exp	K-Means
FMKL OVLP	μ	Overlapping
FMKL P_OVLP	Conditional Exp	Overlapping
FMKL D_OVLP	Dictionary Learning	Overlapping

The base kernels were generated from four reproducing kernel functions which are Laplacian kernel, Gaussian kernel, polynomial kernel and sigmoidal kernel. The parameters of Laplacian and Gaussian kernel were chosen from the set $(\frac{1}{2^{-10}}, \frac{1}{2^{-9}}, ..., \frac{1}{2^{10}})$. The

degree of the polynomial in polynomial kernel was chosen from 1 to 4 for generating the base kernels.

The models used for comparison with FMKL models were SimpleMKL[19] (SVM-SMO + MKL), GL-MKL [32] and SVM-SMO[38] for which the hyperparameters were optimized using 5-fold cross validation. In FMKL models, suitable kernel for each cluster was determined using 5 fold cross validation techniques.

The number of kernels in the proposed models is equal to number of clusters formed as part of the clustering algorithm. In SimpleMKL, 10 best performing kernels were used, which were selected by using 5-fold cross validation technique.

The parameter C of SVM and the threshold T of OC algorithm and the β of (3.9) were estimated using 5-fold cross validation method. The β was 0.3 for all the experiments with mean and conditional expectation as feature vectors and 0.09 for normal feature vector. This difference is mainly due to the dimension of feature point for clustering. The normal feature vector has dimension equal to number for training points (N) where other models have dimension as 2.

The performance measures used were accuracy, $F_{measure}$ and time of execution. The accuracy and F-measure are given by

$$Accuracy = \left(\frac{no\ of\ correct\ predictions}{N_t}\right) * 100$$
$$F_{measure} = 2 * \frac{precision * recall}{precision + recall}$$

where N_t is the number of testing points.

3.3.2 Results and Discussions

The result of the experiment is shown in the Table 3.3. The table consists of the results of each of the ten models with all the eight datasets.

Table 3.3: FMKL Experiment Results

Models		Heart	Ionosphere	Wdbc	Sonar	Parkinsons	Pima	Musk (V2)	Arrhythmia
SMO SVM	**Accuracy**	83.07 ± 3.30	93.36 ± 2.83	96.53 ± 1.65	84.24 ± 4.76	87.14 ± 4.70	76.73 ± 2.56	91.19 ± 2.21	78.48 ± 4.40
	F-Measure	80.89 ± 4.08	94.91 ± 2.26	96.87 ± 1.87	82.28 ± 5.68	91.12 ± 2.96	60.36 ± 4.38	90.13 ± 2.73	80.18 ± 6.55
	Time	64.59	106.19	57.71	27.94	24.79	27.05	230.98	36.82
	No. of Kernels	1	1	1	1	1	1	1	1
Simple MKL	**Accuracy**	82.78 ± 2.87	93.87 ± 2.20	97.06 ± .83	85.69 ± 4.47	87.39 ± 4.01	75.76 ± 2.86	90.51 ± 2.68	77.53 ± 3.71
	F-Measure	81.13 ± 3.45	95.26 ± 1.75	97.17 ± .64	83.77 ± 5.36	91.98 ± 2.65	61.61 ± 4.09	88.66 ± 3.10	79.96 ± 3.77
	Time	125.70	150.56	172.31	85.54	85.43	102.34	1380.65	132.28
	No. of Kernels	10	10	10	10	10	10	10	10
GL MKL	**Accuracy**	83.03 ± 3.60	92.60 ± 2.44	97.36 ± .83	84.08 ± 5.32	85.80 ± 3.76	76.40 ± 2.14	91.53 ± 2.39	75.77 ± 2.94
	F-Measure	81.56 ± 4.17	94.33 ± 1.98	97.76 ± 0.66	83.67 ± 5.10	90.84 ± 2.66	61.54 ± 3.72	90.46 ± 2.42	78.30 ± 2.56
	Time	75.62	105.72	117.35	83.79	56.19	102.37	280.38	103.46
	No. of Kernels	10	10	10	10	10	10	10	10
FMKL Spectral	**Accuracy**	83.74 ± 3.47	93.86 ± 1.99	97.62 ± 1.07	82.47 ± 4.30	87.12 ± 5.12	75.69 ± 2.03	90.42 ± 2.46	78.88 ± 2.76
	F-Measure	81.54 ± 4.14	95.22 ± 1.64	98.08 ± 0.90	79.78 ± 5.33	91.11 ± 3.39	61.38 ± 3.78	88.67 ± 2.81	81.09 ± 2.59
	Time	80.45	124.97	154.21	91.67	134.27	108.78	1886.91	966.57
	No. of Kernels	2	2	2	2	2	2	2	3
FMKL K Means	**Accuracy**	84.07 ± 3.41	94.18 ± 2.06	97.96 ± .90	85.53 ± 4.51	88.17 ± 3.89	75.86 ± 3.41	91.21 ± 2.57	78.53 ± 3.53
	F-Measure	81.56 ± 4.06	95.52 ± 1.65	98.37 ± .71	83.70 ± 5.14	92.04 ± 2.75	57.49 ± 2.01	90.03 ± 2.97	81.08 ± 2.37
	Time	82.34	148.26	162.30	77.02	65.34	94.04	1675.25	109.48
	No. of Kernels	4	3	2	2	3	2	3	3
FMKL P_Kmeans	**Accuracy**	83.25 ± 3.72	93.42 ± 2.40	97.25 ± 1.01	84.73 ± 5.82	87.70 ± 10.07	75.67 ± 3.57	92.69 ± 1.86	78.22 ± 2.97
	F-Measure	81.28 ± 4.27	95.04 ± 1.82	97.84 ± .77	82.20 ± 7.01	91.24 ± 10.26	59.54 ± 1.98	91.36 ± 2.04	80.93 ± 2.38
	Time	88.51	259.20	153.06	54.03	127.86	98.34	1656.75	1567.57
	No. of Kernels	4	2	3	2	3	2	2	2
FMKL Normal	**Accuracy**	83.25 ± 3.62	93.46 ± 2.06	96.56 ± 1.58	85.00 ± 4.35	87.21 ± 8.04	76.45 ± 1.38	91.47 ± 2.45	75.11 ± 2.32
	F-Measure	80.73 ± 4.86	94.95 ± 1.57	97.24 ± 1.30	82.97 ± 4.64	90.98 ± 9/86	62.39 ± 2.88	90.34 ± 3.06	78.37 ± 2.29
	Time	65.16	132.45	133.40	56.00	134.56	123.89	1871.65	134.66
	No. of Kernels	2	2	2	2	3	2	4	4
FMKL P_OVLP	**Accuracy**	84.00 ± 2.90	93.95 ± 2.02	98.37 ± 0.95	87.36 ± 3.70	90.39 ± 3.24	76.55 ± 1.61	92.79 ± 2.47	79.58 ± 2.79
	F-Measure	82.35 ± 3.22	95.22 ± 1.62	98.72 ± 0.73	85.84 ± 3.97	93.62 ± 2.25	62.78 ± 2.44	91.74 ± 2.80	82.34 ± 2.68
	Time	84.65	140.56	65.52	37.33	78.47	90.07	1204.24	105.57
	No. of Kernels	3	3	2	2	3	2	3	2
FMKL OVLP	**Accuracy**	84.69 ± 2.99	94.28 ± 2.33	97.57 ± 0.98	85.98 ± 4.42	89.41 ± 4.29	76.95 ± 2.23	93.06 ± 2.02	79.56 ± 2.85
	F-Measure	82.05 ± 3.68	95.56 ± 1.82	97.99 ± 0.79	84.15 ± 5.11	92.20 ± 2.70	62.27 ± 3.39	92.13 ± 2.34	82.23 ± 2.88
	Time	73.87	73.58	74.69	123.22	57.34	78.49	1314.24	92.93
	No. of Kernels	2	2	2	2	2	2	2	3
FMKL D_OVLP	**Accuracy**	85.14 ± 2.61	94.83 ± 2.57	97.92 ± 0.9	88.26 ± 3.58	90.73 ± 3.72	77.02 ± 2.87	93.62 ± 2.29	79.31 ± 2.46
	F-Measure	82.97 ± 3.82	96.28 ± 1.91	98.12 ± 0.88	86.26 ± 4.75	93.84 ± 2.59	63.17 ± 3.65	92.68 ± 2.60	82.82 ± 2.91
	Time	62.32	57.52	48.03	105.62	45.65	61.58	362.31	82.11
	No. of Kernels	2	3	3	4	2	2	3	4

3.3.2.1 Performance Analysis

From the results in Table 3.3 we can see that FMKL D_OVLP model which used the proposed clustering algorithm performed better in comparison with other models which used spectral, k-means and normal MKL formulation. The performance of FMKL models were as good as existing models with datasets such as ionosphere, pima while the results was promising with datasets such as sonar, parkinsons etc. If different features choose different kernels, then the FMKL models give improved results. But if all the features perform almost similar with same kernel, then performance of FMKL and other MKL formulations are the same.

The Table 3.4 shows the comparative study of performance of the overlapping clustering algorithms, MOC, Fuzzy K-Means [62] and OCC with that of OVLP. All the clustering algorithms used conditional expectation as features with same set of kernels and same MKL formulation. The performance of OVLP was found to be better.

Table 3.4: F-Measure comparison: Overlapping Clustering Algorithms

Dataset/Clustering	MOC	OCC	OVLP	Fuzzy K-Means
Heart	81.48 ± 4.72	81.56 ± 4.47	82.35 ± 3.22	83.46 ± 3.49
Ionosphere	94.94 ± 1.78	95.01 ± 1.71	95.22 ± 1.62	94.91 ± 2.19
Wdbc	97.74 ± 0.97	97.89 ± 0.91	98.54 ± 0.79	97.38 ± 0.95
Sonar	83.38 ± 6.31	83.07 ± 6.57	85.84 ± 3.97	83.63 ± 5.29
Parkinsons	91.24 ± 3.62	90.94 ± 3.21	93.62 ± 2.25	92.08 ± 3.88
Pima	60.42 ± 2.83	59.96 ± 2.77	62.78 ± 2.44	59.79 ± 3.48
Musk 2	90.18 ± 3.54	90.59 ± 3.81	91.74 ± 2.80	90.42 ± 3.99
Arrhythmia	80.29 ± 2.94	81.15 ± 3.15	82.34 ± 2.68	80.92 ± 3.10

Using the Sonar data set ROC curve was plotted (Fig. 3.4). The ROC curves obtained for SimpleMKL and SMO SVM were very much identical and hence SMO SVM is not shown in Fig. 3.4. The area under the curve for MKL was 0.8645, FMKL P_OVLP was 0.8813 and FMKL OVLP was 0.8786.

3.3.2.2 Rank Analysis

In the classification experiments, based on the statistical significance measure the models were ranked for their performance on each data. For example: let M_1 and M_2 be two

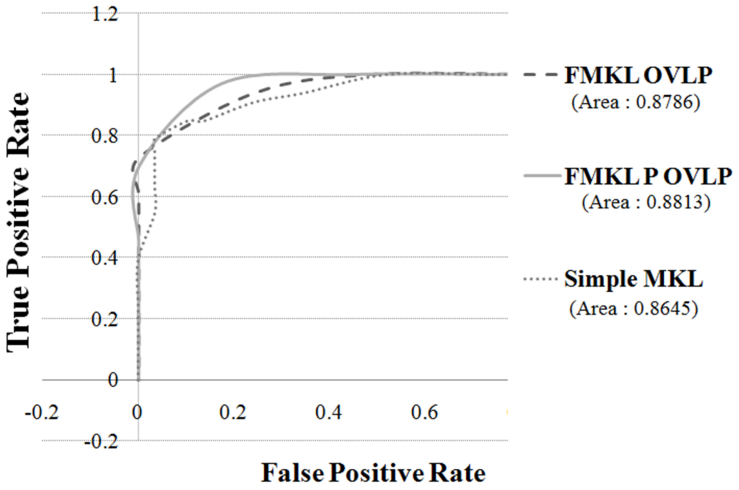

Figure 3.4: ROC-Curve for Sonar Dataset in FMKL Experiments

models; let P_1 and P_2 be the values of a performance measure P for a given data set D. Then we say that M_1 is better than M_2 on the basis of P on D if $P_1 > P_2$ and their difference is statistically significant.

The different models were assigned with ranks based on $F_{measure}$ (Table 3.5). On analyzing the rank table which is based on the performance analysis with different datasets, it can be concluded that the overall performance of the FMKL is better than other MKL formulations.

Table 3.5: Average Rank comparison of FMKL Experiements

Models	Average Rank (F_Measure)
FMKL D_OVLP	1.25
FMKL P_OVLP	2.13
FMKL OVLP	2.5
FMKL K Means	3.88
GL MKL	4.50
SimpleMKL	4.63
FMKL Spectral	4.63
FMKL P_Kmeans	4.63
FMKL Normal	4.88
SMO SVM	5.13

Among all the models, the FMKL D_OVLP gave the best result as shown in Table 3.5. The ranks of different models based on $Accuracy$ are plotted in Fig. 3.5 from which we can see that, FMKL D_OVLP and FMKL P_OVLP models are the top performers.

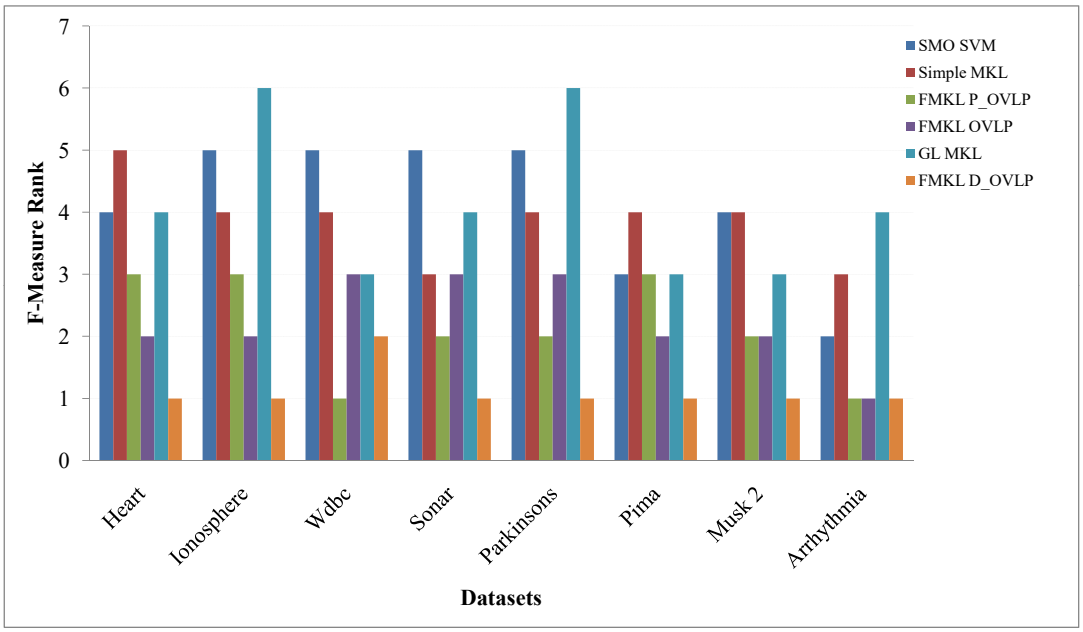

Figure 3.5: Rank Analysis of FMKL Models Based on F-Measure

3.3.2.3 Performance Analysis

The Table 3.6 shows the time analysis of FMKL D_OVLP and SimpleMKL over large datasets in which clustering time is also included for FMKL D_OVLP. The FMKL D_OVLP gave promising results with three clusters, that is, three kernels for Twonorm and Ringnorm data. On the other hand, for these two data sets, SimpleMKL optimization took more time for optimization of parameters as it is associated with ten kernels. The results shows that the FMKL model is trained in shorter time using fewer number of kernels without compromising the accuracy.

Table 3.6: FMKL Execution Time Analysis

Alg.	FMKL D_OVLP		SimpleMKL	
Dataset	Twonorm	Ringnorm	Twonorm	Ringnorm
Accuracy	97.96	98.61	96.79	97.03
No: of clusters	3	3	10	10
Time (seconds)	463.19	442.53	968.32	794.32

3.3.2.4 Impact of kernels chosen by SVM in MKL

In order to understand the impact of the kernels chosen by each MKL model in determining its performance, we analyzed the weights of each model's kernels. In FMKL models, for most of the datasets, the kernel with the highest weight and the kernel which was selected by SVM through cross validation are the same. A few such results are shown in the Table 3.7 where we can see the kernel weights assigned to the SVM chosen kernel in SimpleMKL and FMKL. For sonar dataset, inspite of not selecting the SVM chosen kernel for any of the clusters, the FMKL models performed well.

Table 3.7: Impact of kernel chosen by SVM in MKL vs FMKL

	Kernel Weights	
Dataset	**SimpleMKL**	**FMKL P_OVLP**
Ionosphere	0.31	0.58
Sonar	0	Not Selected
Parkinsons	0	0.68
WDBC	0.13	1

3.3.2.5 Dimensionality reduction

The dimensionality reduction of the data was performed in FMKL models. This could be illustrated by using the results of FMKL P_OVLP on WDBC data. Two clusters were obtained by applying FMKL P_OVLP on WDBC features, of which, the first cluster contained 27 features and second cluster contained 6 features. Among the 6 features of second cluster, 3 were not present in the first cluster. The kernel weight associated with the first cluster that used the same SVM selected kernel is 1 (Table 3.7), which implies that kernel

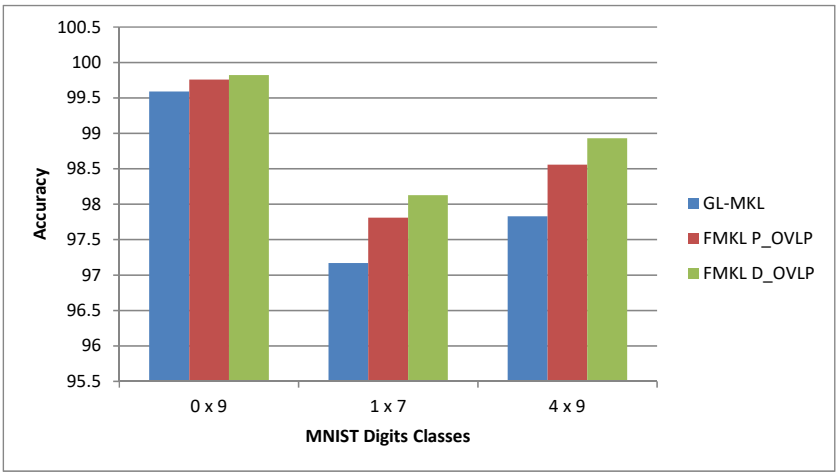

Figure 3.6: Analysis of accuracy between GL-MKL and FMKL P_OVLP over MNIST (0 vs 9) dataset, MNIST (1 vs 7) dataset and MNIST (4 vs 9) dataset

weight of other cluster is zero. The FMKL algorithm showed 98% for accuracy using the features in a single cluster. The SVM gives 96% accuracy by using all the features for WDBC dataset while it showed 98% accuracy by using features selected by FMKL cluster 1. This implies that in FMKL models, the features which are irrelevant in classification are avoided by assigning weight to the kernel associated to those features as zero and hence these models perform feature selection in an implicit manner.

3.3.2.6 Application on Image Dataset

In order to analyze the performance of the proposed model on image data, we chose the problem of recognizing hand written digits of 0 vs 9, 1 vs 7 and 4 vs 9, using the data chosen from MNIST dataset ([63]). The data consisted of 11872 training points and 1989 testing points for 0 vs 9, 13007 training points and 2163 testing points for 1 vs 7 and 11791 training points and 1991 testing points for 4 vs 9. The data features were extracted using techniques such as SIFT[64] and HOG [65]. The data was analysed using GL-MKL[32] and FMKL. The experimental set up for GL-MKL was as same as that of the work in [32]. The results of the analysis are shown in Fig 3.6, which clearly proves the superior performance of FMKL over GL-MKL.

3.4 Conclusion

This chapter deals with the selection of kernels based on features of the data. A theoretical framework for feature wise kernel selection has been developed. The performance of the developed model had been assessed by applying it on real world data sets. A comparative study was conducted to evaluate the performance of the proposed model with that of existing techniques. The results show that the SVM model performs better by applying MKL theory using feature clustering techniques than associating all the features with the same set of kernels. The clustering of features also helps to decrease the computational cost.

Like any other MKL formulations, the FMKL also formulates the kernel learning problem as part of a global classification problem. A separate formulation for the kernel learning is another interesting approach for finding the optimal kernel which is described in the next chapter.

Chapter 4

Classification Approach for Multiple Kernel Learning

The function approximation approach is an efficient method for multiple kernel learning. For that classification as well as regression techniques can be used. A notable work in this direction is that of the work of Kumar et al.[1], where the problem of finding the optimum kernel is modeled as a classification problem, in which the authors used two separate cost functions for finding the parameters of the function that corresponds to the optimum kernel and to the one that has to be learned from the data.

This chapter describes the framework we designed for function approximation approach for multiple kernel learning. In the case of binary classification problems, we formulated the problem of finding the unknown parameters of the optimum kernel and the function that has to be learned from the data (binary classifier) as a single classification task.

The rest of the chapter is organized is as follows: the next section describes the function approximation frame work formulated by Kumar et. al. Such a formulation increases the computational space requirements and hence we incorporate a data compression technique in our formulation whose description is given in section 4.2. The section 4.3 gives the design details of the framework we developed for function approximation in the case of classification problems; details of experimental analysis can be seen in section 4.4 and finally conclusion is given in section 4.5.

4.1 Function Approximation Approach for MKL

We have seen earlier that in the case of linear combination of kernels, the optimal kernel $\hat{k}(.,.) = \sum_{l=1}^{P} d_l k_l(.,.)$ where $\forall\ d_l \geq 0$ and k_l are the base kernels. Therefore, $f(x) = \sum_{i=1}^{N} \alpha_i \sum_{l=1}^{P} d_l k_l(x_i, x)$.

[1] formulated the problem of learning the parameters $\{d_l\}_{l=1}^{P}$ as a classification problem, whose description is given in this section.

Define a function $f^* : \mathcal{X}^* \subset \mathbb{R}^P \rightarrow \mathbb{R}$ such that

$$f^*(z) = d^T z \tag{4.1}$$

where $\mathcal{X}^* = Range(k_1(.,.)) \times Range(k_2(.,.)) \times ... \times Range(k_P(.,.))$ and $d = \{d_1, d_2, \ldots d_P\}^T \in \mathbb{R}^P$ is as given above. From (4.1) it is clear that f^* is a hyperplane defined on $\mathcal{X}^*$.

Therefore

$$f(x) = \sum_{i=1}^{N} \alpha_i f^*(\tilde{K}(x, x_i)) \tag{4.2}$$

where $\tilde{K}(x, x_i) = [k_1(x, x_i) \quad k_2(x, x_i) \quad ... \quad k_p(x, x_i)]^T$. Thus the computation of f involves the computation of yet another function f^*.

The problem of finding f^* can be considered as a supervised classification problem, by adopting the following strategy for generating the outputs of f^*. The ideal kernel for a classification problem is $\hat{k}(x_i, x_j) = y(i) * y(j)$ [22]. As the objective of MKL algorithms is to find the best possible kernel, it could be assumed that

$$f^*(\tilde{K}(x_i, x_j)) = \sum_{l=1}^{P} d_l k_l(x_i, x_j) = y_i y_j, i, j \in \{1, 2, \ldots N\}$$

.

Now f^* can be found using the N^2 data points $\left\{ \left(\tilde{K}(x_i, x_j), y_i y_j \right), i, j = 1, 2, \ldots N \right\}$.

As $O(N^2)$ training points are required to learn f^*, the algorithm complexity increases with increase of data. Hence we resort to a data compression technique for reducing the cardinality of the training set of f^*.

4.2 Data Compression

In order to compress the data, we made use of the empirical mean map of each class in RKHS space. The empirical mean map of positive class (f_+^j) and that of negative class

(f_-^j) corresponding to the base kernel k_j are defined as follows:

$$\begin{aligned} f_+^j &= \frac{1}{N^+} \sum_{x_i \in \{x_i, y_i = +1\}} k_{x_i}^j \\ f_-^j &= \frac{1}{N^-} \sum_{x_i \in \{x_i, y_i = -1\}} k_{x_i}^j \end{aligned} \tag{4.3}$$

where N^+ is number of positive data points, N^- is number of negative data points and $k_{x_i}^j \in \mathcal{F}$ is the representer of evaluation at $x_i \in \mathcal{X}$ for j^{th} base kernel. This empirical mean can be considered as a representative point of the respective class in corresponding RKHS. So in order to reduce the cardinality of data set for f^*, we defined the new training data set as follows

$$[z_1, z_2, \ldots z_{2N}]^T = \left[\tilde{K}_{f_+}(x_1), \tilde{K}_{f_+}(x_2), \ldots, \tilde{K}_{f_+}(x_N), \tilde{K}_{f_-}(x_1), \ldots, \tilde{K}_{f_-}(x_N)\right]^T \tag{4.4}$$

where $\tilde{K}_{f_+}(x) = \left[< k_x^1, f_+^1 >, < k_x^2, f_+^2 >, \ldots, < k_x^P, f_+^P >\right]^T$ and $\tilde{K}_{f_-}(x) = \left[< k_x^1, f_-^1 >, < k_x^2, f_-^2 >, \ldots, < k_x^P, f_-^P >\right]^T$.

The label corresponding to the data points which are related with positive mean is:

$$\begin{aligned} f^*(z_j) =& f^* \left(\begin{bmatrix} < k_{x_j}^1, \frac{1}{N^+} \sum_{x_i \in \{x_i, y_i = +1\}} k_{x_i}^1 > \\ < k_{x_j}^2, \frac{1}{N^+} \sum_{x_i \in \{x_i, y_i = +1\}} k_{x_i}^2 > \\ \cdot \\ \cdot \\ \cdot \\ < k_{x_j}^P, \frac{1}{N^+} \sum_{x_i \in \{x_i, y_i = +1\}} k_{x_i}^P > \end{bmatrix} \right) \\ =& \frac{1}{N^+} \sum_{x_i \in \{x_i, y_i = +1\}} f^* \left(\begin{bmatrix} < k_{x_j}^1, k_{x_i}^1 > \\ < k_{x_j}^2, k_{x_i}^2 > \\ \cdot \\ \cdot \\ \cdot \\ < k_{x_j}^P, k_{x_i}^P > \end{bmatrix} \right) \end{aligned}$$

This is possible, since f^* is linear.

By making use of the concepts of ideal kernel and the above formulation, the label of

z_j becomes

$$\begin{aligned} y_j^* =& \frac{1}{N^+} \sum_{x_i \in \{x_i, y_i = +1\}} y_j . y_i \\ =& \frac{1}{N^+} y_j . \sum_{x_i \in \{x_i, y_i = +1\}} +1 = \frac{y_j N^+}{N^+} = y_j \end{aligned}$$

Similarly, the labels corresponding to negative mean map can be determined.

Thus the labels for the data set in (4.4) are

$$[y_1^*, y_2^*, \ldots y_{2N}^*]^T = [y_1, y_2, \ldots y_N, -y_1, \ldots, -y_N] \tag{4.5}$$

Thus we have $2N$ training points for learning f^*.

4.3 Computation of f and f^* using Single Stage Frame Work

The authors of [1] found f^* and f using separate cost functions by finding f^* in the first stage and then substituting in (4.2) for finding f. We solved (4.2) by finding f and f^* using a single cost function, whose description is given below.We formulated the problem of finding f^* as a classification task as given in the previous section. Let f^* belongs to RKHS $\mathcal{F}^*$, whose reproducing kernel is k^*. For finding f and f^* using a single cost function, we construct a RKHS space in which f and f^* are members, whose formulation is as given below.

4.3.1 Construction of Global RKHS

Let $\mathcal{X}^\dagger = \mathcal{X} \times \mathcal{X}^*$. Define $k^\dagger : \mathcal{X}^\dagger \times \mathcal{X}^\dagger \to \mathbb{R}$, as follows

$$\begin{aligned} & k^\dagger(x_i, x_j) \\ =& \begin{cases} k(\phi(x_i), \phi(x_j)) & ; \text{if } \phi^*(x_i) = \vec{0} \ \& \ \phi^*(x_j) = \vec{0} \\ k^*(\phi^*(x_i), \phi^*(x_j)) & ; \text{if } \phi(x_i) = \vec{0} \ \& \ \phi(x_j) = \vec{0} \\ 0 & ; otherwise \end{cases} \end{aligned} \tag{4.6}$$

where $\phi : \mathcal{X}^\dagger \to \mathcal{X}$ and $\phi^* : \mathcal{X}^\dagger \to \mathcal{X}^*$ are projection operators.

As k and k^* are valid reproducing kernels, $k^\dagger$ is a valid reproducing kernel. Let $\mathcal{F}^\dagger$ be the RKHS corresponding to $k^\dagger$.

It is clear from (4.6) that $\mathcal{F}^\dagger/\mathcal{X} \equiv \mathcal{F}$ and $\mathcal{F}^\dagger/\mathcal{X}^* \equiv \mathcal{F}^*$. Also, $\mathcal{F}$ and $\mathcal{F}^*$ are orthogonal subspaces of $\mathcal{F}^\dagger$ and complements to each other.

Hence

$$\mathcal{F}^\dagger = \mathcal{F} \oplus \mathcal{F}^* \tag{4.7}$$

So every $f^\dagger \in \mathcal{F}^\dagger$ can be represented as

$$f^\dagger = f + f^*, f \in \mathcal{F}, f^* \in \mathcal{F}^* \tag{4.8}$$

Using (4.7) and (4.8)

$$||f^\dagger||^2 = ||f||^2 + ||f^*||^2 \tag{4.9}$$

Thus f and f^* can be found by finding $f^\dagger \in \mathcal{F}^\dagger$.

$f^\dagger$ is determined using the $N + N^2$ points from $\mathcal{X}^\dagger$, where N points are associated with f and N^2 with that of f^*. That is, the data points for $f^\dagger$ are $\{(x_i^\dagger, y_i)\}_{i=1}^N \cup \{(z_i^\dagger, y_i^*)\}_{i=1}^{2N}$, where $x_i^\dagger = x_i \times \vec{0}, x_i \in \mathcal{X}, \vec{0} \in \mathcal{X}^*$ and $z_i^\dagger = \vec{0} \times z_i, \vec{0} \in \mathcal{X}, z_i \in \mathcal{X}^*$.

4.3.2 Formulation using SVM

By virtue of semi-parametric form of representer theorem [66], the decision boundary corresponding to data points $\left[\{(x_i, y_i)_{i=1}^{3N}\} = \{(x_i^\dagger, y_i)\}_{i=1}^N, \{(z_i^\dagger, y_i^*)\}_{i=1}^{2N}\right]$ can be represented as

$$g^\dagger(x) = f^\dagger(x) + b = \sum_{i=1}^{3N} \alpha_i k^\dagger(x_i, x) + b, \alpha_i, b \in \mathbb{R}, i = 1, 2, \ldots 3N \tag{4.10}$$

We used SVM, for finding $f^\dagger$ and b. Hence the optimization problem corresponding to (4.10) is

$$\begin{aligned} &\min_{f^\dagger} \frac{1}{2}\|f^\dagger\|^2 + C\sum_{i=1}^{3N} \xi_i^\dagger \\ &subj.\ to \\ &\qquad y_i^\dagger[\langle f^\dagger, k^\dagger{}_{x_i^\dagger}\rangle + b] - 1 + \xi_i^\dagger \geq 0 \forall i = 1 \ldots 3N \\ &\qquad \xi_i^\dagger \geq 0 \forall i = 1 \ldots 3N \end{aligned} \tag{4.11}$$

Since $\mathcal{F}$ and $\mathcal{F}^*$ are orthogonal, (4.11) can be written as

$$\min_{f,f^*} \frac{1}{2}\|f\|^2 + C\sum_{i=1}^{N}\xi_i + \frac{1}{2}\|f^*\|^2 + C\sum_{j=1}^{2N}\xi_j^*$$

$$\begin{aligned} &subj.\ to \\ &\quad y_i[\langle f, k_{x_i}\rangle + b] - 1 + \xi_i \geq 0, \ \ \forall i = 1\ldots N \\ &\quad y_j^*[\langle f^*, k_{z_j}^*\rangle + b] - 1 + \xi_j^* \geq 0, \ \ \forall j = 1\ldots 2N \\ &\quad \xi_i \geq 0, \ \ \forall i = 1\ldots N \\ &\quad \xi_j^* \geq 0, \ \ \forall j = 1\ldots 2N \end{aligned} \tag{4.12}$$

Corresponding Lagrangian objective function is as follows

$$\begin{aligned} \mathcal{L}(f, f^*, \xi, \xi^*) =& \frac{1}{2}\|f\|^2 + \frac{1}{2}\|f^*\|^2 + C\sum_{i=1}^{N}\xi_i + C\sum_{j=1}^{2N}\xi_j^* \\ &- \sum_{i=1}^{N}\alpha_i[y_i(\langle f, \hat{k}_{x_i}\rangle + b) - 1 + \xi_i] \\ &- \sum_{j=1}^{2N}\beta_j[y_j^*[\langle f^*, k_{z_j}^*\rangle + b] - 1 + \xi_j^*] \\ &- \sum_{i=1}^{N}\mu_i\xi_i - \sum_{j=1}^{2N}\mu_j^*\xi_j^* \end{aligned} \tag{4.13}$$

Taking partial derivative

$$\frac{\partial \mathcal{L}}{\partial f} = f - \sum_{i=1}^{N}\alpha_i y_i \hat{k}_{x_i} = 0 \tag{4.14}$$

Hence

$$f = \sum_{i=1}^{N}\alpha_i y_i \hat{k}_{x_i} \tag{4.15}$$

Now taking partial for f^* gives

$$\frac{\partial \mathcal{L}}{\partial f^*} = f^* - \sum_{j=1}^{2N}\beta_j y_j^* k_{z_j}^* = 0 \tag{4.16}$$

Therefore from equation (4.16) we can write,

$$f^* = \sum_{j=1}^{2N} \beta_j y_j^* k_{z_j}^* \tag{4.17}$$

Other results are shown below:

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial \xi_i} = \alpha_i - \mu_i = 0, i = 1, 2, \ldots N \\
&\frac{\partial \mathcal{L}}{\partial \xi_j^*} \Rightarrow C - \beta_j - \mu_j^* = 0, j = 1, 2, \ldots 2N \\
&\frac{\partial \mathcal{L}}{\partial b} \Rightarrow \sum_{i=1}^{N} \alpha_i y_i + \sum_{j=1}^{2N} \beta_j y_j^* = 0
\end{aligned}$$

So the dual function can be written as

$$\begin{aligned}
W(\alpha, \beta) = &\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \sum_{k=1}^{2N} \beta_k y_k^* k^*(z_k^*, \tilde{K}(x_i, x_j)) \\
&+ \sum_{j=1}^{2N} \beta_j - \frac{1}{2} \sum_{k=1}^{2N} \sum_{l=1}^{2N} \beta_k \beta_l y_k^* y_l^* k^*(z_k^*, z_l^*) \\
sub.\ to& \\
&0 \leq \alpha_i \leq C \\
&0 \leq \beta_j \leq C \\
&\sum_{i=1}^{N} \alpha_i y_i + \sum_{j=1}^{2N} \beta_j y_j^* = 0
\end{aligned} \tag{4.18}$$

4.3.2.1 SMO based optimization

The dual function defined in (4.18) can be optimized using two step optimization algorithm.

4.3.2.1.1 Step 1 In step one α is optimized keeping β as a constant. The cost function for first step is as follows (where all β terms are kept as constant)

$$\begin{aligned} W_\beta(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \sum_{k=1}^{2N} \beta_k y_k^* k^*(z_k^*, \tilde{K}(x_i, x_j)) + \gamma_\beta \\ sub.\ to \\ 0 \leq \alpha_i \leq C \\ \sum_{i=1}^{N} \alpha_i y_i = \gamma'_\beta \end{aligned} \tag{4.19}$$

where γ_β and γ'_β are constants. This is a usual SVM formulation which could be solved using any of the SVM solver. Here we use SMO for solving the SVM.

4.3.2.1.2 Step 2 In second step of algorithm α is fixed as a constant and β is optimized. Therefore the cost function can be written as follows

$$\begin{aligned} W_\alpha(\beta) = \sum_{j=1}^{2N} \beta_j - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \sum_{k=1}^{2N} \beta_k y_k^* k^*(z_k^*, \tilde{K}(x_i, x_j)) \\ - \frac{1}{2} \sum_{k=1}^{2N} \sum_{l=1}^{2N} \beta_k \beta_l y_k^* y_l^* k^*(z_k^*, z_l^*) + \gamma_\alpha \\ sub.\ to \\ 0 \leq \beta_j \leq C \\ \sum_{j=1}^{2N} \beta_j y_j = \gamma'_\alpha \end{aligned} \tag{4.20}$$

where γ_α and γ'_α are constants.

The procedure for solving $W_\alpha(\beta)$ using SMO is given below.

Consider

$$\begin{aligned}\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \sum_{k=1}^{2N}\beta_k y_k^* k^*(z_k^*, \tilde{K}(x_i, x_j)) \\ =&\frac{1}{2}\sum_{k=1}^{2N}\beta_k y_k^* \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k^*(z_k^*, \tilde{K}(x_i, x_j)) \\ =&\frac{1}{2}\sum_{k=1}^{2N}\beta_k y_k^* u_k\end{aligned} \tag{4.21}$$

where $u_k = \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k^*(z_k^*, \tilde{K}(x_i, x_j))$.

Substituting (4.21) into (4.20)

$$\begin{aligned}W_\alpha(\beta) = &\sum_{j=1}^{2N}\beta_j - \frac{1}{2}\sum_{k=1}^{2N}\sum_{l=1}^{2N}\beta_k\beta_l y_k^* y_l^* k^*(z_k^*, z_l^*) \\ &- \frac{1}{2}\sum_{k=1}^{2N}\beta_k y_k^* u_k + \gamma_\alpha \\ sub.\ &to \\ &0 \le \beta_j \le C \\ &\sum_{j=1}^{2N}\beta_j y_j = \gamma'_\alpha\end{aligned} \tag{4.22}$$

This equation is again a quadratic equation with box constraints. Let β_1 and β_2 be the parameters that SMO is updating at each step. The upper bound and lower bound of the parameters are same as that of in SMO derivation. However the expression to find $\beta' s$ will be different since additional terms are included in the cost function.

The β_1^{new} and β_2^{new} are obtained as follows.

$$\begin{aligned}\beta_2^{new} =&\beta_2^{old} + \frac{y_2^*(\mathcal{E}_1^* - \mathcal{E}_2^* + \frac{1}{2}(u_1 - u_2))}{\eta} \\ \beta_1^{new} =&\beta_1^{old} + y_1^* y_2^*(\beta_2^{old} - \beta_2^{new})\end{aligned} \tag{4.23}$$

where $\mathcal{E}_1^* = (f^*(z_1) - y_1^*)$, $\mathcal{E}_2^* = (f^*(z_2) - y_2^*)$ and $\eta = k^*(z_1, z_1) + k^*(z_2, z_2) - 2k^*(z_2.z_1)$.

The update process is continued until convergence. Change in cost function during each iteration is used for determining convergence in such a way that if the change is less than a predefined $\epsilon > 0$ then the iteration stops. The algorithm is shown in Alg. 2

Algorithm 2 Function Approximation Algorithm

```
1: procedure FUNCTION-MKL(datapoints)
2:     Calculate K̃ using mean maps
3:     Learn initial f* using SMO-SVM;
4:     while until convergence do
5:         Learn f using SMO-SVM
6:         Update α
7:         Learn f* using Modified SMO-SVM
8:         Update β
9:     end while
10: end procedure
```

4.3.2.2 Time Complexity Analysis

The FTSMKL (Kumar et. al. 2012) is having two stages. The first stage formulates the kernel target alignment as function approximation and it uses PEGASOS algorithm which has a complexity of $O(d/\lambda\epsilon)$, where d is the dimension of the data, $\lambda > 0$ is the regularization parameter and ϵ is the expected solution accuracy. The second stage comprises the SMO SVM algorithm which is having the complexity of $O(N^{2.2})$. Since the first stage depends linearly on dimension d and SMO depends quadratically on the number of training points N, the complexity of FTSMKL is $O(N^{2.2})$.

In algorithm 2, the time complexity of mean map based kernel computation for $3N$ data points is $O(N^3 d)$. The steps 3, 5 and 7 use SMO-SVM, whose complexity is $O(N^{2.2})$. The number of iterations the algorithm takes for convergence is less than dimension d. Therefore the overall time complexity of the algorithm is $O(N^3 d)$ + $O(N^{2.2} d) \simeq O(N^3 d)$.

The data compression method, which uses mean map, reduces the number of training points for f^* from N^2 to $2N$. Therefore without data compression the overall complexity for the kernel computation is $O((N^2)^3 d) \simeq O(N^6 d)$. The data compression technique, in effect, reduces the space complexity as well as the time complexity.

4.3.3 Online Learning MKL

The work is extended to online learning and for that the technique used in kernel online learning algorithm developed by Kivinen et al [67] has been adopted.

Let (x_{t+1}, y_{t+1}) be the data arrived at time $(t+1)$. On the arrival of a new data point x_{t+1} at $(t+1)^{th}$ iteration, a training point is created for updating f^* as (z_{t+1}, y^*_{t+1}) where $z_{t+1} = \tilde{K}(x_t, x_{t+1})$ and $y^*_{t+1} = y_t y_{t+1}$. Hence at time (t+1) the points under consideration are $\Big[\{(x^\dagger_{t+1}, y_{t+1})\}\{(z^\dagger_{t+1}, y^*_{t+1})\}\Big]$. The optimal function is found by minimizing

$$g(f, f^*) = \min_{f^\dagger \in \mathcal{F}^\dagger} \frac{\lambda}{2}\|f^\dagger\|^2 + l(.) \tag{4.24}$$

where $l(.)$ is the hinge loss function.

Since $f^\dagger = f + f^*$,

$$g(f, f^*) = \min_{f \in \mathcal{F}, f^* \in \mathcal{F}^*} \frac{\lambda}{2}\|f\|^2 + \frac{\lambda}{2}\|f^*\|^2 + l(.) \tag{4.25}$$

By applying stochastic gradient,

$$f^* = f^* - \eta^* \frac{\partial g}{\partial f^*} \tag{4.26}$$

and

$$f = f - \eta \frac{\partial g}{\partial f} \tag{4.27}$$

where $\eta > 0$ and $\eta^* > 0$ are the learning rates.

Expanding (4.26), the coefficients of f^* at each step are updated as follows

$$\begin{aligned}
\beta_i^{new} &= \begin{cases} [l]\beta_i^{old}(1 - \eta^*\lambda) & ; \quad 1 \le i \le t \\ \eta^* y_i^* & ; \quad \text{if } (y^*_{t+1} f^\dagger(z_i) < 1) \ \& \ i = t+1 \\ 0 & ; \quad \text{if } (y^*_{t+1} f^\dagger(z_i) \ge 1) \ \& \ i = t+1 \end{cases} \\
b^{new} &= \begin{cases} [l]b^{old} + \eta^* y_i^* & ; \quad \text{if } (y^*_{t+1} f^\dagger(z_{t+1}) < 1) \\ b^{old} & ; \quad \text{if } (y^*_{t+1} f^\dagger(z_{t+1}) \ge 1) \end{cases}
\end{aligned} \tag{4.28}$$

On expanding (4.27), the coefficients of f at each step are updated as follows

$$\begin{aligned}
\alpha_i^{new} &= \begin{cases} [l]\alpha_i^{old}(1 - \eta\lambda) & ; \quad 1 \le i \le t \\ \eta y_i & ; \quad \text{if } (y_{t+1} f^\dagger(x_i) < 1) \ \& \ i = t+1 \\ 0 & ; \quad \text{if } (y_{t+1} f^\dagger(x_i) \ge 1) \ \& \ i = t+1 \end{cases} \\
b^{new} &= \begin{cases} [l]b^{old} + \eta y_i & ; \quad \text{if } (y_{t+1} f^\dagger(x_{t+1}) < 1) \\ b^{old} & ; \quad \text{if } (y_{t+1} f^\dagger(x_{t+1}) \ge 1) \end{cases}
\end{aligned} \tag{4.29}$$

The pseudo-code of the algorithm for the online function approximation multiple kernel learning is given in Algorithm 3. It is a two step algorithm where when each data point arrives, the first step updates for f^* and second step updates for f.

Algorithm 3 Function Approximation Algorithm - Online Algorithm

```
1: procedure ONLINE-FUNCTION-MKL(datapoints)
2:     while new data point arrives do
3:         Generate the new K̃ using the new data point
4:         Update β using the equation (4.28)
5:         Update α using the equation (4.29)
6:     end while
7: end procedure
```

4.4 Experiments

Table 4.1: Classification Datasets for FSSMKL

Dataset	Repository	Dim.	Data Points
Arrythmia	UCI	276	452
Blood Transfusion	UCI	4	748
Haberman	UCI	3	306
Heart	UCI	13	303
Ionosphere	UCI	33	351
Liver	UCI	6	345
Musk 2	UCI	166	476
Parkinsons	UCI	22	195
Pima	UCI	8	768
Sonar	UCI	60	208
Vert. Column	UCI	6	310
WDBC	UCI	30	569
Whole. Cust.	UCI	7	440
Twonorm	IDA	20	7400
Ringnorm	IDA	20	7400

4.4.1 Experimental Setup

The performance of the algorithms was assessed experimentally. The MKL model used in the analysis consisted of 42 kernels (19 Laplacian kernels, 19 Guassian kernels and 4 polynomial kernels) whose description is given below:

- Laplacian Kernel, $k(x,z) = \exp\left(-\frac{\|x-z\|}{\sigma}\right)$, where $\sigma \in \{2^{-9}, 2^{-8}, ..., 2^{9}\}$.
- Gaussian Kernel, $k(x,z) = \exp\left(-\frac{\|x-z\|^2}{\sigma}\right)$, where $\sigma \in \{2^{-9}, 2^{-8}, ..., 2^{9}\}$.
- Polynomial Kernel, $k(x,z) = (\alpha x^T z + c)^d$, where $\alpha = 1, c = 0$ and $d \in \{1, 2, 3, 4\}$.

As f^* is a hyper plane, k^* is chosen to be linear kernel.

The synthetic as well as real time data sets are used for experiments. The synthetic data was generated using

$$f(x) = \sum_{i=1}^{100} \alpha_i \sum_{l=1}^{10} d_l k_l(x_i, x) + \rho \tag{4.30}$$

where $x_i \in [0,1] \times [0,1]$, $d_i \in [0,1], \alpha_i \in [-10.0, 10.0], i = 1, 2 \ldots 100$, $\rho \sim N(0,1)$ is the noise and $k_l, l = 1, 2, \ldots 10$ are 10 randomly selected kernels. The analysis was done using 200 randomly generated training points.

The real time datasets used are listed in Table 4.1, which were taken from UCI Repository[60] and IDA benchmark repository [61].

The model developed that uses function approximation concepts and single stage learning is named as 'Function approximation using single stage MKL (FSSMKL)'. The performance of FSSMKL is compared with the following models:

1 **NonMKL [68]** A Non Linear MKL proposed for Hyperspectral Classification which is also adapted for binary classification.

2 **SimpleMKL[19] :** Linear Combination of kernel approach is used in this model.

3 **FTSMKL[1] :** The model uses function approximation concepts and two stage learning for finding f and f^*, which we named as 'Function approximation using two stage MKL (FTSMKL)'

4 **GMKL[21] :** A regularized formulation of MKL termed as generalized multiple kernel learning algorithm.

5 **LpMKL [69]** An Lp Norm regularized MKL Algorithm.

6 **DMKL [70]** A Discriminative MKL proposed for Hyperspectral Classification which is adapted to binary classification.

Online version of the proposed model is named as OFSSMKL. For comparative study with OFSSMKL we used the following online models.

1 **OKC [67]** Online kernel based algorithm for classification developed by Kivinen et al, that makes use of gradient descent optimization.

2 **OMKC [71]** Online kernel classification where kernel weights and function parameters are updated only during miss classification iteration.

All the models used SVM as the classifier. Except GMKL, LpMKL, OKC and OMKC, the codes of the other models used for comparison were available in jKernelMachine framework [59]. For GMKL and OMKC, we used the matlab code available in authors urls [72] and [73]. For LpMKL the code from LibMKL [74] library is used which is an extension over LibSVM. The online algorithm OKC is implemented over the jKernelMachine library. OFSSMKL is also implemented over jKernelMachine in which maximum parallel execution was provided for much faster updation of function in each iteration. The partial sum technique of parallel algorithms was used for parallel implementation of the online algorithm. We customized all the codes to use same kernels and same procedures in approximating the hyperparameters for a fair comparison of performance.

30-time holdout technique is used for validating the models whose description is as follows. In each iteration, the given data was divided into two parts: training data & validation data. The training data was normalized using max-min normalization and then normalized the validation data using the maximum and minimum value calculated from the training data. The normalized training data was used for building the respective model. The resulting function was used for predicting the results of validation data. This process was repeated for 30 times and the average F-measure & accuracy was calculated for assessing the performance of the model.

The t-test was performed over the 30 times hold out results for verifying the statistical significance of the results (significance level $\alpha = 0.1$). Based on the statistical significance measure, the models were ranked for their performance on each data as described in Chapter 3.

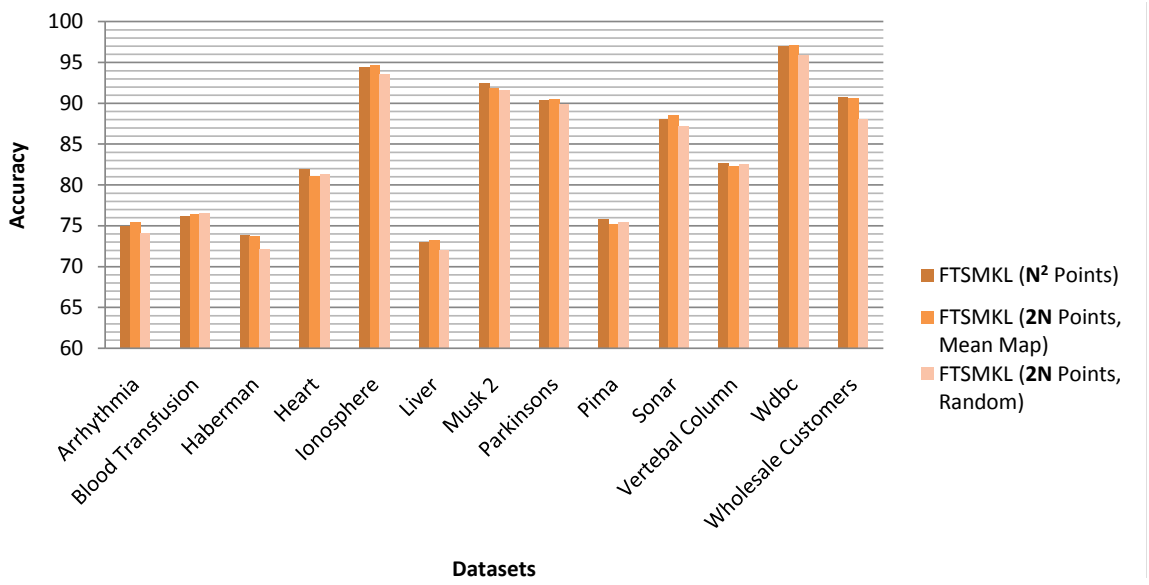

Figure 4.1: Accuracy Graph: FTSMKL Models

4.4.2 Results and Discussions

For understanding the efficiency of data compression techniques used , the performance of FTSMKL is analyzed (a) N^2 (b) $2N$ points that resulted from using empirical means and (c) $2N$ points by selecting a random representor of evaluation from each class in RKHS instead of empirical means. The results of this analysis are given in Fig. 4.1. To check the effectiveness of data compression in online setup we have devised three models. They are

Model 1 : When a new point comes, we construct $\tilde{K}$ using pairs of each existing training points and the newly arrived point itself. These T points are used for updating the f^*

Model 2 : Actual OFSSMKL

Model 3 : When a new point comes, we construct $\tilde{K}$ using pair of mean maps and the newly arrived point itself. The mean maps are constructed using only the arrived points. These 2 points are used for updating the f^*.

The performance comparison of these three models is given in the Fig. 4.2. From results in both Fig. 4.1 and Fig. 4.2, it is clear that the data reduction using empirical means does not deteriorate the performance of the model. Also, random evaluators instead of empirical means give inconsistent results which shows the relevance of using empirical

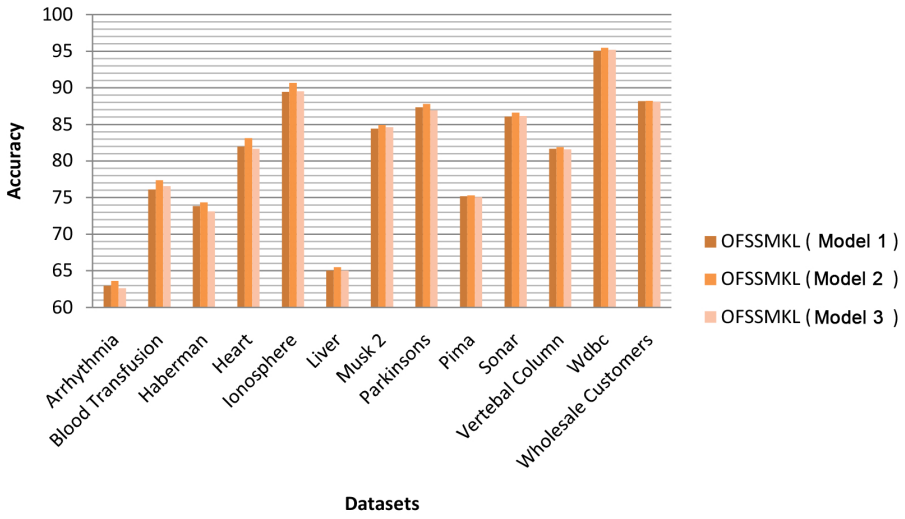

Figure 4.2: Data Reduction using Empirical Means: OFSSMKL Models

means. Hence we used empirical mean data compression for FTSMKL and FSSMKL for all the analysis described below.

The synthetic data analysis results are shown in Fig. 4.3. Fig. 4.3(a) shows the 200 training points generated using equation (4.30) in which positive points are shown as red cross and negative points are shown as blue circle. The true function from which the data generated is shown in Fig. 4.3(b). The decision boundaries generated by SVM-SMO, SimpleMKL, FTSMKL and the proposed model FSSMKL are shown respectively in Fig. 4.3(c), Fig. 4.3(d), Fig. 4.3(e) and Fig. 4.3(f). From the figures, it is clear that of all the four models, the one that generated by FSSMKL closely resembles the true function.

The accuracies of the models on real world datasets are shown in Table 4.2. The rank is displayed in open brackets in each cell of Table 4.2 and 4.3. Finally we computed the average rank for all the models. The FSSMKL scored the highest rank in the case of accuracy as well as F measure.

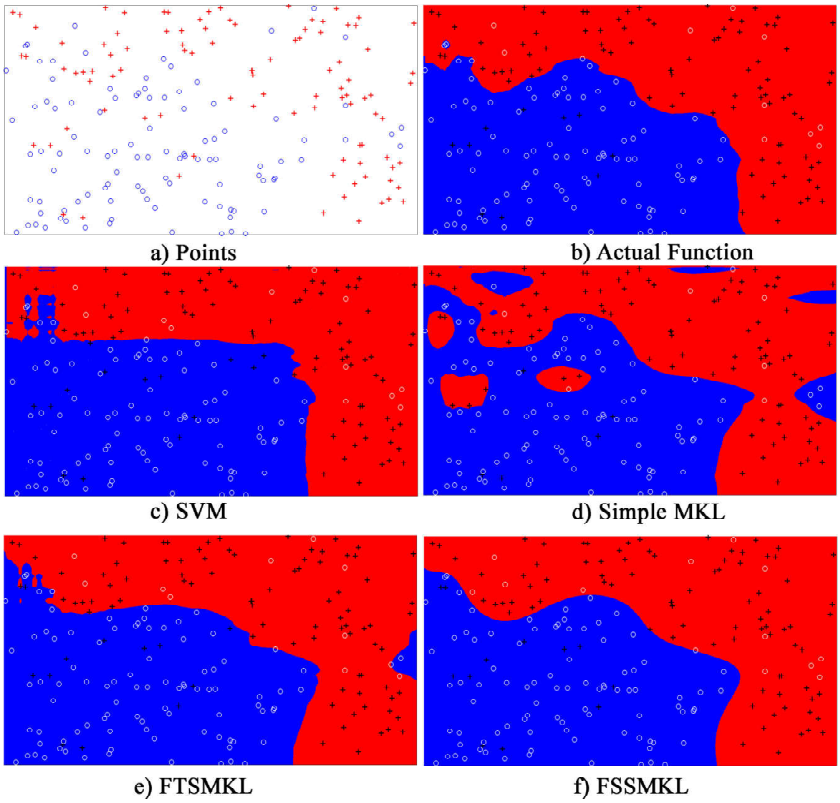

Figure 4.3: Decision Boundary for the Synthetic Data

Table 4.2: Accuracy: Classification based MKL models and other state-of -the art MKL models

Dataset/Models	NonMKL	Simple MKL	FTSMKL	GMKL	LpMKL	DMKL	FSSMKL
Arrhythmia	76.92 ± 3.97 (2)	75.89 ± 3.19 (3)	75.40 ± 3.26 (3)	76.42 ± 3.17 (2)	76.81 ± 2.60 (2)	76.67 ± 3.81 (2)	77.87 ± 2.73 (1)
Blood Transfusion	77.16 ± 2.93 (1)	77.03 ± 2.82 (1)	76.43 ± 2.72 (2)	76.66 ± 2.86 (2)	76.67 ± 1.34 (2)	77.32 ± 2.87 (1)	77.83 ± 2.86 (1)
Haberman	73.19 ± 5.17 (2)	73.31 ± 5.23 (2)	73.74 ± 4.31 (1)	73.75 ± 5.36 (1)	73.21 ± 1.71 (2)	73.29 ± 5.97 (2)	74.13 ± 3.68 (1)
Heart	83.17 ± 3.64 (2)	82.78 ± 2.87 (3)	81.11 ± 5.00 (4)	83.21 ± 3.19 ()2	82.84 ± 3.37 (2)	83.43 ± 3.54 (2)	84.74 ± 3.36 (1)
Ionosphere	93.17 ± 2.09 (2)	93.87 ± 2.20 (1)	94.63 ± 2.24 (1)	93.93 ± 2.31 (1)	94.52 ± 1.69 (1)	93.67 ± 2.34 (1)	94.49 ± 1.95 (1)
Liver	72.02 ± 4.21 (2)	71.87 ± 3.97 (2)	73.22 ± 4.47 (1)	71.25 ± 3.24 (2)	70.66 ± 3.69 (3)	71.87 ± 4.28 (2)	73.17 ± 3.33 (1)
Musk 2	91.86 ± 2.67 (2)	92.51 ± 2.68 (1)	91.82 ± 2.46 (2)	92.34 ± 2.61 (1)	92.10 ± 2.66 (2)	91.79 ± 2.45 (2)	92.89 ± 2.09 (1)
Parkinsons	88.41 ± 4.91 (3)	88.39 ± 4.01 (3)	90.49 ± 5.13 (2)	89.32 ± 4.25 (3)	90.29 ± 2.66 (2)	88.12 ± 4.77 (3)	92.93 ± 3.32 (1)
Pima	76.26 ± 2.34 (1)	76.76 ± 2.86 (1)	75.14 ± 2.68 (2)	74.89 ± 3.63 (2)	76.28 ± 2.39 (1)	76.16 ± 2.70 (1)	76.21 ± 2.42 (1)
Sonar	87.58 ± 4.34 (3)	85.69 ± 4.47 (5)	88.55 ± 4.71 (2)	86.24 ± 4.92 (4)	84.63 ± 4.01 (6)	87.40 ± 4.82 (3)	89.46 ± 4.25 (1)
Vertebal Column	83.04 ± 3.47 (3)	84.13 ± 4.25 (1)	82.32 ± 2.01 (2)	83.06 ± 2.78 (3)	83.23 ± 2.85 (2)	83.16 ± 3.61 (2)	84.62 ± 3.15 (1)
Wdbc	97.14 ± 0.96 (2)	97.06 ± .83 (2)	97.06 ± 1.28 (2)	97.29 ± 1.13 (2)	97.32 ± 1.01 (2)	97.04 ± 0.96 (2)	97.83 ± 0.78 (1)
Wholesale Customers	90.21 ± 2.13 (2)	90.89 ± 2.52 (2)	90.78 ± 2.06 (2)	90.65 ± 2.82 (2)	90.58 ± 1.45 (2)	90.55 ± 1.76 (2)	92.06 ± 1.41 (1)
Avg. Rank	2.07	2.07	2	2.07	2.23	1.92	1

Table 4.3: F-Measure: Classification based MKL models and other state-of -the art MKL models

Dataset/Models	NonMKL	Simple MKL	FTSMKL	GMKL	LpMKL	DMKL	FSSMKL
Arrhythmia	79.24 ± 3.97 (2)	77.91 ± 3.24 (3)	78.27 ±3.26 (3)	77.23 ± 3.42 (3)	77.93 ± 1.89 (3)	79.39 ± 3.26 (2)	81.76 ± 2.38 (1)
Blood Transfusion	13.63 ± 16.28 (4)	21.23 ± 13.88 (3)	26.68 ± 19.03 (3)	15.39 ± 14.17 (4)	39.44 ± 4.06 (1)	24.88 ± 12.56 (3)	34.11 ± 7.65 (2)
Haberman	83.86 ± 4.67 (2)	83.11 ± 5.01 (2)	84.08 ± 2.87 (1)	83.54 ± 4.15 (2)	84.11 ± 1.26 (1)	83.12 ± 4.59 (2)	84.81 ± 2.56 (1)
Heart	81.37 ± 4.64 (2)	81.13 ± 3.45 (2)	80.68 ± 4.68 (2)	80.73 ± 3.57 (2)	80.87 ± 4.12 (2)	81.29 ± 4.70 (2)	82.51 ± 4.37 (1)
Ionosphere	94.73 ± 1.82 (1)	95.26 ± 1.75 (1)	95.72 ± 1.72 (1)	95.38 ± 1.53 (1)	95.04 ± 1.28 (1)	95.18 ± 1.85 (1)	95.08 ± 1.47 (1)
Liver	77.09 ± 4.66 (2)	77.13 ± 4.12 (2)	78.42 ± 3.63 (1)	77.21 ± 4.22 (2)	75.17 ± 3.39 (2)	76.97 ± 4.93 (2)	78.90 ± 3.15 (1)
Musk 2	90.26 ± 3.47 (2)	90.66 ± 3.10 (2)	91.57 ± 2.42 (1)	90.67 ± 3.06 (2)	90.89 ± 3.01 (2)	90.33 ± 3.53 (2)	91.91 ± 2.38 (1)
Parkinsons	92.38 ± 3.65 (2)	91.98 ± 2.65 (2)	92.54 ± 3.01 (2)	91.83 ± 2.58 (2)	91.91 ± 1.76 (2)	92.59 ± 3.54 (2)	95.27 ± 2.41 (1)
Pima	62.43 ± 3.31 (1)	62.61 ± 4.09 (1)	61.05 ± 4.80 (1)	62.29 ± 3.92 (1)	61.38 ± 4.22 (1)	62.37 ± 3.53 (1)	62.69 ± 3.58 (1)
Sonar	83.28 ± 5.72 (4)	83.77 ± 5.36 (4)	86.59 ± 4.67 (2)	85.36 ± 4.62 (3)	83.31 ± 4.31 (4)	83.45 ± 5.54 (4)	88.20 ± 4.75 (1)
Vertebal Column	87.67 ± 3.67 (1)	87.81 ± 3.98 (1)	86.95 ± 3.91 (2)	87.27 ± 2.83 (2)	87.63 ± 2.14 (1)	87.82 ± 3.50 (1)	88.42 ± 2.72 (1)
Wdbc	97.03 ± 0.80 (2)	97.17 ± 0.64 (2)	97.32 ± 1.04 (2)	97.23 ± 0.60 (2)	97.53 ± 0.79 (2)	97.27 ± 0.89 (2)	98.28 ± 0.62 (1)
Wholesale Customers	92.70 ± 1.83 (2)	93.24 ± 1.87 (1)	93.56 ± 2.37 (1)	90.92 ± 1.57 (3)	93.07 ± 1.07 (2)	92.84 ± 1.76 (2)	94.06 ± 1.16 (1)
Avg.Rank	2.07	2	1.69	2.23	1.92	2	1.07

4.4.2.1 Comparison of FSSMKL with FTSMKL

Both FTSMKL and the proposed method FSSMKL use function approximation technique for MKL learning. Such a formulation has to learn two functions from the data: f, the classifier related with data and f^*, the classifier related with MKL learning. The main difference between FSSMKL and FTSMKL is that, FSSMKL uses a single cost function for finding the classifiers associated with data and MKL learning while FTSMKL uses two separate cost functions for finding these functions.

The experimental analysis ascertained the generalization capacity of the FSSMKL as for all the data sets we used either it showed superior performance or comparable performance with other state of art techniques. On the other hand the performance of FTSMKL deteriorates in the case of some data sets. This is because in the two stage process of FTSMKL the learning of kernel is performed separately from the learning of prediction function and hence there is no transfer of knowledge between those two stages. This results in finding a reproducing kernel more close to ideal kernel which leads to overfitting. This problem has been rectified in our model by formulating the entire problem as a single function approximation problem, which inturn helps to use the prediction function f's characteristics in finding the kernel function, f^*.

In experimental results, in the case of accuracy, FTSMKL scored rank 1 for all the data sets while FTSMKL scored rank 1 for only 3 data points out of 13 (Table 4.2). For Fmeasure FSSMKL scored rank 1 for all data sets except one, while FTSMKL scored rank 1 for only 6 data sets (Table 4.3).

The baseline method of FTSMKL uses PEGASOS [75] algorithm which solves primal SVM for linear kernel. In this approach a modified SMO based algorithm is used which is mostly batch processing and applies over dual problem. Therefore FSSMKL takes more time to train than FTSMKL.

4.4.2.2 Analysis of online algorithms

Table 4.4 shows the accuracy and F-measure results of the online algorithms. The OFSSMKL showed a superior performance in this case.

4.5 Conclusion

The optimal kernel searching problem and decision function is formulated as a single function approximation problem. The solution of the approximation problem lies in a RKHS

Table 4.4: Online learning Performance Results: Classification based MKL models and other state-of -the art MKL models

Models	Online Kernel		OMKC		OFSSMKL	
Dataset/Parameters	**Accuracy**	**F-Measure**	**Accuracy**	**F-Measure**	**Accuracy**	**F-Measure**
Arrhythmia	62.91 ± 3.98 (1)	69.61 ± 3.96 (2)	63.42 ± 4.21 (1)	70.19 ± 3.87 (2)	63.73 ± 3.42 (1)	74.98 ± 3.04 (1)
Blood Transfusion	75.63 ± 2.59 (2)	15.91 ± 10.45 (1)	76.51 ± 1.29 (1)	13.53 ± 5.24 (2)	77.39 ± 2.41 (1)	17.67 ± 5.68 (1)
Haberman	72.78 ± 3.66 (2)	83.71 ± 2.68 (2)	69.34 ± 3.39 (3)	79.79 ± 2.45 (3)	74.67 ± 3.48 (1)	85.21 ± 2.29 (1)
Heart	80.51 ± 5.04 (2)	76.67 ± 5.37 (3)	79.33 ± 4.16 (2)	77.53 ± 4.45 (2)	83.02 ± 3.25 (1)	80.53 ± 3.98 (1)
Ionosphere	89.96 ± 2.97 (1)	92.61 ± 2.29 (1)	88.13 ± 2.69 (2)	91.40 ± 1.59 (2)	90.36 ± 2.58 (1)	93.19 ± 2.03 (1)
Liver	64.88 ± 3.67 (1)	71.52 ± 4.04 (2)	62.29 ± 3.61 (2)	72.14 ± 3.89 (2)	65.76 ± 3.59 (1)	73.67 ± 3.65 (1)
Musk 2	84.53 ± 3.47 (1)	83.49 ± 3.82 (1)	80.92 ± 4.16 (2)	82.44 ± 3.59 (2)	84.08 ± 3.03 (1)	83.51 ± 3.53 (1)
Parkinsons	86.55 ± 5.89 (2)	91.66 ± 3.62 (1)	85.82 ± 3.02 (2)	89.69 ± 2.70 (2)	87.47 ± 3.13 (1)	91.22 ± 2.05 (1)
Pima	74.47 ± 2.74 (1)	62.15 ± 4.48 (2)	74.62 ± 2.28 (1)	60.09 ± 4.52 (3)	75.38 ± 2.58 (1)	63.51 ± 4.28 (1)
Sonar	84.89 ± 4.11 (2)	82.55 ± 5.07 (2)	84.51 ± 4.23 (2)	82.81 ± 4.91 (2)	86.97 ± 3.67 (1)	86.15 ± 4.08 (1)
Vertebal Column	80.04 ± 3.57 (2)	84.91 ± 2.77 (2)	80.63 ± 3.57 (1)	83.52 ± 2.76 (2)	81.32 ± 2.82 (1)	86.70 ± 2.13 (1)
Wdbc	94.98 ± 1.34 (1)	96.09 ± 1.04 (1)	92.18 ± 2.92 (2)	94. 67 ± 1.31 (2)	95.25 ± 2.48 (1)	96.63 ± 1.73 (1)
Wholesale Customers	86.18 ± 2.64 (2)	89.67 ± 2.17 (2)	86.06 ± 2.55 (2)	88.09 ± 3.48 (3)	88.72 ± 3.10 (1)	91.88 ± 2.37 (1)
TWO Norm	93.54 ± 0.44 (2)	93.77 ± 0.44 (3)	93.06 ± 0.29 (2)	94.42 ± 0.37 (2)	94.24 ± 0.65 (1)	94.95 ± 0.59 (1)
Ring Norm	86.09 ± 1.20 (2)	84.15 ± 1.56 (2)	86.06 ± 0.75 (2)	84.67 ± 1.55 (1)	87.61 ± 0.92 (1)	85.34 ± 1.34 (1)
Ring Norm	1.6	1.8	1.8	2.13	1	1

space, which is obtained from the direct sum of RKHS related with that of function associated with optimum kernel and prediction. Such a formulation increases the performance of the model as evident from our experimental analysis. The superior performance of the proposed models is due to the fact that in them, the tasks related with finding the functions associated with optimum kernel and with that of decision boundary influences each other, as they are the constituents of the solution of the approximation problem under consideration.

This approach is suitable for binary classification problems only. We have formulated function approximation approach for regression problems also, whose description is given in the next chapter.

Chapter 5

Regression Approach for Multiple Kernel Learning

In the last chapter we have seen a classification approach for finding the optimal kernel. We have formulated MKL as a regression problem for analyzing the regression data, whose description is given in this chapter. For that the methodology used for classification based MKL desribed in [1] is adopted. We have proved that the ideal kernel for this formulation is same as that used in the work of [1].

Both linear as well as nonlinear formulations of regression approach for MKL have been developed. The function resulting from nonlinear combination of base kernels may not be positive semi definite and hence reproducing kernel Krein space (RKKS) concepts are used for formulating the model. In our experimental analysis it has been found that the time and space requirements for nonlinear combination is comparable with that of linear.

The regression framework and the proof for the ideal kernel for regression is given in the next section. The section 5.3 gives the MKL regression formulation for nonlinear settings. The details of experimental analysis can be seen in section 5.4 and finally conclusion is given in section 5.5.

5.1 Regression Framework for MKL

As in the case of classification MKL model, the regression MKL model also consists of finding f^* and f. For developing f^* using regression, input and output data are needed. As the objective of MKL algorithms is to find the best possible kernel, it could be assumed that the output of f^* is the same as the output of the best available kernel (ideal kernel). The ideal kernel formulation for regression is given below.

5.1.1 Ideal Kernel over Regression Data

The minimization problem corresponding to kernel ridge regression (KRR) is

$$\min_{\alpha \in \mathbb{R}^N} \quad \frac{1}{2}\|K\alpha - y\|^2 + \frac{\lambda}{2}\alpha^T K\alpha$$

where K is the kernel matrix, y is the training output vector and $\lambda > 0$ is the regularization parameter. The optimal value of α is given by

$$\alpha = (K + \lambda I)^{-1} y \tag{5.1}$$

Let v be the actual output value for a data point x. Then its predicted output label v_{pred} can be written as

$$\tilde{k}^T \ \alpha = v_{pred} \tag{5.2}$$

where $\tilde{k} = [k(x_1, x) \ k(x_2, x) \ldots k(x_N, x)]^T$.

If the ij^{th} element of the kernel matrix is $k(x_i, x_j) = y_i * y_j$ then (5.1) can be written as below

$$\alpha = (yy^T + \lambda I)^{-1} y \tag{5.3}$$

where $y = [y_1, y_2, \ldots y_N]^T$.

Now $\tilde{k} = yv$ and hence (5.2) becomes

$$v_{pred} = vy^T \ \alpha$$

Using eqn. (5.3)

$$v_{pred} = vy^T \ (yy^T + \lambda I)^{-1} y \tag{5.4}$$

Using Sherman-Morrison Theorem, inverse associated with (5.4) can be found. If A is an invertible square matrix and u, v are column vectors, then Sherman-Morrison formula states that

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1} u} \tag{5.5}$$

If we consider $A = \lambda I$ and $u = v = y$ then

$$(\lambda I + yy^T)^{-1} = (\lambda I)^{-1} - \frac{(\lambda I)^{-1} yy^T (\lambda I)^{-1}}{1 + y^T(\lambda I)^{-1} y} = \frac{I}{\lambda} - \frac{\frac{yy^T}{\lambda^2}}{1 + \frac{y^T y}{\lambda}} \tag{5.6}$$

Now

$$\begin{aligned} y^T(yy^T + \lambda I)^{-1} y &= y^T \left(\frac{I}{\lambda} - \frac{\frac{yy^T}{\lambda^2}}{1 + \frac{y^T y}{\lambda}} \right) y = \frac{y^T y}{\lambda} - \frac{\frac{y^T yy^T y}{\lambda^2}}{1 + \frac{y^T y}{\lambda}} \\ &= \frac{\frac{y^T y}{\lambda}}{1 + \frac{y^T y}{\lambda}} \end{aligned} \tag{5.7}$$

Therefore

$$y^T \ (yy^T + \lambda I)^{-1} y \to 1, \text{when } \lambda \to 0 \tag{5.8}$$

Subsituting eqn. (5.8) in eqn. (5.4) we get

$$v_{pred} = vy^T \ (yy^T + \lambda I)^{-1} y \sim v \times 1 \sim v \tag{5.9}$$

This means that $k(x_i, x_j) = y_i y_j$ is an ideal kernel for regression problems.

5.1.1.1 Data Compression

As discussed in previous chapter, the data points corresponding to f^* scales as $O(N^2)$ and hence the algorithm complexity increases with increase of data. The supervised pre-clustering approach, whose decsription is given in section 2.4.1, is used for compressing the data in an efficient manner.

If $M < N$ are the data points after compression, then f^* can be found using the $M^2 < N^2$ data points $\left\{ \left(\tilde{K}(x_i, x_j), y_i y_j \right), i, j = 1, 2, \ldots M \right\}$.

Let the reproducing kernel corresponding to f^* be k^*. On the basis of M^2 data points obtained after data compression,

$$f^*(\tilde{K}(x_k, x_l)) = \sum_{i=1}^{M^2} \beta_i k^*(z_i, \tilde{K}(x_k, x_l)), \beta_i \in \mathbb{R} \tag{5.10}$$

where $z_i \in \{\tilde{K}(x_i, x_j), i, j = 1, 2, \ldots M\}$.

5.2 Linear Combination of Kernels

As seen in the last chapter, for linear MKL, f^* is a hyperplane and hence k^* is a linear kernel. Therefore (5.10) becomes

$$f^*(\tilde{K}(x_k, x_l)) = \sum_{i=1}^{M^2} \beta_i z_i^T \tilde{K}(x_k, x_l) = d^T \tilde{K}(x_k, x_l)$$

where $d = \sum_{i=1}^{M^2} \beta_i z_i$. Once the weight vector d is calculated, the negative values from the weight vector are clipped to make sure that the overall linear combination of kernels is positive semidefinite.

5.2.1 Two Stage Approach

We used two stage optimization for finding f and f^*, that is f^* is first solved and then f is found out using the new f^*. Given below is the discussion of finding the parameters of the model using kernel ridge regression.

As described earlier, M^2 data points found out using pre-clustering approach are used to train f^*, whose corresponding outputs are generated using the ideal kernel, discussed in section 5.1.1.

Let $\hat{K}$ be the kernel matrix associated with f. Then its ij^{th} element $\hat{k}_{ij} = f^*(\tilde{K}(x_i, x_j))$. The optimal α associated with f is found out by minimizing

$$\frac{1}{2}\|\hat{K}\alpha - y\|^2 + \frac{\lambda}{2}\alpha^T \hat{K}\alpha$$

On solving this equation, we get α as

$$\alpha = (\hat{K} + \lambda I)^{-1} y \tag{5.11}$$

5.3 Nonlinear Combination of Kernels

We extended the above described MKL theory to nonlinear settings whose description is given below.

In the case of nonlinear MKL, f^* is not a hyperplane. Also, the nonlinear combination of kernels may not be positive definite, which can be explained as follows.

Corresponding to each $z_i, i = 1, 2, \ldots M^2$ as given in (5.10), define reproducing kernel $k^i : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that,

$$k^i(x_k, x_l) = k^* \left(z_i, \tilde{K}(x_k, x_l) \right), x_k, x_l \in \mathcal{X} \tag{5.12}$$

With respect to the data $\{x_i\}_{i=1}^N$, let K^i be the kernel matrix associated with reproducing kernel k^i. Let

$$K^P = \sum_{\beta_i > 0} \beta_i K^i$$

and

$$K^N = \sum_{\beta_i < 0} \beta_i K^i$$

where $\beta_i, i = 1, 2, \ldots M$ is as given in (5.10).

Therefore in the case of nonlinear MKL, the matrix $\hat{K} = K^P - K^N$ and hence it may not be positive semi-definite matrix. Hence in this case it is assumed that f lies in RKKS $\mathcal{K}$ with Hermitian kernel $\tilde{k}$. The RKKS concepts are explained below.

5.3.1 Reproducing Kernel Krein Space

The Krein space is an indefinite innerproduct space endowed with a Hilbertian topology.

5.3.1.0.1 Indefinite inner product space [76, 77] : Let $\mathcal{K}$ be a vector space over $\mathbb{R}$. An indefinite inner product $< ., . >_\mathcal{K}$ on $\mathcal{K}$ is a bilinear form where for all $f, g, h \in \mathcal{K}, \alpha \in \mathbb{R}$:

- $< f, g >_\mathcal{K} = < g, f >_\mathcal{K}$
- $< \alpha f + g, h >= \alpha < f, h >_\mathcal{K} + < g, h >_\mathcal{K}$
- $< f, g >= 0, \forall g \in \mathcal{K} \implies f = 0$

5.3.1.0.2 Krein Space [76, 77] : An indefinite inner product space, $(\mathcal{K}, < ., . >_\mathcal{K})$ is a Krein space if there exists two Hilbert spaces $\mathcal{H}_+, \mathcal{H}_-$ spanning $\mathcal{K}$, such that,

- $\forall f_\mathcal{K} \in \mathcal{K}, f_\mathcal{K} = f_+ + f_-$, where $f_+ \in \mathcal{H}_+$ and $f_- \in \mathcal{H}_-$
- $\forall f_\mathcal{K}, g \in \mathcal{K}, < f_\mathcal{K}, g >_\mathcal{K} = < f_+, g_+ >_{\mathcal{H}_+} - < f_-, g_- >_{\mathcal{H}_-}$

$\mathcal{K}$ is a reproducing kernel Krein space (RKKS) if $\mathcal{H}_+$ and $\mathcal{H}_-$ are RKHSs [77, 76]. In a RKKS $\mathcal{K}$, there is a unique symmetric kernel $k(x, x')$ with $k_x \in \mathcal{K}$ such that the

reproducing property holds: for all $f \in \mathcal{K}$, $f(x) =< f, k_x >_{\mathcal{K}}$ and $k = k_+ + k_-$ where k_+ and k_- are the reproducing kernels of the RKHSs H_+ and H_-. Corresponding to any symmetric non-positive kernel k that can be decomposed as the difference of two positive kernels k_+ and k_-, there exists a RKKS associated to it.

In the next section we derive the indefinite kernel ridge regression.

5.3.2 Formulation of Indefinite Kernel Ridge Regression

The authors of [76] formulated SVM in Krein space. By making use of those concepts we derived indefinite kernel ridge regression in Krein space.

The minimization problem related with KRR can be formulated in Krein space as

$$\underset{f_{\mathcal{K}} \in \mathcal{K}}{stabilize} \ \frac{1}{2}\sum_{i=1}^{N} < (f_{\mathcal{K}}(x_i) - y_i), (f_{\mathcal{K}}(x_i) - y_i) > + \frac{\lambda}{2} < f_{\mathcal{K}}, f_{\mathcal{K}} > \tag{5.13}$$

As per the Krein space theory, $f_{\mathcal{K}}(x) = f_+(x) - f_-(x)$. Therefore (5.13) becomes

$$\begin{aligned} \min_{f_+ \in \mathcal{H}_+} \max_{f_- \in \mathcal{H}_-} \ & \frac{1}{2}\sum_{i=1}^{N} \left(f_+(x_i) - f_-(x_i) - y_i\right)^2 \\ & + \frac{\lambda}{2} < f_+, f_+ > - \frac{\lambda}{2} < f_-, f_- > \end{aligned} \tag{5.14}$$

Let K^+ and K^+ be the kernel matrices associated with f_+ and f_- respectively. Therefore, (5.14) becomes

$$\min_{\alpha^+ \in \mathbb{R}^{\mathbb{N}}} \max_{\alpha^- \in \mathbb{R}^{\mathbb{N}}} \ \frac{1}{2}\|K^+\alpha^+ - K^-\alpha^- - Y\|^2 + \frac{\lambda}{2}\alpha^+ K^+ \alpha^+ - \frac{\lambda}{2}\alpha^- K^- \alpha^- \tag{5.15}$$

The solution for the above equation can be found by fixing the value of one parameter to find the other.

On fixing the α^-, the problem statement becomes

$$\min_{\alpha^+ \in \mathbb{R}^{\mathbb{N}}} \ \frac{1}{2}\|K^+\alpha^+ - G_-\|^2 + \frac{\lambda}{2}\alpha^+ K^+ \alpha^+ - const \tag{5.16}$$

where $G_- = Y + K^-\alpha^-$. The normal equation corresponding to the above minimization problem is

$$K^+K^+\alpha^+ - K^+G_- + \lambda K^+\alpha^+ = 0$$

On fixing α^+, due to the negative quadratic term, the problem becomes a concave optimization problem, that is,

$$\max_{\alpha^- \in \mathbb{R}^{\mathbb{N}}} \quad \frac{1}{2}\|G^+ - K^-\alpha^-\|^2 - \frac{\lambda}{2}\alpha^- K^-\alpha^- + const \tag{5.17}$$

where $G_+ = K^+\alpha^- - Y$. The corresponding normal equation is

$$K^-K^-\alpha^- - K^-G_- - \lambda K^-\alpha^- = 0$$

The solutions to the above normal equations can be found by using two stage formulation as described in section 5.2.1.

5.3.3 Computation of Parameters of Nonlinear MKL

As described earlier, the regression model involves the computation of f and f^*. The function f^* from RKHS $\mathcal{F}$ can be found out using KRR. In nonlinear case it is assumed that f lies in RKKS $\mathcal{K}$. We applied indefinite kernel ridge regression to find the parameters corresponding to f.

5.4 Experiments

5.4.1 Experimental Setup

The performance of the algorithms was assessed experimentally. The MKL model used in the analysis consisted of 42 kernels as similar to the experimental setup in chapter 4.

The linear MKL model described in this chapter is named as Two stage Multiple kernel learning approach for regression (TSMKLR) and that of nonlinear as Nonlinear Two stage Multiple kernel learning approach for regression (NTSMKLR). The regression algorithms were implemented in Matlab. The performance of NTSMKLR and TSMKLR was compared with that of SimpleMKL [19] and SPG-MKL[78] (a modified version of GMKL[21]). The codes for SimpleMKL [19] and SPG-MKL[78] were taken from the repository [79] and [80] respectively. All the experiments were conducted in the same machine throughout with 80 GB RAM, intel Xeon Processor. The performance for the proposed model was assessed using root mean square (RMSE). The datasets were collected

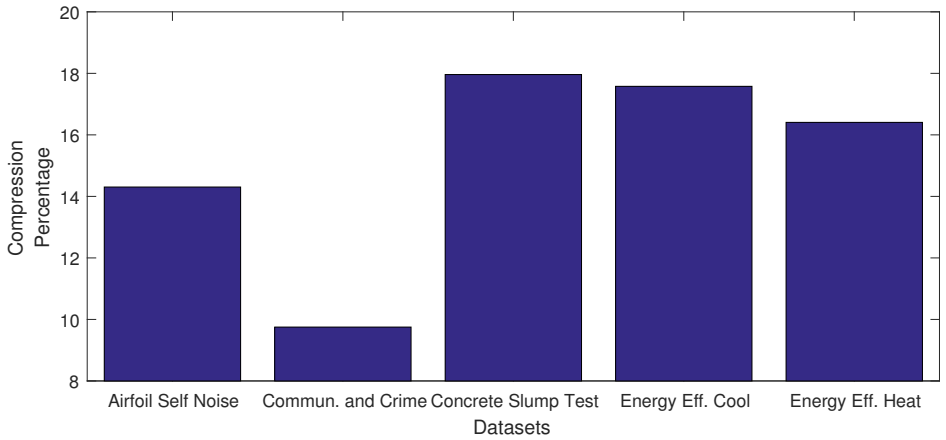

Figure 5.1: Data Compression Rate using Preclustering Algorithm

from UCI repository [60].

In the regression experiments, as similar to the rank analysis for classification experiments in chapter 3, the models were ranked for their performance on each data. For example: let M_1 and M_2 be two models; let P_1 and P_2 be the values of a performance measure P for a given data set D. Then we say that M_1 is better than M_2 on the basis of P on D if $P_1 < P_2$ and their difference is statistically significant.

5.4.2 Results and Discussions

Using pre-clustering approach the data was compressed. The ratio of compression for the datasets is shown in Fig. 5.1. The compressed data is used to compute the training points for f^*. Using f^*, f was computed. The experimental results are shown in Table 5.1. It shows that NTSMKLR produced superior results in comparison with other models as it scored the highest rank for all the data sets.

Table 5.1: TSMKL Results

Dataset	KRR	SimpleMKL	SPG-GMKL	TSMKL	NTSMKL
Airfoil.	4.22529 $\pm$ +0.17282 (4)	3.83287 $\pm$ +0.20978 (3)	3.40593 $\pm$ +0.32411 (2)	3.13291 $\pm$ 0.29307 (1)	3.09874 $\pm$ 0.26234 (1)
Commun. & Cr.	5.82782 $\pm$ +0.33076 (2)	5.79657 $\pm$ +0.29028 (2)	5.86840 $\pm$ +0.29237 (2)	5.00437 $\pm$ 0.31056 (1)	4.91283 $\pm$ 0.26034 (1)
Conc. Sl.	7.53245 $\pm$ +0.51391 (5)	6.48337 $\pm$ +0.45852 (4)	6.09983 $\pm$ +0.52536 (3)	5.46865 $\pm$ +0.38802 (2)	5.276845 $\pm$ +0.36923 (1)
Eff. Cool	1.85125 $\pm$ +0.12772 (4)	1.33792 $\pm$ +0.10755 (3)	1.23957 $\pm$ +0.10164 (3)	1.15763 $\pm$ 0.10176 (2)	1.097123 $\pm$ 0.09034 (1)
Eff. Heat	2.68947 $\pm$ +0.18045 (5)	2.40471 $\pm$ +0.21294 (4)	1.40673 $\pm$ +0.13037 (3)	1.04312 $\pm$ +0.14548 (2)	1.00458 $\pm$ +0.09172 (1)
AVERAGE RANK	4	3.2	2.6	1.6	1

5.5 Conclusion

The two stage MKL algorithm for regression domain is discussed in this chapter. The proof of the ideal kernel regression, that is $k(x_i, x_j) = y_i y_j$ is also described. The supervised pre-clustering approach is used to select the vital data points for reducing the overall time and space complexity. Using the concepts of Krein space, nonlinear MKL is formulated. The experiment results clearly proved that the proposed framework is a suitable approach in finding the optimal kernel for regression data.

In the next chapter we introduce a MKL model based on composite functions.

Chapter 6

Learning Kernel using Composite Kernel Functions

The formulation of MKL using composite kernel functions (MKLCKF), in which the optimal kernel is represented as the linear combination of composite kernel functions is described in this chapter. For constructing a composite kernel model, with reference to each data point, a composite kernel function is introduced such that it makes use of the information of all the given P base kernels for finding the image at each of the points in its domain. We are proposing two variants of this formulation. In the first variant, the optimal kernel is represented as a linear combination of newly designed kernels. As each composite kernel function is built upon a data point, we introduce a second variant in which the coefficients of the linear combination are replaced with a neighborhood function of the reference data point. This representation makes the algorithm more computationally efficient. We verified the efficiency of the proposed models using real world data sets and compared its performance with existing techniques. The proposed methods showed excellent performance. Of the two variants of the approach, the performance of the second variant was found to be better.

In section 6.1, the details of the theory behind the proposed weighted kernel approach and its applications are given. The section 6.1.4 explains the localized approach and its applications while the experimental analysis is given in section 6.2.

6.1 Multiple Kernel Learning using Composite Kernel Functions (MKLCKF)

Consider a pool of P kernels $\{k_1, k_2, \ldots k_p\}$ from which the best combination of kernels has to be chosen. For that using each data as the reference point we constructed N composite kernel functions. This section describes the construction of those kernels.

Define an operator $\tilde{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^P$ as

$$\tilde{K}(x, z) = [k_1(x, z) \;\; k_2(x, z) \;\; \ldots \;\; k_p(x, z)]^T$$

where P is the number of base kernels.

Corresponding to each $x_j, j = 1, 2, \ldots N$, composite function, $k^j : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, is constructed where

$$k^j(x_k, x_l) \; = < k^j_{x_k}, k^j_{x_l} > = k^* \left(\tilde{K}(x_k, x_j), \tilde{K}(x_j, x_l) \right), \tag{6.1}$$

Here $k^* : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ is any valid reproducing kernel and $Z = \mathcal{R}(\tilde{K})$, where, $\mathcal{R}(\tilde{K})$ is the range space of $\tilde{K}$.

Theorem 1. *The composite function given by* (6.1) *is a valid kernel.*

Proof. Let $\phi_j(x_k) = \tilde{K}(x_k, x_j)$ and $\phi_j(x_l) = \tilde{K}(x_j, x_l)$. Then (6.1) becomes

$$k^j(x_k, x_l) = k^* \left(\phi_j(x_k), \phi_j(x_l) \right)$$

By [4], $\hat{k}(x, x') = k_1(\phi(x), \phi(x'))$ is a valid kernel where $\phi : \mathcal{X} \to R^M$ and k_1 is a valid kernel defined on R^M. Therefore k^j is a valid kernel. □

Each k^j consists of two layers of functions, where the first layer is defined from $\mathcal{X} \times \mathcal{X} \to Z \times Z$ and the second layer is from $Z \times Z \to \mathbb{R}$. With the aid of such a design the composite kernel makes use of the information of all the base kernels for finding the image.

Using this idea we developed two variants of MKL algorithm which we named as MKLCKF:I and MKLCKF: II.

6.1.1 MKLCKF:I

Let $\mathcal{F}^j$ be the RKHS corresponding to $k^j, j = 1, 2, \ldots N$. The function f is assumed to lie in a RKHS, $\mathcal{F}$, with reproducing kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ where

$$\begin{aligned} k(x_k, x_l) &= \sum_{j=1}^{N} \beta_j k^j(x_k, x_l) \\ &= \sum_{j=1}^{N} \beta_j k^* \left(\tilde{K}(x_j, x_k), \tilde{K}(x_j, x_l) \right) \end{aligned} \tag{6.2}$$

where, $\beta_j \geq 0, \forall j = 1, 2, \ldots, N$.

Theorem 2. *The kernel given by* (6.2) *is a valid kernel.*

Proof. By theorem 1, $k^j, j = 1, 2, \ldots N$ are valid kernels. Therefore k is a conical linear combination of N kernel functions. Hence by [4] k is a valid reproducing kernel. □

The representation of k as the linear combination of k^j helps to include the information of all the P base kernels in an efficient manner for finding f.

The cost function used for approximating f is as given below.

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{N} l\left(y_i, f(x_i)\right) + \eta \|f\|^2. \tag{6.3}$$

where l is a differentiable loss function. Then,

$$\begin{aligned} f(x) &= \sum_{i=1}^{N} \alpha_i k(x_i, x) \\ &= \sum_{i=1}^{N} \alpha_i \sum_{j=1}^{N} \beta_j k^* \left(\tilde{K}(x_i, x_j), \tilde{K}(x_j, x) \right) \end{aligned} \tag{6.4}$$

In order to impose a controlled regularization, the constraint of $\sum_{i=1}^{N} \beta_i = 1$ can be imposed. The whole idea is summarized in Fig. 6.1.

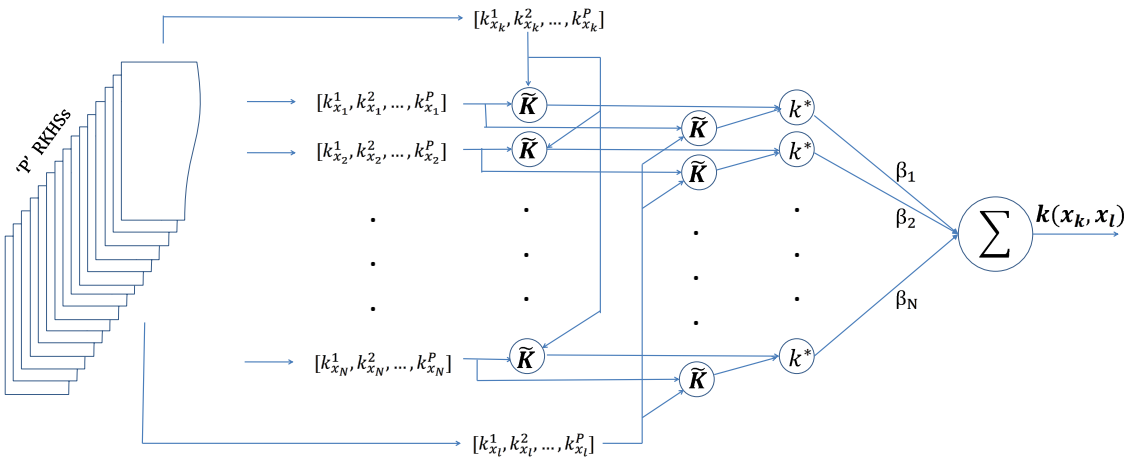

Figure 6.1: MKLCKF:I

6.1.2 Application in Support Vector Machine (SVM)

For classification problems, SVM algorithm was used to determine the unknown function. The optimization problem corresponding to SVM classification is

$$\frac{1}{2}\|f\|^2 + C\sum_i \xi_i$$

$$sub\ \ to$$

$$y_i(f(x_i) + b) - 1 + \xi_i \geq 0$$

$$\xi_i \geq 0$$

The corresponding dual using the kernel given in equation (6.2) can be written as follows.

$$\sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \sum_k \beta_k k^* \left(\tilde{K}(x_i, x_k), \tilde{K}(x_k, x_j)\right)$$

$$sub\ \ to$$

$$0 \leq \alpha_i \leq C$$

$$\beta_k \geq 0$$

$$\sum_i \alpha_i y_i = 0$$

$$\sum_{k=1}^{N} \beta_k = 1$$

6.1.2.1 Optimization

For solving the above described optimization problem, in the first stage β is fixed as $\frac{1}{N}$ and then optimize for α using conventional SVM sovler, namely, SMO. In the second stage, the β is updated using reduced gradient descent method as in [19].

6.1.3 Application in Support Vector Regression (SVR)

SVR algorithm was applied to determine the unknown function for regression problems. The optimization problem corresponding to SVR is

$$\begin{aligned}
&\frac{1}{2}\|f\|^2 + C\sum_i [\xi_i + \xi_i^*] \\
&sub\ \ to \\
&\quad y_i - f(x_i) - b + \epsilon - \xi_i \leq 0 \\
&\quad f(x_i) + b - y_i + \epsilon - \xi_i^* \leq 0 \\
&\quad \xi_i \geq 0 \\
&\quad \xi_i^* \geq 0
\end{aligned}$$

where $b \in \mathbb{R}$ is the bias and $C > 0$ is the regularization parameter. The corresponding dual using the kernel given in equation (6.2) can be written as follows.

$$\begin{aligned}
&\sum_{i=1}^{N} \alpha_i y_i - \epsilon \sum_{i=1}^{N} |\alpha_i| - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j \sum_k \beta_k k^* \left(\tilde{K}(x_i, x_k), \tilde{K}(x_k, x_j)\right) \\
&\quad sub\ \ to \\
&\qquad -C \leq \alpha_i \leq C \\
&\qquad \beta_k \geq 0 \\
&\qquad \sum_i \alpha_i = 0 \\
&\qquad \sum_{k=1}^{N} \beta_k = 1
\end{aligned}$$

The optimization is performed similar to that explained in section 6.1.2.1.

6.1.4 MKLCKF:II

Using (6.2),

$$k(x_k, x_l) =< k_{x_k}, k_{x_l} >= \left\langle \begin{bmatrix} \sqrt{\beta_1}k^1_{x_k} \\ \sqrt{\beta_2}k^2_{x_k} \\ \vdots \\ \sqrt{\beta_N}k^N_{x_k} \end{bmatrix}, \begin{bmatrix} \sqrt{\beta_1}k^1_{x_l} \\ \sqrt{\beta_2}k^2_{x_l} \\ \vdots \\ \sqrt{\beta_N}k^N_{x_l} \end{bmatrix} \right\rangle \tag{6.5}$$

where,

$$\begin{aligned} k_{x_k} &= \left[\sqrt{\beta_1}k^1_{x_k}, \sqrt{\beta_2}k^2_{x_k}, \ldots \sqrt{\beta_N}k^N_{x_k}\right]^T \\ &= \left[\sqrt{\beta_1}k^*_{\tilde{K}(x_1,x_k)}, \sqrt{\beta_2}k^*_{\tilde{K}(x_2,x_k)} \cdots \sqrt{\beta_N}k^*_{\tilde{K}(x_N,x_k)}\right]^T \end{aligned}$$

Thus in this method N parameters $\{\beta_1, \beta_2, \ldots \beta_N\}$ have to be learned . For making the approach more computationally effective we formulated the algorithm MKLCKF: II, in which a neighborhood function η_i of x_i is introduced in the place of β_i. The whole idea is summarized in Fig. 6.2. We defined

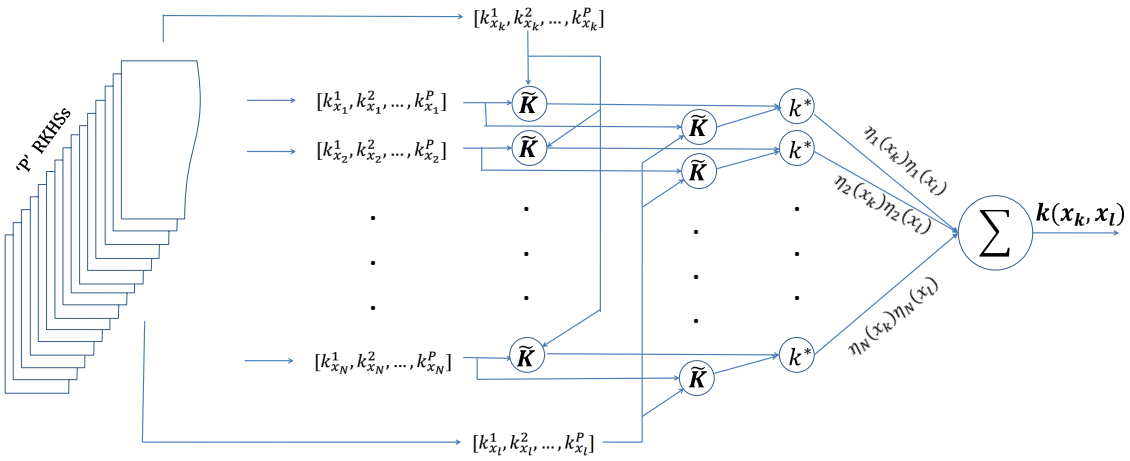

Figure 6.2: MKLCKF:II

$$\eta_i(x) = exp\left(-\gamma d(x_i, x)\right) \tag{6.6}$$

where $d(x_i, x)$ is a distance metric and $\gamma > 0$. Such a formulation is used as the composite kernel k^i depends greatly on the data point x_i. We used Euclidean distance metric in our experiments. It is clear from (6.6), η_i's contribution is highest for x_i's neighbors and hence

it helps to capture the local information of x_i. Now

$$k_{x_k} = \begin{bmatrix} \eta_1(x_k)k^1_{x_k} \\ \eta_2(x_k)k^2_{x_k} \\ \vdots \\ \eta_N(x_k)k^N_{x_k} \end{bmatrix} = \begin{bmatrix} \eta_1(x_k)k^*_{\tilde{K}(x_1,x_k)} \\ \eta_2(x_k)k^*_{\tilde{K}(x_2,x_k)} \\ \vdots \\ \eta_N(x_k)k^*_{\tilde{K}(x_N,x_k)} \end{bmatrix} \tag{6.7}$$

Therefore, the final kernel is,

$$\begin{aligned} k(x_k, x_l) &= \sum_{j=1}^{N} \eta_j(x_k)\ \eta_j(x_l) k^j(x_k, x_l) \\ &= \sum_{j=1}^{N} \eta_j(x_k)\ \eta_j(x_l)\ k^*\left(\tilde{K}(x_j, x_k), \tilde{K}(x_j, x_l)\right) \end{aligned} \tag{6.8}$$

Theorem 3. *The kernel given by 6.8 is a valid kernel.*

Proof. By theorem 1 $k^j, j = 1, 2, \ldots N$ are valid kernels. Using (6.8), construct the $N \times N$ matrix $K = (k(x_i, x_j)), i, j = 1, 2, \ldots N$. Then K can be represented as

$$K = \sum_{j=1}^{N} \Lambda^j\ K^j\ \Lambda^j \tag{6.9}$$

where Λ^j is a diagonal matrix of order $N \times N$ such that, i^{th} diagonal element, $\Lambda^j(i, i) = \eta_j(x_i)$ and K^j is the kernel matrix of k^j corresponding to N points $\{x_1, x_2, \ldots x_N\}$. Λ^j is positive semidefinite (p.s.d) matrix as it is a diagonal matrix with diagonal entries positive . Hence $\Lambda^j\ K^j\ \Lambda^j$ is symmetric positive definite matrix. As K is a linear combination of symmetric p.s.d matrices it is a symmetric p.s.d [4], [3]. Hence k is a valid kernel. □

6.1.4.1 Computational Advantage of MKLCKF

The MKLCKF:I uses SimpleMKL for solving the dual function for which time complexity is $O(N^2P) + O(N^{2.2})$ where N is the number of training points and P is the number of kernels. In MKLCKF:I approach, $P = N$. Therefore overall time complexity for MKLCKF:I is $O(N^3P) + O(N^3) \simeq O(N^3P)$.

In case of MKLCKF:II, there are no additional parameters for learning the kernel. The construction of kernel corresponding to MKLCKF:II is given in Algorithm 4. The time complexity for generating P base kernel matrices is $O(N^2Pd)$. Those kernel matrices are

Algorithm 4 Kernel Construction Algorithm for MKLCKF : II

1: **procedure** COMPUTEKERNEL(datapoints)
2: Initialize $K = zeros(N, N)$ ▷ N is the number of data points
3: **for** $i = 1\ to\ N$ **do**
4: $Ktemp = kernel(i, datapoints)$ ▷ kernel() is the function which computes the kernel matrix from (6.1)
5: $eta = eta(i, datapoints)$ ▷ eta is the function which computes η vector from (6.6)
6: $K = K + ((eta * eta^T). * Ktemp)$
7: **end for**
8: **return** K
9: **end procedure**

used for computing the output of $\tilde{K}$ (line 2). In Algorithm 4, the complexity for computing the function $kernel(i, datapoints)$ is $O(N^2P)$ (line 4) and complexity for computing the function $eta(i, datapoints)$ is $O(Nd)$ (line 5). The time complexity for line 6 is $O(N^2)$. The lines 4, 5 and 6 are executed for N times. Thus the total time complexity of the Algorithm 4 is $O(N^3P) + O(N^2Pd) \simeq O(N^3P)$ (considering $d < N$). Hence this algorithm has the similar time complexity as that of the other high performing MKL approaches.

6.2 Experiments

6.2.1 Experimental Setup

The experiments were conducted using classification and regression datasets. The classification datasets used are listed in Table 6.1 which are binary class classification problems taken from UCI repository[60] and IDA benchmark repository [61]. The regression datasets used for analysis are listed in Table 6.2. The datasets are taken from UCI, mldata.org and DELVE repositories.

In MKL algorithm the P base reproducing kernels are the 42 base kernels that were generated as in the experimental setup with Chapter 4.

The specification of the machine used for the experiment was intel i7 M4810 processor with 16GB RAM. The hyperparameters of the models used for our study were determined using 5-fold cross validation. We used 30 times holdout technique for model evaluation. Their performance were assessed using root mean square (RMSE) for regression and accuracy and F-measure for classification.

For our study we chose both '*large data*' as well as *'not large data'*. The *large data*

Table 6.1: Classification Datasets for Analysis using MKLCKF

Dataset	**Repo.**	**Dim.**	**Data Points**
Arrythmia	UCI	276	452
Haberman	UCI	3	306
Heart	UCI	13	303
Ionosphere	UCI	33	351
Liver	UCI	6	345
Musk 2	UCI	166	476
Parkinsons	UCI	22	195
Pima	UCI	8	768
Sonar	UCI	60	208
Vert. Column	UCI	6	310
WDBC	UCI	30	569
Whole. Cust.	UCI	7	440
Twonorm	IDA	20	7400
Ringnorm	IDA	20	7400

Table 6.2: Regression Datasets for Analysis using MKLCKF

Dataset	**Repository**	**Points.**	**Dim.**
Ailerons	mldata.org	13750	40
Airfoil Self Noise	UCI	1503	5
Bank32NH	DELVE	8192	32
Commun. and Crime	UCI	2215	99
Concrete Slump Test	UCI	1030	8
Elevators	Exp. of Rui Camacho	16599	18
Energy Eff. Cool	UCI	768	8
Energy Eff. Heat	UCI	768	8
2D-Planes	breiman1984	40768	10
Video Char.	UCI	68784	20
Protein Tertiary Str.	UCI	45730	9

is defined as those for which application of MKL produces out of memory problem in the machine which we used for computation and others are being called 'not large'.

The performance of MKLCKF:I and MKLCKF:II was compared with the following models:

1 **SMO-SVM :** Normal SMO-SVM model that uses a single kernel.

2 **SimpleMKL[19] :** Linear Combination of kernel approach is used in this model.

3 **TSMKL[1] :** The model uses function approximation concepts and two stage learning process termed as Two Stage MKL.

4 **GMKL[21] :** A regularized formulation of MKL termed as generalized multiple kernel learning algorithm.

5 **LpMKL [69]** An Lp Norm regularized MKL Algorithm.

Codes for GMKL is taken from authors url [72] . For LpMKL we used the code from LibMKL [74]. The framework in java for MKL named jKernelMachine [59] is used for other state-of-the-art algorithms. The same framework was customized for implementing the proposed algorithm.

The kernel k^* for MKLCKF models was found using cross validation technique. The kernel k^* and the kernel selected for single kernel approach (SVM-SMO) was the same for all the experiments.

6.2.2 Classification Experiments

Large data and *not large data* related with classification are selected for evaluating the performance of the models. For *not large data*, the accuracy results are shown in Table 6.3 and F-measure results in Table 6.4. It is evident from the Table 6.3 that for datasets such as Arrhythmia, Haberman, Heart, Ionosphere, Musk 2, Parkinsons, Pima and Wholesale Customers, the proposed model MKLCKF:II produced good results. The MKLCKF:I is performing well on Ionosphere, Liver and Parkinsons. For all other models, the results are comparable with the best among state-of-the-art approaches. The F-measure analysis results shown in Table 6.4 depicts a similar picture. Eventhough a few of the state-of-the-art models produced good results for some datasets, none of those algorithms performed consistently over all datasets. On the other hand our methods showed a consistent performance.

Table 6.3: Accuracy: Composite Kernel Function Models and other State-of-the -art Models

Dataset/Models	SMO SVM	SimpleMKL	TSMKL	LpMKL	GMKL	MKLCKF:I	MKLCKF:II
Arrhythmia	**76.98 ± 3.18**	75.89 ± 3.19	74.96 ± 2.52	76.81 ± 2.60	76.42 ± 3.17	75.45 ± 3.29	**79.38 ± +3.27**
Haberman	73.22 ± 5.76	73.31 ± 5.23	73.84 ± 4.41	73.21 ± 1.71	**73.75 ± 5.36**	73.87 ± 3.49	**74.62 ± +3.54**
Heart	83.48 ± 3.28	82.78 ± 2.87	81.96 ± 3.08	82.84 ± 3.37	**83.21 ± 3.19**	84.23 ± 2.74	**84.96 ± +3.07**
Ionosphere	93.74 ± 2.20	93.87 ± 2.20	94.44 ± 2.15	**94.52 ± 1.69**	93.93 ± 2.31	**95.55 ± 1.67**	**95.14 ± +2.20**
Liver	71.42 ± 4.97	71.87 ± 3.97	**73.04 ± 2.91**	72.66 ± 3.69	71.25 ± 3.24	**73.34 ± 2.89**	72.83 ± +3.40
Musk 2	92.13 ± 2.76	**92.51 ± 2.68**	**92.44 ± 1.80**	92.10 ± 2.66	92.34 ± 2.61	92.79 ± 2.09	**93.10 ± +1.88**
Parkinsons	88.21 ± 4.67	88.39 ± 4.01	90.34 ± 3.40	**90.29 ± 2.66**	89.32 ± 4.25	**91.42 ± 3.43**	**92.47 ± +3.60**
Pima	76.53 ± 2.07	**76.76 ± 2.86**	75.78 ± 2.95	**76.28 ± 2.39**	74.89 ± 3.63	76.81 ± 2.41	**77.74 ± +2.60**
Sonar	85.03 ± 4.10	85.69 ± 4.47	**88.06 ± 4.27**	87.63 ± 4.01	86.24 ± 4.92	87.15 ± 3.29	88.39 ± +3.54
Vertebal Column	83.90 ± 3.64	**84.13 ± 4.25**	82.65 ± 2.46	83.23 ± 2.85	83.06 ± 2.78	84.08 ± 3.60	84.80 ± +3.16
Wdbc	96.94 ± 0.91	97.06 ± 0.83	97.03 ± 1.28	**97.32 ± 1.01**	97.29 ± 1.13	97.15 ± 1.13	97.12 ± +1.16
Wholesale Customers	90.73 ± 2.00	**90.89 ± 2.52**	90.78 ± 2.06	90.58 ± 1.45	90.65 ± 2.82	**91.59 ± 1.96**	**92.22 ± +1.68**

Table 6.4: F Measure: Composite Kernel Function Models and other State-of-the -art Models

Dataset/Models	SMO SVM	SimpleMKL	TSMKL	LpMKL	GMKL	MKLCKF:I	MKLCKF:II
Arrhythmia	**79.71 ± 3.02**	77.91 ± 3.24	77.17 ±2.26	77.93 ± 1.89	**79.23 ± 3.42**	78.53 ± 3.02	**82.11 ± +2.70**
Haberman	83.41 ± 4.05	83.11 ± 5.01	**84.08 ± 2.87**	**84.11 ± 1.26**	83.54 ± 4.15	84.43 ± 2.31	**85.42 ± +2.33**
Heart	**81.28 ± 4.17**	**81.13 ± 3.45**	80.10 ± 4.10	80.87 ± 4.12	80.73 ± 3.57	**82.01 ± 3.36**	**82.90 ± +3.62**
Ionosphere	95.20 ± 1.73	95.26 ± 1.75	**95.72 ± 1.72**	95.11 ± 1.28	95.38 ± 1.53	**96.57 ± 1.29**	96.08 ± +1.85
Liver	76.79 ± 4.42	77.13 ± 4.12	**78.42 ± 3.09**	77.17 ± 3.39	77.21 ± 4.22	78.78 ± 2.89	77.98 ± +2.96
Musk 2	90.91 ± 3.46	90.66 ± 3.10	**91.52 ± 1.77**	90.89 ± 3.01	90.67 ± 3.06	91.61 ± 2.38	91.94 ± +2.31
Parkinsons	92.53 ± 3.18	91.98 ± 2.65	**92.69 ± 2.29**	91.91 ± 1.76	91.83 ± 2.58	**94.54 ± 2.30**	**94.93 ± +2.47**
Pima	62.37 ± 3.42	**62.61 ± 4.09**	61.46 ± 4.50	61.38 ± 4.22	62.29 ± 3.92	62.54 ± 4.89	**65.41 ± +3.56**
Sonar	83.24 ± 5.12	83.77 ± 5.36	**86.80 ± 4.71**	85.31 ± 4.31	85.36 ± 4.62	85.98 ± 3.90	**87.77 ± +3.55**
Vertebal Column	**87.99 ± 3.64**	**87.81 ± 3.98**	87.07 ± 2.07	87.63 ± 2.14	87.27 ± 2.83	**88.37 ± 2.96**	**88.66 ± +2.23**
Wdbc	97.14 ± 0.76	97.17 ± 0.64	**97.64 ± 1.04**	97.53 ± 0.79	97.23 ± 0.60	97.84 ± 0.92	97.73 ± +0.91
Wholesale Customers	93.10 ± 1.55	**93.24 ± 1.87**	93.12 ± 1.58	93.07 ± 1.07	93.56 ± 2.37	93.82 ± 1.37	**94.82 ± +1.36**

6.2.2.1 Analysis on Large Data Sets

MKL analysis over ringnorm and twonorm datasets ended in out of memory error. We analysed the performance on these datasets by applying dictionary learning technique explained in 2.4.2. We followed a 30 times hold out approach for fixing the size of dictionary. The dictionary points thus obtained are used as the fixed points in eqn. (6.8). *That is* if $d_j,\ j = 1 \dots N_1$ are the dictionary atoms, then (6.8) becomes

$$k(x_k, x_l) = \sum_{j=1}^{N_1} \eta_i(x_k) k^* \left(\tilde{K}(d_j, x_k), \tilde{K}(d_j, x_l) \right) \eta_i(x_l) \tag{6.10}$$

The analysis results are plotted in Fig. 6.3 and Fig. 6.4. The superior performance of MKLCKF models is evident from these results.

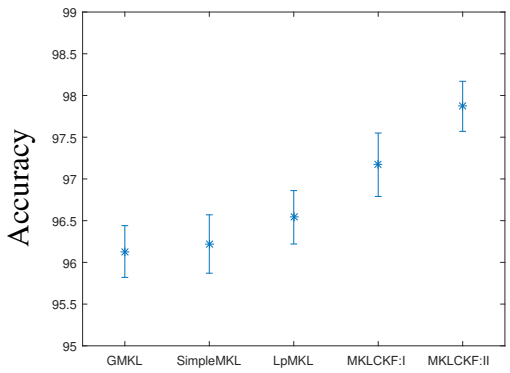

Figure 6.3: Accuracy Analysis using Two Norm Data: MKLCKF and other state-of-the-art MKL models

6.2.3 Analysis on Image Datasets

Apart from the data sets described above, we did analysis using image data sets also. Using Caltech101 [81] and UIUC sports scene classification [82], the performance of MKLCKF: II had been verified. Since MKLCKF:I is computationally expensive, we restricted the evaluation to only MKLCKF:II. Caltech101 dataset consists of 101 classes and a total of 3131 images where as UIUC dataset consists of 1579 images. In this case, we followed 5-fold cross validatstion for comparing the performance of the models.

Deep Convolutional Neural Network [83] had been used for extracting the features. We used a pretrained CNN which was trained over imagenet from [84]. As the CNN

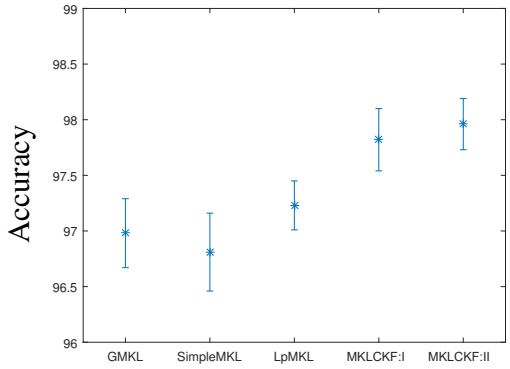

Figure 6.4: Accuracy Analysis using Ring Norm Data: MKLCKF and other state-of-the-art MKL models

was trained over whole imagenet, a feature selection algorithm had been applied over the features extracted from CNN, that is, the penultimate layer output of the CNN was fed to a feature selection algorithm. The feature selection algorithm, we used was random forest using scikit learn library [85]. .

The result over Caltech101 is shown in the Fig. 6.5 while the result for the experiment on UIUC sports scene classification is shown in Fig. 6.6. The MKLCKF:II performed well on image data.

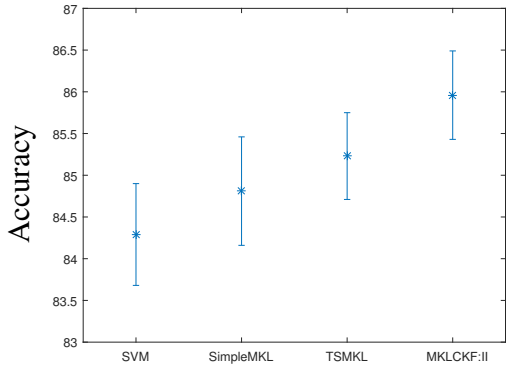

Figure 6.5: Accuracy: Caltech101 Image Classification using MKLCKF

6.2.4 Application in Kernelized Locally-Sensitive Hashing

We chose an application domain also for assessing the performance of the models. Using MKLCKF: II we performed kernelized locally sensitive hashing (KLSH) [86] on image data described in section 6.2.3. KNN based approach was used for evaluating the perfor-

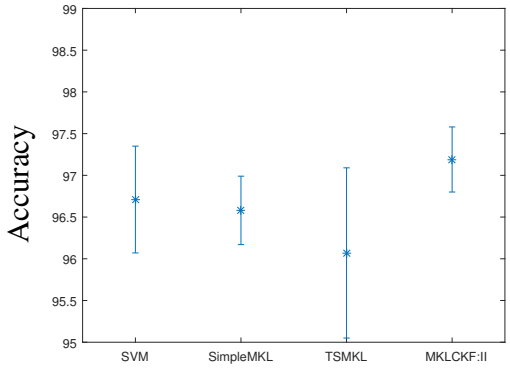

Figure 6.6: Accuracy: UIUC Sports Scene Classification using MKLCKF

mance of KLSH.

We compared the proposed model MKLCKF:II with single kernel approach and TSMKL approach ([1] only as other models need theoretical modification for applying on KLSH.

The accuracy results for the experiments are shown in Fig. 6.7 and Fig. 6.8. The graph plots the mean accuracy and its standard deviation of different models over 30 iterations of hold out approach. The superior performance of MKLCKF:II is evident from these figures.

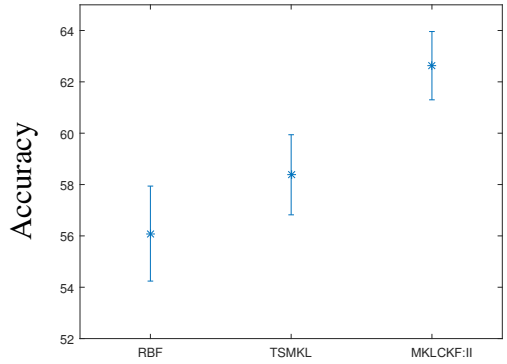

Figure 6.7: Accuracy: Caltech101 Image Classification using KNN-KLSH

6.2.5 Regression Experiments

Regression experiments are also carried over *large* data as well as *not large* data. In case of regression experiments over large data, the preclustering algorithm for data compression, described in section 2.4.1 is used. The hyperparameters of the preclustering algorithm are the ϵ value and step length h. In order to optimize the ϵ value, we took a small part of the data from the dataset and applied cross validation over different values of ϵ from

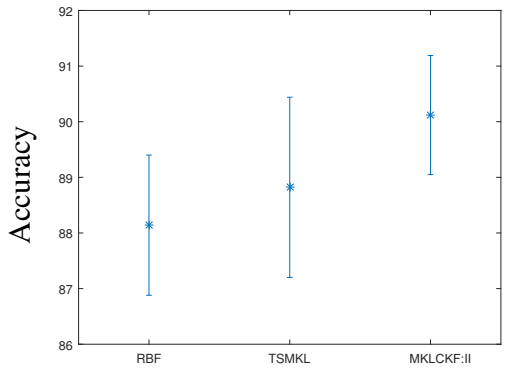

Figure 6.8: Accuracy: UIUC Sports Scene Classification using KNN-KLSH

$\{0.1, 0.11, ...1\}$. Based on the mean squared error performance as well as compression rate for each ϵ value, we chose the ϵ which gives comparable MSE (within a threshold range) and maximum compression. Rather than picking up a small subset from the dataset randomly, we constructed a ball $\mathcal{B}(p, r)$, and the points coming in this ball are taken as subset. In this ball, p is any random point and r is the radius of the ball. We adjusted the radius value in such a way to get an optimum number of data points in the subset. The optimum value of h is also found using a subset of the data.

6.2.5.1 Results and Discussions

The performance of the algorithms on *'not large data'* was assessed with all points and pre clustered points. The performance of MKLCKF is comparable with that of conventional MKL and single kernel SVR on *'not large data'* (Table 6.5).

Table 6.5: RMSE using all Data Points

Dataset	SVR	Simple MKL (SVR)	MKLCKF:I (SVR)	MKLCKF:II (SVR)
Airfoil Self Noise	3.83287 $\pm$ 0.20978	3.40593 $\pm$ 0.32411	3.53291 $\pm$ 0.29307	3.47891 $\pm$ 0.31476
Commun. and Crime	5.79657 $\pm$ 0.29028	5.86840 $\pm$ 0.29237	5.80437 $\pm$ 0.31056	5.81379 $\pm$ 0.31648
Concrete Slump Test	6.48337 $\pm$ 0.45852	6.09983 $\pm$ 0.52536	6.16865 $\pm$ 0.38802	6.08913 $\pm$ 0.36248
Energy Eff. Cool	1.33792 $\pm$ 0.10755	1.23957 $\pm$ 0.10164	1.25763 $\pm$ 0.10176	1.24912 $\pm$ 0.11345
Energy Eff. Heat	2.40471 $\pm$ 0.21294	1.40673 $\pm$ 0.13337	1.34312 $\pm$ 0.14548	1.36918 $\pm$ 0.13421

We analysed *'not large data'* using preclustering approach also, Fig. 6.9 shows the level of compression achieved using pre-clustering algorithm which is calculated on the basis of the average of the number of pre-clustered points obtained in each iteration during hold out validation, while Table 6.6 shows the RMSE of each model with pre-clustered points. The

results in Table 6.6 and 6.5 are comparable and therefore pre-clustering approach helped to compress the data in a very effective manner. Also it is clear from the figures that with the aid of pre-clustering the MKLCKF models showed better results than others.

Table 6.6: RMSE using Pre-clustered Data Points

Dataset	SVR	SimpleMKL (SVR)	MKLCKF:I (SVR)	MKLCKF:II (SVR)
Airfoil Self Noise	4.14588 ± 0.30715	3.96222 ± 0.39076	3.52020 ± 0.20978	3.49316 ± 0.21016
Commun. and Crime	6.45198 ± 0.49322	6.53542 ± 0.49109	6.16050 ± 0.45235	6.09381 ± 0.41351
Concrete Slump Test	6.54374 ± 0.42661	6.35594 ± 0.55881	6.12909 ± 0.43515	6.13249 ± 0.41277
Energy Eff. Cool	2.56802 ± 0.16839	2.54866 ± 0.19781	2.41040 ± 0.18034	2.29346 ± 0.17199
Energy Eff. Heat	2.59222 ± 0.22298	2.57326 ± 0.22317	2.47445 ± 0.18807	2.26533 ± 0.16736

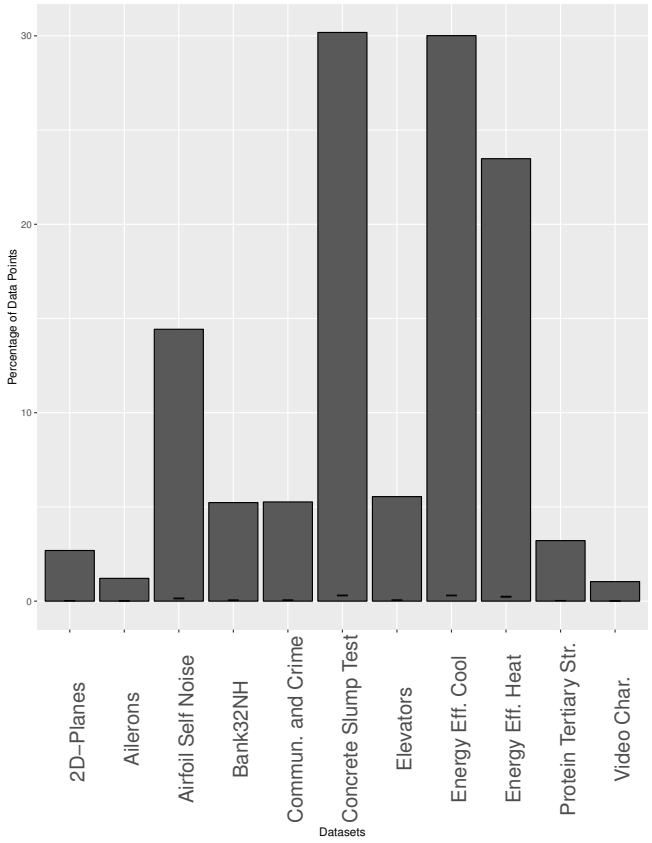

Figure 6.9: Data Compression Ratio using Pre-clustering for MKLCKF

6.2.5.2 Analysis on Large Data Sets

We analysed the performance on large regression data sets by using the data compression approach alone since other appraches are failing with out-of-the-memory error. With the

aid of pre-clustering we successfully applied the proposed models over large data sets. For all the large data sets we used, MKLCKF models showed a superior performance over other models (Fig. 6.10). This is due to the fact that pre-clustering helps to remove the redundant points and hence the proposed model is more efficient with the informative data points.

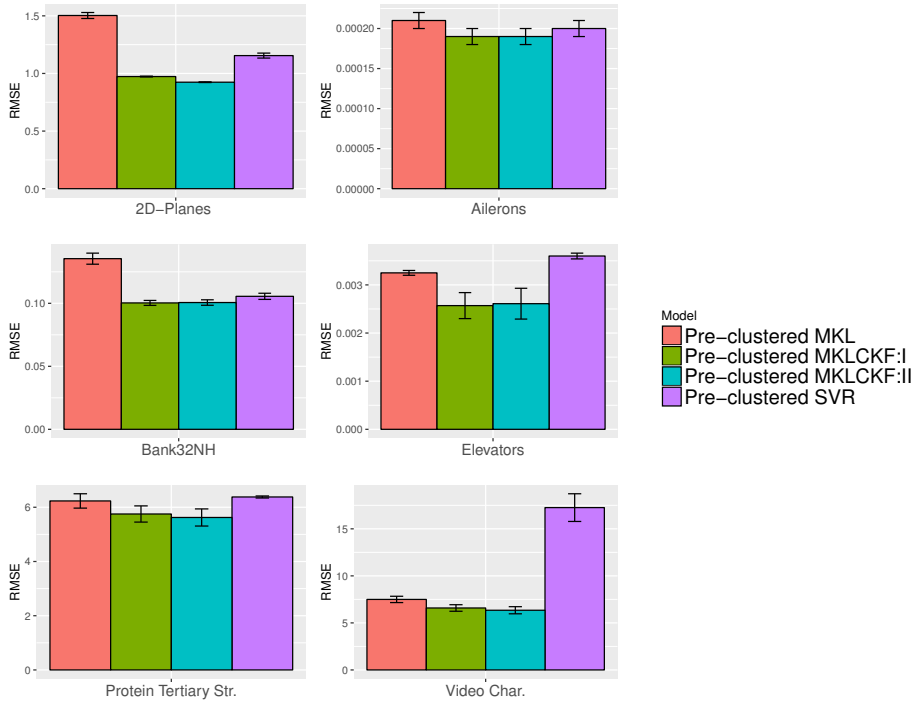

Figure 6.10: Root Mean Squared Error for large Datasets over all Models

6.3 Conclusion

The MKLCKF, uses composite kernel functions for MKL. Using each data as reference, composite kernels are designed in such a way that each of those consists of two functions in which the first function makes use of all the P base kernels under consideration and the second is a single valid kernel function. The optimal kernel is then represented as a linear combination of N kernels. Two versions of MKLCF are formulated and assessed their performance on classification as well as regression problems. For applying on large data sets, to avoid the 'out of memory' error, we have discussed the supervised pre-clustering for regression and data dictionary methods for classification for finding the vital points. The models we introduced showed a superior performance on large data sets in comparison with

other state of art techniques. In the next chapter, we discuss the real world applications of the MKL models we developed.

Chapter 7

Real World Applications

The MKL architectures were applied on two real world applications. A detailed study of their performance is given in this chapter. The applications were chosen from the areas of demand sensing and news classification. The experiments we chose were dealt with the sales forecasting of Mercedes C class and E class in USA. The ARIMA model was used for the time series forecasting and the generated values were integrated with social media sentiment score as well as news media sentiment score from data tweets and news for improving the forecasting. Those sentiment scores were collected using crimson hexagon API. The problem was formulated as a regression problem.

The second application described in this chapter is about the classification of public news that publishes on various web sites. It consisted of 3 different classification tasks, namely, the task that related with the classification of sectors, technology and startup related news. The deep learning was used to generate the features of the data. The MKL models we developed as well as the state-of-the-art MKL models and LSTM based model were used for analysis. LSTM model is designed as a two layered neural network with first layer is an LSTM layer with 32 hidden units and second layer is a fully connected dense network with single output neuron. The loss function used is mean absolute error.

7.1 Demand Sensing for Mercedes cars sales prediction

The task is to forecast the sales of Mercedes C class and E class in USA for the year, July 2016 to June 2017 using the sales data from Jan 2008 till Jun 2016 on a monthly basis. Conventionally, the sales prediction is implemented through statistical methods. Such techniques do not take into account the immediate and near future fluctuations in the related market factors. Hence we made use of the demand sensing techniques. The demand sensing deals with the correction of the long term forecasts using near real time

information([87]).

The statistical methods such as ARIMA which are effective in finding out overall prediction has to be enhanced with additional features to capture near future fluctuations. For this purpose, in our analysis, social media sentiment and news media sentiment scores were generated by hexagon API from the data tweets and news. On the basis of those scores, the social media news and tweets were divided into positive, negative & neutral and those information were incorporated with the ARIMA data. That is, for a training point (x_t, y_t), y_t represents the actual sale for the time t and x_t consists of the following attributes:

- The sales forecasted by the ARIMA model for the time t. For forecasting, the ARIMA model made use of the data collected from three consecutive years prior to that of t.

- The following information collected from the previous month of time t: average of positive news; average of neutral news; average of negative news; average of positive tweets; average of neutral tweets; average of negative tweets.

The machine learning model has to predict the sales at time t. The process described above is summarized in the block diagram of demand sensing architecture (Fig. 7.1).

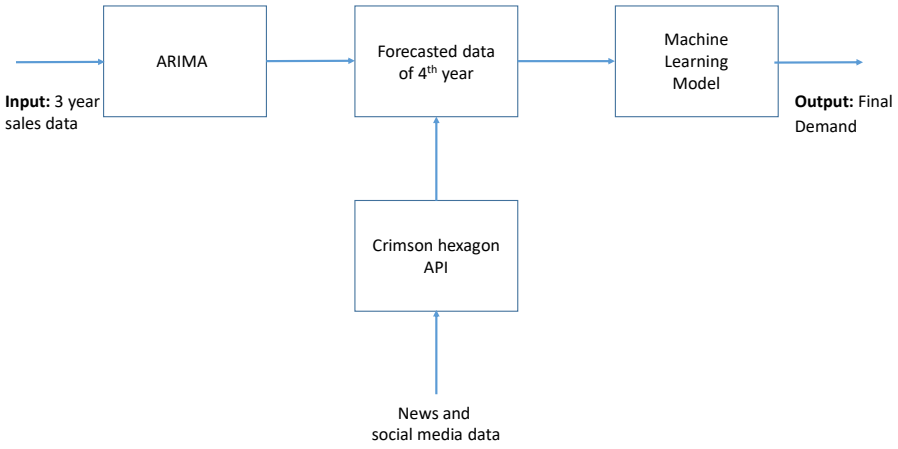

Figure 7.1: Demand Sensing Architecture: Block Diagram

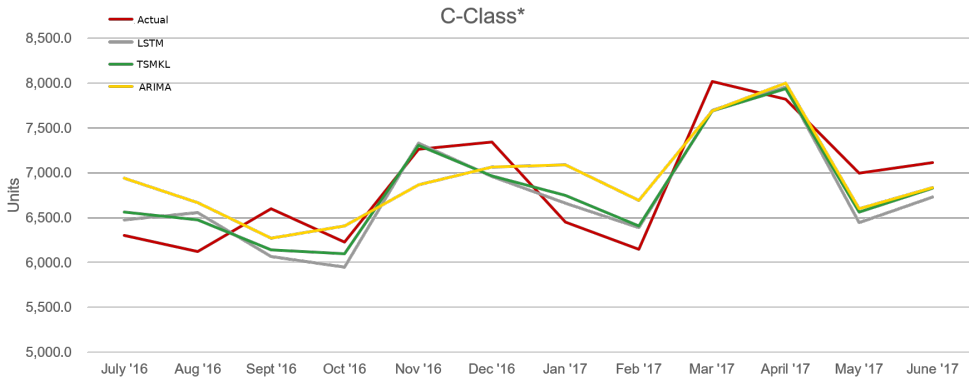

Figure 7.2: Monthly prediction for C-Class

7.1.1 Results and Discussion

The data was analyzed using the existing models, namely, ARIMA, SVR, SimpleMKL, SPG-MKL, LSTM and the models we developed, namely, TSMKL, MKLCKF:II. The performance measure used was RMSE value. The results are tabulated in Table 7.1. The detailed monthly wise predictions for C-Class are given in the Fig. 7.2 and for E-Class in the Fig. 7.3. The TSMKL showed the best performance followed by MKLCKF:II.

Table 7.1: Root Mean Squared Error in Sales Forecasting

Models	**C-Class**	**E-Class**
ARIMA	423.92	645.61
SVR	401.45	627.81
SimpleMKL	372.93	618.38
SPG-MKL	332.97	559.34
LSTM	357.61	608.16
TSMKL	**303.50**	**533.71**
MKLCKF:II	325.76	542.87

7.2 News Classification

The task is about classifying news as that related with sectors, technology, organization etc. It is very much essential for the organizations like technology firms to get updated themselves with the news related with start-ups, their competitors and those that of the clients of the company, as such information help to plan the strategies related with matters like profit making. However there exist millions of websites which carry useful information

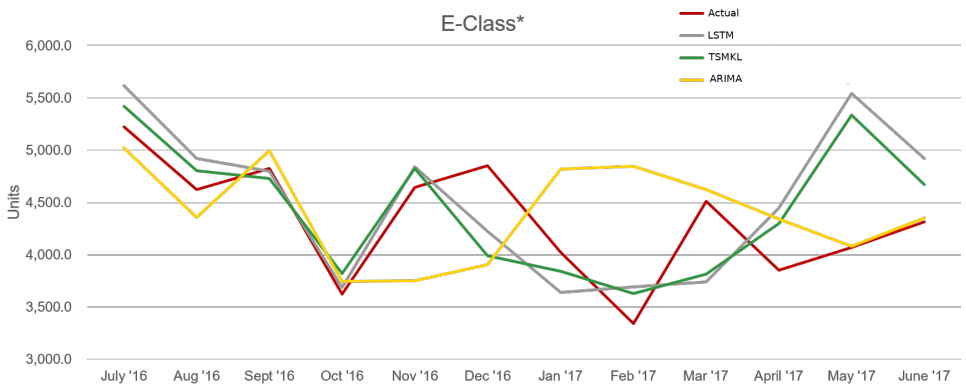

Figure 7.3: Monthly prediction for E-Class

and hence one has to do real time extraction and classification of all the information into meaningful visualizations, which makes the task challenging as it involves a number of tasks as listed below.

- From the large collection of web pages available, the relevant pages have to be identified using factors like popularity, rating and interests of the websites.
- Considering the information explosion happening around, it is crucial to prioritize the information and identify the key trends happening.
- The selected news materials have to go through an analytical stage where it has to be checked whether the news agree with current trends or not.
- The information identified has to be properly summarized for drawing conclusions. Such a condensed form helps the management people to get the critical insights of the scenario under consideration with minimum time and energy.

7.2.1 Classification tasks

The problem under study comprises of three main classification tasks as given below:

1. Sector classification
   The objective in this task is to classify the news into following sectors: Banking and Capital Marketing (BCM), Customer Products and Retail (CPR), Oil and Gas (O & G), Wealth and Asset Management (WAM), Supply Chain and Operations (SCO), Power and Utilities (P & U), Goverment and Public Sectors (GPS) and General.

2. Technology classification

   The subclasses involved in this task are Artificial Intelligence, Internet of Things, Blockchain, Additive Manufacturing, Robotics and Mixed Reality.

3. Organization Classification

   In this task, news has to be classified on the basis of Startups versus Other Organizations.

7.2.2 Experimental Settings

Two different approaches were used for analysing the news data, whose discussion is given below.

1. LSTM approach

   This approach consisted of the following steps. Each sentence was converted into a vector representation. Skip-Thought vectors were used for this purpose. Skip-thought vectors are the encoded vectors generated by the model skip-thoughts, where skip-thought is an encoder-decoder model in which the encoder maps the input sentence to a sentence vector and the decoder generates the sentences surrounding the original sentence [88]. Thus each news article can be considered as sequence of skip-thought vectors. These sequences were then fed into a LSTM (Long Short Term Memory) architecture where LSTM is an advanced neural network architecture that can capture the recurrent relations involved in the data.

2. Shallow learning

   For applying shallow learning techniques, Doc2Vec ([89]) method was used for creating a vector representation of each news article. Doc2Vec takes a sequence of sentences (paragraphs, news etc.) as input and map this input to a vector representation. Thus it create a numeric representation of a document, regardless of its length.

   In our analysis, Doc2Vec was implemented in the following way: a neural network model which was trained over wiki-dataset was again retrained over the given news data set for creating a more robust model and it was used for generating vectors corresponding to each news. The shallow learning techniques were applied on the resulting data.

7.2.3 Results and Discussions

The algorithms used for the analysis were SVM, GMKL, LpMKL, LSTM, FSSMKL, MKLCKF:I and MKLCKF:II. The results are tabulated in the Table 7.2. The best results were obtained for FSSMKL. Eventhough the LSTM based deep learning architecture is able to capture the hidden features, due to lack of enough training points (800 points), the MKLCKF architectures produced better results in our analysis.

Table 7.2: Accuracy Table for News Classification

	Sector	Technology	Organization
SVM	82.45 $\pm$ 2.35	87.81 $\pm$ 3.84	72.93 $\pm$ 5.45
GMKL	85.39 $\pm$ 3.59	88.54 $\pm$ 3.78	73.23 $\pm$ 5.84
LpMKL	87.36 $\pm$ 2.71	89.38 $\pm$ 3.74	74.23 $\pm$ 5.98
LSTM	87.23 $\pm$ 2.78	88.34 $\pm$ 3.82	74.96 $\pm$ 5.68
FSSMKL	**90.12 $\pm$ 2.18**	**91.98 $\pm$ 3.23**	**77.83 $\pm$ 5.27**
MKLCKF:I	88.18 $\pm$ 2.09	89.92 $\pm$ 3.37	75.19 $\pm$ 5.83
MKLCKF:II	89.79+-2.82	90.23 $\pm$ 3.65	76.70 $\pm$ 5.46

Chapter 8

Conclusion & Future Work

The multiple kernel learning deals with the selection of optimum kernel for kernel algorithms. We developed four different approaches for MKL. In the first approach, the data feature's intrinsic properties are made use for finding the optimal kernel. For that the data features are subjected to clustering and a kernel is assigned to each cluster. The linear combination of such kernels is used for representing the optimal kernel.

In the second approach, the MKL is represented as a supervised classification problem, in which the problem of finding the optimal kernel and the decision function related with the data, is formulated as a single approximation problem. For that it is assumed that the solution of the approximation problem lies in a RKHS space, which is constructed as the direct sum of RKHSs related with that of the unknown functions under consideration. This formulation enables the kernel and classification function learning phases to associate with each other. This approach is suitable for binary classification problems only.

The MKL is formulated as a regression problem also by adapting two stage optimization frame work for analysing regression data. For that it is proved that ideal kernel for regression is $k(x_i, x_j) = y_i y_j$. The supervised pre-clustering approach is used to select the vital data points for reducing the overall time and space complexity. Using the concepts of Krein space, nonlinear MKL for regression is formulated.

The concept of composite kernel functions has also been used for formulating MKL. Using each data point as a reference point, a composite kernel is constructed such that it makes use of all the P base kernels under construction. The optimal kernel is constructed as the linear combination of such composite kernels. The framework is formulated for classification as well as regression problems.

The comparative study of the performance of the models has been done by applying them on two real world sets.

8.1 Future Work

We have made contributions to the field of multiple kernel learning. There is very much scope for further work in this domain. Given below is the description of the areas where the developed work can be extended.

8.1.1 Feature wise combination of kernels

The feature wise kernel combination uses a cluster based approach for the kernel formulation. A function approximation approach can be introduced for automatically assigning features to appropriate kernels.

8.1.2 Multi-layered kernel learning

The single stage function approximation based binary classification can be made more powerful by incorporating faster optimization techniques. A multi layered, deep like architecture can be implemented for function approximation based MKL.

8.1.3 Ideal kernel in regression domain

The ideal kernel we formulated for kernel ridge regression can be used for developing ideal kernel based target techniques for regression problems.

8.1.4 Nonlinear kernel learning in Krein space for classification problems

Using Krein space concepts, the representation of optimal kernel using nonlinear combination of kernels, can be developed for classification based MKL techniques.

Bibliography

[1] A. Kumar, A. Niculescu-Mizil, K. Kavukcuoglu, and H. Daume, III, "A Binary Classification Framework for Two-Stage Multiple Kernel Learning," *ArXiv e-prints*, Jun. 2012.

[2] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[3] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[5] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American Mathematical Society*, vol. 68, no. 3, pp. 337–404, 1950.

[6] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92. New York, NY, USA: ACM, 1992, pp. 144–152. [Online]. Available: http://doi.acm.org/10.1145/130385.130401

[7] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998. [Online]. Available: http://dx.doi.org/10.1162/089976698300017467

[8] A. Pozdnoukhov, "The analysis of kernel ridge regression learning algorithm." IDIAP, Martigny, Switzerland, Idiap-RR Idiap-RR-54-2002, 0 2002.

[9] R. Chitta, R. Jin, T. C. Havens, and A. K. Jain, "Approximate kernel k-means: Solution to large scale kernel clustering," in *Proceedings of the 17th ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, ser. KDD '11. New York, NY, USA: ACM, 2011, pp. 895–903.

[10] D. Tax and P. Juszczak, "Kernel whitening for one-class classification," in *Pattern Recognition with Support Vector Machines*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, vol. 2388, pp. 40–52.

[11] G. Mehmet and A. Ethem, "Multiple kernel learning algorithms," *Journal of Machine Learning and Research*, vol. 12, pp. 2211–2268, Jul. 2011.

[12] P. Pavlidis, J. Weston, J. Cai, and W. N. Grundy, "Gene functional classification from heterogeneous data," in *Proceedings of the Fifth Annual International Conference on Computational Biology*, ser. RECOMB '01. New York, NY, USA: ACM, 2001, pp. 249–255.

[13] I. M. de Diego, A. Muñoz, and J. M. Moguerza, "Methods for the combination of kernel matrices within a support vector framework," *Machine Learning*, vol. 78, no. 1, p. 137, Aug 2009. [Online]. Available: https://doi.org/10.1007/s10994-009-5135-5

[14] S. Qiu and T. Lane, "A framework for multiple kernel support vector regression and its applications to sirna efficacy prediction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 6, no. 2, pp. 190–199, April 2009.

[15] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semi-definite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[16] J. Kandola, J. Shawe-Taylor, and N. Cristianini, "Optimizing kernel alignment over combinations of kernels," Department of Computer Science, Royal Holloway, University of London, UK, Tech. Rep. 121, 2002.

[17] C. Igel, T. Glasmachers, B. Mersch, N. Pfeifer, and P. Meinicke, "Gradient-based optimization of kernel-target alignment for sequence kernels applied to bacterial gene start detection," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 4, no. 2, pp. 216–226, Apr. 2007.

[18] S. Yu, L.-C. Tranchevent, B. D. Moor, and Y. Moreau, *Kernel-based Data Fusion for Machine Learning*. Springer Berlin Heidelberg, 2011, vol. 345.

[19] A. Rakotomamonjy, F. R. Bach, S. Canu, , and Y. Grandvalet, "Simple MKL," *Journal of Machine Learning Research*, vol. 9, pp. 2491–2521, 2008.

[20] M. Gonen and E. Alpaydin, "Localized algorithms for multiple kernel learning," *Pattern Recognition*, vol. 46, no. 3, pp. 795 – 807, 2013.

[21] M. Varma and B. Babu, "More generality in efficient multiple kernel learning," in *Proceedings of the International Conference on Machine Learning*, June 2009, pp. 1065–1072.

[22] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor, "On kernel-target alignment," in *Advances in Neural Information Processing Systems 14*. MIT Press, 2002, pp. 367–373.

[23] T. Damoulos and M. A. Girolami, "Probabilistic multi-class multi-kernel learning: on protein fold recognition and remote homology detection," *Bioinformatics*, vol. 10, pp. 1264–1270, 2008.

[24] K. P. Bennett, M. Momma, and M. J. Embrechts, "Mark: A boosting algorithm for heterogeneous kernel models," in *Proceedings KDD-2002: Knowledge Discovery and Data Mining*, 2002, pp. 24–31.

[25] S. Wang, Q. Huang, S. Jiang, and Q. Tian, "S^3MKL: Scalable semi-supervised multiple kernel learning for real-world image applications," *IEEE Transactions on Multimedia*, vol. 14, no. 4, pp. 1259–1274, 2012.

[26] C. Hsu and W. S. Lee, Eds., *Proceedings of the 3rd Asian Conference on Machine Learning, 2011, Taoyuan, Taiwan, November 13-15, 2011*, ser. JMLR Proceedings, vol. 20. JMLR.org, 2011. [Online]. Available: http://jmlr.org/proceedings/papers/v20/

[27] X. Liu, L. Wang, J. Yin, Y. Dou, and J. Zhang, "Absent multiple kernel learning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15. AAAI Press, 2015, pp. 2807–2813. [Online]. Available: http://dl.acm.org/citation.cfm?id=2886521.2886712

[28] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04. ACM, 2004, pp. 6–.

[29] S. Sonnenburg, G. Ratsch, C. Schafer, and B. Scholkopf, "Large scale multiple kernel learning," *Journal of Machine Learning Research*, vol. 7, pp. 1531–1565, 2006.

[30] C. Cortes, M. Mohri, and A. Rostamizadeh, "Learning non-linear combinations of kernels," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds., 2009, pp. 396–404.

[31] X. Liu, L. Wang, J. Zhang, and J. Yin, "Sample-adaptive multiple kernel learning," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 2014, pp. 1975–1981.

[32] Y.-R. Yeh, T.-C. Lin, Y.-Y. Chung, and Y.-C. Wang, "A novel multiple kernel learning framework for heterogeneous feature fusion and variable selection," *Multimedia, IEEE Transactions on*, vol. 14, no. 3, pp. 563–574, June 2012.

[33] A. D. Dileep and C. Sekhar, "Representation and feature selection using multiple kernel learning," in *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, June 2009, pp. 717–722.

[34] S. S. Nair and T. J. Dodd, "Supervised pre-clustering for sparse regression," *International Journal of Systems Science*, vol. 46, no. 7, pp. 1161–1171, 2015.

[35] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent K-SVD: Learning a discriminative dictionary for recognition," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 35, no. 11, pp. 2651–2664, 2013.

[36] T. Dodd, S. Nair, and R. Harrison, "Effect of the order of parameterisation in gradient learning for kernel methods," *IET Control Theory and Applications*, vol. 4, pp. 2141–2151, Oct 2009.

[37] A. Argyriou, C. A. Micchelli, and M. Pontil, "When is there a representer theorem? vector versus matrix regularizers," *CoRR*, vol. abs/0809.1590, 2008.

[38] Platt and J. C., "Advances in kernel methods." Cambridge, MA, USA: MIT Press, 1999, ch. Fast Training of Support Vector Machines Using Sequential Minimal Optimization, pp. 185–208.

[39] V. Tresp, "Scaling kernel-based systems to large data sets," *Data Mining and Knowledge Discovery*, vol. 5, no. 3, pp. 197–211, 2001. [Online]. Available: /papers/upload_9187_kdd_journal2.ps

[40] R. Collobert, S. Bengio, and C. Williamson, “SVMTorch: Support vector machines for large-scale regression problems,” *Journal of Machine Learning Research*, vol. 1, pp. 143–160, 2001.

[41] C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis, “Improved fast Gauss transform and efficient kernel density estimation,” in *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ser. ICCV ’03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 464–. [Online]. Available: http://dl.acm.org/citation.cfm?id=946247.946593

[42] Y. Li, L. Yang, and J. Ding, “A minimum enclosing ball-based support vector machine approach for detection of phishing websites,” *Optik - International Journal for Light and Electron Optics*, vol. 127, no. 1, pp. 345 – 351, 2016.

[43] R. Strack, V. Kecman, B. Strack, and Q. Li, “Sphere support vector machines for large classification tasks,” *Neurocomput.*, vol. 101, pp. 59–67, Feb. 2013.

[44] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, “Core vector machines: Fast SVM training on very large data sets,” *J. Mach. Learn. Res.*, vol. 6, pp. 363–392, Dec. 2005.

[45] G. Loosli and S. Canu, “Comments on the "core vector machines: Fast SVM training on very large data sets",” *J. Mach. Learn. Res.*, vol. 8, pp. 291–301, May 2007.

[46] D. Pavlov, D. Chudova, and P. Smyth, “Towards scalable support vector machines using squashing,” in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’00. New York, NY, USA: ACM, 2000, pp. 295–299. [Online]. Available: http://doi.acm.org/10.1145/347090.347155

[47] M. Aharon, M. Elad, and A. Bruckstein, “SVDD: An algorithm for designing overcomplete dictionaries for sparse representation,” *Trans. Sig. Proc.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[48] K. Skretting and K. Engan, “Recursive least squares dictionary learning algorithm,” *IEEE Transactions on Signal Processing*, vol. 58, no. 4, pp. 2121–2130, April 2010.

[49] F. R. Bach, “Consistency of the group lasso and multiple kernel learning,” *Journal of Machine Learning Research*, vol. 9, pp. 1179–1225, Jun. 2008.

[50] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu, "Simple and efficient multiple kernel learning by group lasso." in *ICML*. Omnipress, 2010, pp. 1175–1182.

[51] I. S. Dhillon, S. Mallela, and R. Kumar, "A divisive information theoretic feature clustering algorithm for text classification," *J. Mach. Learn. Res.*, vol. 3, pp. 1265–1287, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944973

[52] H. Al-Mubaid and S. A. Umair, "A new text categorization technique using distributional clustering and learning logic," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 9, pp. 1156–1165, 2006.

[53] F. Rossi, D. François, V. Wertz, M. Meurens, and M. Verleysen, "Fast selection of spectral variables with b-spline compression," *CoRR*, vol. abs/0709.3639, 2007. [Online]. Available: http://arxiv.org/abs/0709.3639

[54] C. Scherrer, A. Tewari, M. Halappanavar, and D. Haglin, "Feature clustering for accelerating parallel coordinate descent," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 28–36.

[55] F. Bonchi, A. Gionis, and A. Ukkonen, "Overlapping correlation clustering," in *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ser. ICDM '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 51–60.

[56] A. Banerjee, C. Krumpelman, and J. Ghosh, "Model-based overlapping clustering," in *In KDD*. ACM Press, 2005, pp. 532–537.

[57] J.-Y. Jiang, R.-J. Liou, and S.-J. Lee, "A fuzzy self-constructing feature clustering algorithm for text classification," *IEEE Trans. on Knowl. and Data Eng.*, vol. 23, no. 3, pp. 335–349, Mar. 2011.

[58] D. Picard, N. Thome, and M. Cord, "Jkernelmachines - a simple framework for kernel machines," *Journal of Machine Learning Research*, vol. 14, pp. 1417–1421, Mar. 2013.

[59] ——, "Jkernelmachines," 2013. [Online]. Available: http://mloss.org/software/view/409/

[60] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html

[61] S. Sonnenburg, C. S. Ong, S. Henschel, and M. Braun, "mldata.org," 2011. [Online]. Available: https://mldata.org/repository/data/

[62] J. C. Bezdek, "A convergence theorem for the fuzzy ISODATA clustering algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, pp. 1–8, 1980.

[63] Y. Lecun and C. Cortes, "The MNIST database of handwritten digits," 2009. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[64] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[65] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *In CVPR*, 2005, pp. 886–893.

[66] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*, ser. COLT '01/EuroCOLT '01. London, UK, UK: Springer-Verlag, 2001, pp. 416–426. [Online]. Available: http://dl.acm.org/citation.cfm?id=648300.755324

[67] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online Learning with Kernels," *IEEE Transactions on Signal Processing*, vol. 52, pp. 2165–2176, August 2004.

[68] Y. Gu, T. Liu, X. Jia, J. A. Benediktsson, and J. Chanussot, "Nonlinear multiple kernel learning with multiple-structure-element extended morphological profiles for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 6, pp. 3235–3247, June 2016.

[69] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien, "Lp-norm multiple kernel learning," *Journal of Machine Learning Research*, vol. 12, pp. 953–997, Jul. 2011.

[70] Q. Wang, Y. Gu, and D. Tuia, "Discriminative multiple kernel learning for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 7, pp. 3912–3927, July 2016.

[71] S. Hoi, R. Jin, P. Zhao, and T. Yang, "Online multiple kernel classification," *Machine Learning*, vol. 90, no. 2, pp. 289–316, 2013.

[72] B. Babu and M. Varma, June 2009. [Online]. Available: http://research.microsoft.com/en-us/um/people/manik/code/GMKL/gmkl.tgz

[73] J. Kivinen, A. J. Smola, and R. C. Williamson, August 2004. [Online]. Available: http://research.larc.smu.edu.sg/MLG/omkc/OMKC_package_1.0(code).zip

[74] X. Xu, May 2016. [Online]. Available: https://sites.google.com/site/xinxingxu666/LibMKL_14-05-06.rar?attredirects=0

[75] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal estimated sub-gradient solver for SVM," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML 2007. New York, NY, USA: ACM, 2007, pp. 807–814.

[76] G. Loosli, S. Canu, and C. S. Ong, "Learning SVM in Krein spaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 6, pp. 1204–1216, June 2016.

[77] C. S. Ong, X. Mary, S. Canu, and A. J. Smola, "Learning with non-positive kernels," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04. New York, NY, USA: ACM, 2004, pp. 81–. [Online]. Available: http://doi.acm.org/10.1145/1015330.1015443

[78] A. Jain, S. V. N. Vishwanathan, and M. Varma, "SPG-GMKL: Generalized multiple kernel learning with a million kernels," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, August 2012.

[79] M. Varma and B. Babu, June 2009. [Online]. Available: http://asi.insa-rouen.fr/enseignants/~arakoto/code/mklindex.html

[80] M. Varma and B. Babu, R, June 2009. [Online]. Available: http://www.cs.cornell.edu/~ashesh/pubs/code/SPG-GMKL/download.html

[81] F.-F. Li, M. Andreetto, and M. A. Ranzato, "Caltech101 image dataset," 2003. [Online]. Available: http://www.vision.caltech.edu/Image_Datasets/Caltech101/

[82] L.-J. Li and L. Fei-Fei, "Uiuc sports event dataset," *IEEE Intern. Conf. in Computer Vision (ICCV)*, 2007. [Online]. Available: http://vision.stanford.edu/lijiali/event_dataset/

[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[84] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," http://www.vlfeat.org/matconvnet/pretrained/, 2008.

[85] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[86] K. Grauman and B. Kulis, "Kernelized locality-sensitive hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. undefined, pp. 1092–1104, 2011.

[87] D. Raza and P. Kilbourn, "The impact of point-of-sale data in demand planning in the south african clothing retail industry," *Journal of Transport and Supply Chain Management*, vol. 11, no. 0, p. 8, 2017. [Online]. Available: https://jtscm.co.za/index.php/jtscm/article/view/304

[88] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler, "Skip-thought vectors," *CoRR*, vol. abs/1506.06726, 2015.

[89] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, pp. II–1188–II–1196.

List of Publications

Publications

1. Shiju S.S., Salim A., Sumitra S. (2017) *Formulation of Two Stage Multiple Kernel Learning Using Regression Framework.*, Pattern Recognition and Machine Intelligence. PReMI 2017. Lecture Notes in Computer Science, vol 10597. Springer, Cham. ***[Received Best Student Presentation Award from Springer]***

2. Shiju S. S., Asif Salim, S. Sumitra, *Multiple Kernel Learning Algorithm using Composite Kernel Functions (2017)*, Engineering Applications of Artificial Intelligence 64, 391-400.

3. Shiju S. S and Sumitra S. *Multiple Kernel Learning using Single Stage Function Approximation for Binary Classification Problems (2017).* International Journal of Systems Science 48(16): pp. 3569-3580.