GRAPH BASED OPTIMIZATION, FEATURE SELECTION AND LEARNING

Thesis submitted by

Pranjal Rai (2018EE10484)
Shauryasikt Jena (2018EE10500)
Tanvir Singh Bal (2018EE10508)

under the guidance of

Prof. Sandeep Kumar, IIT Delhi
Prof. Jayadeva, IIT Delhi

in partial fulfilment of the requirements
for the award of the degree of

Bachelor of Technology

Department Of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY DELHI
November 2021

THESIS CERTIFICATE

This is to certify that the thesis titled **Graph based optimization, feature selection and learning**, submitted by **Pranjal Rai (2018EE10484), Shauryasikt Jena (2018EE10500) and Tanvir Singh Bal (2018EE10508)**, to the Indian Institute of Technology, Delhi, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by them under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Sandeep Kumar
Assistant Professor
Dept. of Electrical Engineering
IIT-Delhi, 110 016

Prof. Jayadeva
Professor
Dept. of Electrical Engineering
IIT-Delhi, 110 016

Place: New Delhi
Date: 23rd November 2021

ACKNOWLEDGEMENTS

We would like to thank Prof. Sandeep Kumar and Prof. Jayadeva for guiding us throughout the project and helping us understand the principles of theoretical Machine learning and optimization. The insights given by Prof. Kumar and Prof. Jayadeva ensured that we moved on the right direction. Prof. Jayadeva's innovative state of the art research problems and novel applications were truly an inspiration for us to pursue a career in research.

The project provided us a great opportunity to understand the world of theoretical Machine learning and optimization, along with directly contributing to the research field.

We would also like to pay our gratitude to Prof. Sumeet Agrawal, who critically appraised the project thesis and provided valuable suggestions. Lastly, we all are grateful to our families for their constant support and motivation throughout.

ABSTRACT

KEYWORDS: Semi-supervised learning; Manifold regularization; Minimizing the VC dimension, Trend filtering, Feature selection, Pattern recognition, Optimization

Limited labelled data is a big roadblock in practical applications of Machine learning models. Semi-supervised learning solves this problem by learning from both the labelled as well as unlabelled data. One of the popular framework to learn from unlabelled data is Manifold regularization, which tries to incorporate the intrinsic geometry of the marginal distribution by introducing a Graph-Laplacian based penalty term in the objective. The present work employs the Manifold Regularization framework to Minimal Complexity Machines thus, formulating a novel semi-supervised learning algorithm called, Laplacian MCM. Further, a novel variation of Laplacian MCM is proposed based on Trend filtering which results in a L-1 norm penalty term. The work also proposes Laplacian Twin SVM for learning from skewed datasets, along with providing novel application of LapMCM in the form of Feature selection.

Contents

List of Tables

List of Figures

ABBREVIATIONS

MCM	Minimal Complexity Machines
Lap MCM	Laplacian Minimal Complexity Machines
TF MCM	Trend Filtered Minimal Complecity Machines

NOTATION

n	Total number of samples in the dataset
l	Total number of labelled samples in the dataset
u	Total number of unlabelled samples in the dataset
W	Weight matrix of the graph
L	Laplacian matrix of the graph
Δ	Graph Difference Operator (GDO) such that, $\Delta^T\Delta = L$
K	The Gram matrix

Chapter 1

Introduction

1.1 Motivation

In today's world supervised algorithms have an inherent disadvantage that they require massive amounts of labeled data to give accurate results. Thus, there is always a need to formulate an algorithm that can learn from limited amount of labelled data. An approach to formulate such algorithms is to incorporate the information contained by unlabelled data samples. But how can the unlabelled data samples be incorporated in our learning algorithm to boost performance ?
Consider the case shown in figure 1.1, where we try to assess the choice of our prior on first, labelled data and second, labelled as well as unlabelled data. It can clearly be seen that the choice of our prior changes drastically as we consider unlabelled data along with labelled data, this change is attributed to our understanding of the intrinsic geometry of the unlabelled data samples.

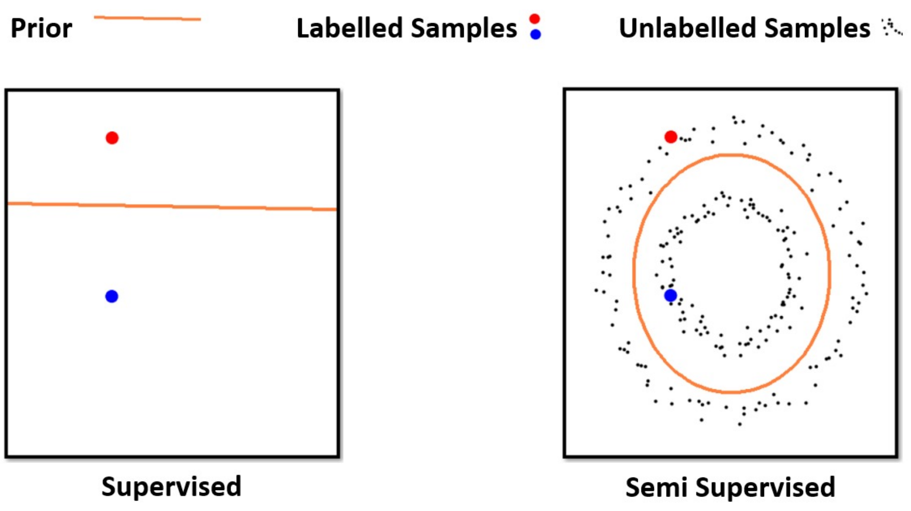


Figure 1.1: Semi Supervised learning

We in this thesis, formulated such a geometric algorithm and proposed a framework to formulate more of such algorithms, that can incorporate the geometrical information contained by unlabelled samples and thus, can give better generalizations and performance even with limited labelled data. A popular framework called Manifold Regularization, to formulate similar semi-supervised geometric algorithms was proposed by [BNS06].

1.2 Manifold Regularization

[BNS06] proposed a geometric framework which tries to incorporate the intrinsic geometry of the marginal, by proposing a penalty term in the objective. Conventional learning algorithms can be represented by the following generalized optimization problem:

$$f^* = \arg\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A ||f||_A^2 \tag{1.1}$$

where, V is some loss function, f is the estimated function and $\{(x_i)\}_{i=1}^{l}$ are the labelled samples from the training set with the corresponding labels, $\{y_i\}_{i=1}^{l}$. The term $||f||_A^2$ is an norm penalty in the ambient space. The Manifold regularization framework introduces another penalty term in the above problem, $||f||_I^2$ which is a norm penalty in the Intrinsic space of the marginal. This term is the central entity, which incorporates the intrinsic geometry of the marginal into the learning algorithm, with an assumption that the conditional varies smoothly on the geodesics of the marginal [BNS06]. The resulting manifold regularized semi-supervised learning framework, can be generalized by the following optimization problem:

$$f^* = \arg\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A ||f||_A^2 + \gamma_I ||f||_I^2 \tag{1.2}$$

[BNS06] further proposed that this intrinsic norm penalty term can be estimated using the Graph-Laplacian of the data. So, if L represents the Graph-Laplacian then the learning framework can be generalized by,

$$f^* = \arg\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A ||f||_A^2 + \frac{\gamma_I}{(u+l)^2} f^T L f \tag{1.3}$$

where, l is the number of labelled samples and u is the number of unlabelled samples.
[BNS06] further applied this learning framework on Support Vector Machines (SVM) and proposed a semi-supervised version called Laplacian Support Vector Machines or LapSVM which we quickly describe in the following section.

1.2.1 LapSVM

Consider, the case where we have l labelled samples $\{x_i\}_{i=1}^{l}$ with corresponding labels $\{y_i\}_{i=1}^{l}$ and u unlabelled samples $\{x_j\}_{j=l+1}^{l+u}$. If K is the Gram matrix or the Kernel matrix such that, $K_{ij} = K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ and L is the Graph-Laplacian, [BNS06] proposed the following optimization problem called LapSVM,

$$\min_{\lambda,b,q} \frac{1}{l}\sum_{i=1}^{l} q_i + \gamma_A \lambda^T K \lambda + \frac{\gamma_I}{(l+u)^2}\lambda^T KLK\lambda \tag{1.4}$$

such that,

$$y_i\Big(\sum_{j=1}^{l+u}\lambda_j K(x_i, x_j) + b\Big) + q_i \geq 1$$

$$q_i \geq 0$$

$$\forall i = 1, 2, 3, ..., l$$

here, λ are vectors in the Empirical Feature Space (EFS) and essential to estimate the function as, $f = \sum_{j=1}^{l+u}\lambda_j K(x, x_j) + b$. This problem can be converted to a constraint quadratic problem in the dual form as derived in [BNS06].
We in this thesis, use the same manifold regularization learning framework to formulate a classification algorithm with tight bound on the VC dimension. We further also propose a similar semi-supervised geometric learning framework inspired by Graph Trend Filtering (GTF). All the details are in the following chapters.

Chapter 2

Manifold Regularized Minimal Complexity Machines - Lap MCM

2.1 Introduction to Minimal Complexity Machines - MCM

The idea of Mininal Complexity Machines (MCM) [Jay15] is to minimize the tight bound on the VC dimension, thus constructing a classifier with better generalization performance [Bur98] and, minimal complexity. In [Vap98], Vapnik derived that the VC dimension is bounded by,

$$VC \leq 1 + \min\left(\frac{R^2}{d_{min}^2}, n\right) \tag{2.1}$$

for the set of all the hyperplane classifiers with with margin $d \geq d_{min}$ where, R is the radius of all the training samples and, n is the dimension of the training samples. [Jay15] minimizes the $\frac{R}{d}$ ratio and formulated MCM, a linear optimization problem which implicitly minimizes the tight bound on VC dimension. The linear form of the problem is as follows:

$$\min_{h,b,q} \quad h + C\sum_{i=1}^{l} q_i \tag{2.2}$$

$$\begin{gathered} \text{such that,} \\ h \geq y_i(w^T x_i + b) \\ y_i(w^T x_i + b) + q \geq 1 \\ q_i \geq 0, h \geq 1 \\ \forall i = 1, 2, 3, ..., m \end{gathered}$$

where, C is a hyper-parameter, $X = \{x_1, x_2, ..., x_m\}$ are the training samples with the labels, $Y = \{y_1, y_2, ..., y_m\}$ where each y_i can either take the value 1 or -1. The resulting hyperplane is given by $y = w^t x + b$ and the h^2 represents the tight bound on the VC dimension i.e,

$$\alpha h^2 \leq VC \leq \beta h^2 \implies h^2 \sim \theta(VC) \tag{2.3}$$

[Jay15] also gave the kernel version of the above problem, called the Kernel MCM with the following formulation,

$$\min_{h,\lambda,b,q} \quad h + C\sum_{i=1}^{l} q_i \tag{2.4}$$

$$\text{such that,}$$
$$h \geq y_i\Big(\sum_{j=1}^{m} \lambda_j K(x_i, x_j) + b\Big)$$
$$y_i\Big(\sum_{j=1}^{m} \lambda_j K(x_i, x_j) + b\Big) + q \geq 1$$
$$q \geq 0, h \geq 1$$
$$\forall i = 1, 2, 3, ..., l$$

where, $\lambda = \{\lambda_1, \lambda_2, ..., \lambda_m\}$ are the vectors in Empirical Feature Space (EFS) and K is the Gram matrix such that, $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$.
MCM performs better than the contemporary Support Vector Machines (SVM) along with significantly smaller number of support vectors, which can be attributed to it's minimal complexity. The MCM has various other formulation but they lie on the central idea of minimizing the VC dimension. We in this thesis, extend this idea of minimizing the VC dimension to semi-supervised learning using Manifold Regularization.

2.2 Laplacian MCM

We propose the semi-supervised version of MCM which along with incorporating the geometry of the marginal, also minimizes the VC dimension. We call our proposed algorithm the Laplacian Minimal Complexity Machines (LapMCM) which can be formulated as the following optimization problem:

$$\min_{h,\lambda,b,q} \quad \frac{h^2}{2} + C\frac{1}{2l}\sum_{i=1}^{l} q_i^2 + \frac{\gamma_A}{2}\lambda^T K\lambda + \frac{\gamma_I}{l+u}\lambda^T KLK\lambda \tag{2.5}$$

$$\text{such that,}$$
$$h \geq y_i\Big(\sum_{j=1}^{l+u} \lambda_j K(x_i, x_j) + b\Big)$$
$$y_i\Big(\sum_{j=1}^{l+u} \lambda_j K(x_i, x_j) + b\Big) + q \geq 1$$
$$\forall i = 1, 2, 3, ..., l$$

where, C, γ_I, γ_A are hyper-parameters and $X_l = \{x_1, x_2, ..., x_l\}$ are the l, labelled-training samples with the labels, $Y_l = \{y_1, y_2, ..., y_l\}$ and, $X_u = \{x_{l+1}, x_{l+2}, ..., x_{l+u}\}$ are the u, unlabelled samples. The optimal solution for the problem, according to the Representator's theorem is given by,

$$f^*(x) = \sum_{j=1}^{l+u} \lambda_j^* K(x, x_j) \tag{2.6}$$

The first term in the objective, corresponds to the tight bound on the VC dimension i.e, $h^2 \sim \theta(VC)$, the second term is the sum of the slack variables which allows for some misclassification. The hyper-parameter, C thus determines the trade-off between the minimization of VC dimension and the classification-accuracy. The third term in the objective, is a conditional term with the hyper-parameter γ_A, which being quadratic improves the convexity of the problem. The fourth and the last term in the objective corresponds to the manifold regularization term and the hyper-parameter γ_I thus, determines the extent of the manifold regularization.
The problem can be converted to it's quadratic-dual form which be obtained by introducing non-negative Lagrange multipliers, $\alpha_1, ..., \alpha_l, \alpha_{l+1}, ..., \alpha_{2l}$. The lagrangian thus, can be written as:

$$\begin{aligned} L(h, \lambda, b, q, \alpha) = \frac{h^2}{2} + \frac{C}{2l}\sum_{i=1}^{l} q_i^2 + \frac{\gamma_A}{2}\lambda^T K \lambda + \frac{\gamma_I}{l+u}\lambda^T K L K \lambda \\ - \sum_{i=1}^{l} \alpha_i \left(h - y_i\Big(\sum_{j=1}^{l+u} \lambda_j K(x_i, x_j) + b\Big)\right) \\ - \sum_{i=1}^{l} \alpha_{l+i} \left(y_i\Big(\sum_{j=1}^{l+u} \lambda_j K(x_i, x_j) + b\Big) + q_i - 1\right) \end{aligned} \tag{2.7}$$

The KKT conditions for the dual to pass are $\frac{\partial}{\partial b} = \frac{\partial}{\partial q} = \frac{\partial}{\partial h} = \frac{\partial}{\partial \lambda} = 0$, thus we have:

$$\frac{\partial}{\partial b} = 0 \implies \sum_{i=1}^{l} \alpha_i\, y_i - \sum_{i=1}^{l} \alpha_{i+l}\, y_i = 0 \tag{2.8}$$

$$\frac{\partial}{\partial h} = 0 \implies h - \sum_{i=1}^{l} \alpha_i = 0 \implies h = \sum_{i=1}^{l} \alpha_i \tag{2.9}$$

$$\frac{\partial}{\partial q_i} = 0 \implies \frac{Cq_i}{l} - \sum_{i=1}^{l} \alpha_{i+l} = 0 \implies q_i = \frac{l}{C}\sum_{i=1}^{l} \alpha_{i+l} \tag{2.10}$$

Substituting, the above obtained results in 2.7 to obtain the reduced Lagrangian, L^R :

$$
\begin{aligned}
L^R = h\Big(\frac{h}{2} - \sum_{i=1}^{l}\alpha_i\Big) + \sum_{i=1}^{l} q_i\Big(\frac{Cq_i}{2l} - \sum_{i=1}^{l}\alpha_{i+l}\Big) + \sum_{i=1}^{l}\alpha_{l+i} + \frac{\gamma_A}{2}\lambda^T K\lambda + \frac{\gamma_I}{2}\lambda^T KLK\lambda \\
- \sum_{i=1}^{l}(\alpha_{l+i} - \alpha_i)\ y_i\Big(\sum_{j=1}^{l+u}\lambda_j K(x_i,,x_j)\Big)
\end{aligned}
$$

using the values from 2.8, 2.9 and 2.10 the reduced Lagrangian, L^R becomes:

$$
\begin{aligned}
L^R = -\frac{1}{2}\Big(\sum_{i=1}^{l}\alpha_i\Big)^2 - \frac{l}{2C}\Big(\sum_{i=1}^{l}\alpha_{l+i}\Big)^2 + \sum_{i=1}^{L}\alpha_{l+i} + \frac{\gamma_A}{2}\lambda^T K\lambda + \frac{\gamma_I}{2(l+u)^2}\lambda^T KLK\lambda \\
- \sum_{i=1}^{l}(\alpha_{l+i} - \alpha_i)\ y_i\Big(\sum_{j=1}^{l+u}\lambda_j K(x_i,,x_j)\Big)
\end{aligned}
$$

Further, we can use $\sum_{i=1}^{l}(\alpha_{l+i} - \alpha_i)\ y_i\Big(\sum_{j=1}^{l+u}\lambda_j K(x_i,,x_j)\Big) = \lambda^T KJ^T Y(\alpha^l - \alpha^0)$ where, $Y = diag(y_1, y_2, ..., y_l)$; $J = [I_{l\times l}\quad 0_{l\times u}]$ (I is the identity matrix) and $\alpha = [\alpha^0\quad \alpha^l]^T$ such that, $\alpha^0 = [\alpha_1, ..., \alpha_l]$ and $\alpha^l = [\alpha_{l+1}, ..., \alpha_{2l}]^T$, Thus, the reduced Lagrangian becomes the following:

$$
\begin{aligned}
L^R = -\frac{1}{2}\Big(\sum_{i=1}^{l}\alpha_i\Big)^2 - \frac{l}{2C}\Big(\sum_{i=1}^{l}\alpha_{l+i}\Big)^2 + \sum_{i=1}^{L}\alpha_{l+i} + \frac{\gamma_A}{2}\lambda^T K\lambda + \frac{\gamma_I}{2(l+u)^2}\lambda^T KLK\lambda \\
- \lambda^T KJ^T Y(\alpha^l - \alpha^0)
\end{aligned}
$$

$$
L^R = -\frac{1}{2}\Big(\sum_{i=1}^{l}\alpha_i\Big)^2 - \frac{l}{2C}\Big(\sum_{i=1}^{l}\alpha_{l+i}\Big)^2 + \sum_{i=1}^{L}\alpha_{l+i} + \lambda^T K\Big\{\Big(\frac{\gamma_I}{(l+u)^2}LK + \gamma_A I\Big)\lambda - J^T Y(\alpha^l - \alpha^0)\Big\} \tag{2.11}
$$

Now, further using the last KKT condition i.e, $\frac{\partial L}{\partial\lambda} = 0$

$$
\frac{\partial}{\partial\lambda} = 0 \implies \frac{\gamma_I}{(l+u)^2}KLK\lambda + \gamma_A K\lambda - KJ^T Y(\alpha^l - \alpha^0) = 0
$$

This, further implies that,

$$
K\Big\{\Big(\frac{\gamma_I}{(l+u)^2}LK + \gamma_A I\Big)\lambda - J^T Y(\alpha^l - \alpha^0)\Big\} = 0 \implies \Big(\frac{\gamma_I}{(l+u)^2}LK + \gamma_A I\Big)\lambda = J^T Y(\alpha^l - \alpha^0)
$$

$$\implies \lambda = \Big(\frac{\gamma_I}{(l+u)^2}LK + \gamma_A I\Big)^{-1} J^T Y(\alpha^l - \alpha^0) \tag{2.12}$$

Now substituting the above obtained result (2.12) in 2.11 to get,

$$L^R = -\frac{1}{2}\Big(\sum_{i=1}^{l}\alpha_i\Big)^2 - \frac{l}{2C}\Big(\sum_{i=1}^{l}\alpha_{l+i}\Big)^2 + \sum_{i=1}^{L}\alpha_{l+i} - \frac{1}{2}J^T Y(\alpha^l - \alpha^0) \tag{2.13}$$

Now, consider a symmetric matrix M such that, $\lambda^T K J^T Y(\alpha^l - \alpha^0 = (\alpha^l - \alpha^0)^T M(\alpha^l - \alpha^0)$ then, $M = (\alpha^l - \alpha^0)^{-T}\lambda^T K J^T Y$ where, $-T$ is the inverse-transpose operation. Now using the value of λ from 2.12 we get the following,

$$M^T = M = \Big((\alpha^l - \alpha^0)^{-T}\lambda^T K J^T Y\Big)^T = YJK\lambda(\alpha^l - \alpha^0)^{-1}$$
$$M = YJK\Big(\frac{\gamma_I}{(l+u)^2}LK + \gamma_A I\Big)^{-1} J^T Y(\alpha^l - \alpha^0)(\alpha^l - \alpha^0)^{-1}$$
$$M = YJK\Big(\frac{\gamma_I}{(l+u)^2}LK + \gamma_A I\Big)^{-1} J^T Y \tag{2.14}$$

Thus, 2.13 becomes a quadratic in α as follows,

$$L^R = -\frac{1}{2}\Big(\sum_{i=1}^{l}\alpha_i\Big)^2 - \frac{l}{2C}\Big(\sum_{i=1}^{l}\alpha_{l+i}\Big)^2 + \sum_{i=1}^{l}\alpha_{l+i} - \frac{1}{2}(\alpha^l - \alpha^0)^T M(\alpha^l - \alpha^0) \tag{2.15}$$

Now, for further simplification we would like to express the reduced lagrangian explicitly in terms of $\alpha = [\alpha^0 \quad \alpha^l]$. We do so as follows,

$$\Big(\sum_{i=1}^{l}\alpha_i\Big)^2 = \begin{bmatrix}\alpha^0 & \alpha^l\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\alpha^0\\ \alpha^l\end{bmatrix} = \alpha^T\begin{bmatrix}1_{l\times l} & 0_{l\times l}\\ 0_{l\times l} & 0_{l\times l}\end{bmatrix}\alpha$$

$$\Big(\sum_{i=1}^{l}\alpha_{l+i}\Big)^2 = \begin{bmatrix}\alpha^0 & \alpha^l\end{bmatrix}\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\alpha^0\\ \alpha^l\end{bmatrix} = \alpha^T\begin{bmatrix}0_{l\times l} & 0_{l\times l}\\ 0_{l\times l} & 1_{l\times l}\end{bmatrix}\alpha$$

$$\alpha^l - \alpha^0 = \begin{bmatrix}-I & I\end{bmatrix}\begin{bmatrix}\alpha^0\\ \alpha^l\end{bmatrix} = \begin{bmatrix}-I_{l\times l} & I_{l\times l}\end{bmatrix}\alpha$$

where, $1_{l\times l}$ is an $l \times l$ matrix having all element as 1s. Similarly, $0_{l\times l}$ is an $l \times l$ matrix having all element as 0s and $I_{l\times l}$ is $l \times l$ Identity matrix. Substituting the above values in 2.15,

$$L^R = -\frac{1}{2}\alpha^T \begin{bmatrix} 1_{l\times l} & 0_{l\times l} \\ 0_{l\times l} & 0_{l\times l} \end{bmatrix} \alpha - \frac{l}{2C}\alpha^T \begin{bmatrix} 0_{l\times l} & 0_{l\times l} \\ 0_{l\times l} & 1_{l\times l} \end{bmatrix} \alpha + \sum_{i=1}^{l} \alpha_{l+i} \\ - \frac{1}{2}\Big(\begin{bmatrix} -I_{l\times l} & I_{l\times l} \end{bmatrix} \alpha \Big)^T M \Big(\begin{bmatrix} -I_{l\times l} & I_{l\times l} \end{bmatrix} \alpha \Big)$$

Which further gives,

$$L^R = -\frac{1}{2}\alpha^T \begin{bmatrix} 1_{l\times l} & 0_{l\times l} \\ 0_{l\times l} & 0_{l\times l} \end{bmatrix} \alpha - \frac{l}{2C}\alpha^T \begin{bmatrix} 0_{l\times l} & 0_{l\times l} \\ 0_{l\times l} & 1_{l\times l} \end{bmatrix} \alpha + \sum_{i=1}^{l} \alpha_{l+i} \\ - \frac{1}{2}\alpha^T \begin{bmatrix} -I_{l\times l} \\ I_{l\times l} \end{bmatrix} M \begin{bmatrix} -I_{l\times l} & I_{l\times l} \end{bmatrix} \alpha$$

Thus, the reduced Lagrangian can be expressed as,

$$L^R = \sum_{i=1}^{l} \alpha_{l+i} - \frac{1}{2}\alpha^T Q \alpha \tag{2.16}$$

where,

$$Q = \begin{bmatrix} 1_{l\times l} & 0_{l\times l} \\ 0_{l\times l} & 0_{l\times l} \end{bmatrix} + \frac{l}{C} \begin{bmatrix} 0_{l\times l} & 0_{l\times l} \\ 0_{l\times l} & 1_{l\times l} \end{bmatrix} + \begin{bmatrix} -I_{l\times l} \\ I_{l\times l} \end{bmatrix} M \begin{bmatrix} -I_{l\times l} & I_{l\times l} \end{bmatrix} \tag{2.17}$$

Thus, the dual of the problem becomes,

$$\alpha^* = \arg\max \;\; \sum_{i=1}^{l} \alpha_{l+i} - \frac{1}{2}\alpha^T Q \alpha \tag{2.18}$$

such that,

$$\sum_{i=1}^{l} \alpha_i \, y_i - \sum_{i=1}^{l} \alpha_{i+l} \, y_i = 0$$

where the constraint comes from 2.8. The constraint can be simplified as, $\sum_{i=1}^{l}(\alpha_{l+i} - \alpha_i) y_i = 0 \implies \begin{bmatrix} 1_{1\times l} \end{bmatrix} Y \begin{bmatrix} -I_{l\times l} & I_{l\times l} \end{bmatrix} \alpha = 0 \implies A\alpha = 0$ where, $1_{1\times l}$ is an $1 \times l$ matrix having all elements as 1, and A is as follows,

$$A = \begin{bmatrix} 1_{1\times l} \end{bmatrix} Y \begin{bmatrix} -I_{l\times l} & I_{l\times l} \end{bmatrix} \tag{2.19}$$

Also, consider a matrix e such that,

$$e^T = \begin{bmatrix} 0_{1\times l} & 1_{1\times l} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 & 1 & 1 & \dots & 1 \end{bmatrix}_{1\times 2l} \tag{2.20}$$

Then, by substituting the values of A and e in 2.18 the simplified dual for LapMCM and by incorporating the non-negative nature of Lagrangian multipliers the final problem becomes,

$$\alpha^* = \arg\max \quad e^T\alpha - \frac{1}{2}\alpha^T Q\alpha \tag{2.21}$$
$$\text{such that,}$$
$$A\alpha = 0, \;\; \alpha \geq 0$$

where values of Q, e and A are given by 2.17, 2.20 and 2.19 respectively.

2.3 Unconstrained Laplacian MCM

2.3.1 Sequential Unconstrained Minimization Technique (SUMT)

The dual of the LapMCM, we formulated and derived in the above section was a constrained quadratic optimization problem. Unconstrained optimization problems offer an inherent advantage over constrained problems, as they can be quickly solved using techniques like the Newton's method. [JJRC12] proposed Sequential Unconstrained Minimization Technique (SUMT) for SVM solvers. Using the SUMT, a constrained problem can be converted to a corresponding unconstrained problem. Let the constrained problem be as follows,

$$\min \quad f(x) \tag{2.22}$$
$$\text{such that,} \;\; \forall\, j = 1, 2, 3, ..n$$
$$h_j(x) = 0$$

The corresponding unconstrained problem will then be constructed by adding the square of each of the constraint in the objective as shown below [JJRC12],

$$\min \quad E_p(x) = f(x) + \sum_{j=1}^{n} \alpha_p \; h_j^2(x) \tag{2.23}$$

The solutions for the above constrained problem can be obtained using following the below mentioned steps as shown by [JJRC12],

1. Set $p = 0$. Choose the coefficient α_0, and an initial state x_0
2. Find the minimum of $E_p(x)$. Denote the solution as x_p^*
3. If all the constraints in the original problem are satisfied, stop

4. If not, choose x_p^* as the new initial state, and choose α_{p+1} such that $\alpha_{p+1} > \alpha_p$. Set $p = p + 1$. Go to step 2

5. In the limit, as $p \to \infty$, the sequence of minimas $x_1^*, x_2^*, ...x_p^*, ...$ will converge to the solution of the original problem

2.3.2 Unconstrained Laplacian MCM using SUMT

The above mentioned SUMT can be incorporated in our LapMCM, to obtain an unconstrained LapMCM. The primal form of the constrained problem is given in 2.5 in which the constraint is generated due to the b term thus, the corresponding unconstrained problem can be formulated by adding b^2 in the objective function to obtain the following optimization problem:

$$\min_{h,\lambda,b,q} \quad \frac{h^2}{2} + \frac{C}{2l}\sum_{i=1}^{l} q_i^2 + \frac{\gamma_A}{2}\lambda^T K\lambda + \frac{\gamma_I}{l+u}\lambda^T KLK\lambda + \frac{p}{2}b^2 \tag{2.24}$$

such that,

$$h \geq y_i\Big(\sum_{j=1}^{l+u}\lambda_j K(x_i,x_j) + b\Big)$$

$$y_i\Big(\sum_{j=1}^{l+u}\lambda_j K(x_i,x_j) + b\Big) + q \geq 1$$

$$\forall i = 1,2,3,...,l$$

where, p is another hyper-parameter which according to the SUMT must be taken large i.e, around 1000-2000. Further, similar to the previous section, the optimal solution is again given by $f^*(x) = \sum_{j=1}^{l+u}\lambda_j^* K(x,x_j)$. To obtain the dual form of the above unconstrained problem we again introduce the non-negative Lagrange multipliers, $\alpha_1, ..., \alpha_l, \alpha_{l+1}, ..., \alpha_{2l}$ as follows:

$$\begin{aligned} L(h,\lambda,b,q,\alpha) = \frac{h^2}{2} + \frac{C}{2l}\sum_{i=1}^{l} q_i^2 + \frac{\gamma_A}{2}\lambda^T K\lambda + \frac{\gamma_I}{l+u}\lambda^T KLK\lambda + \frac{p}{2}b^2 \\ - \sum_{i=1}^{l}\alpha_i\left(h - y_i\Big(\sum_{j=1}^{l+u}\lambda_j K(x_i,x_j) + b\Big)\right) \\ - \sum_{i=1}^{l}\alpha_{l+i}\left(y_i\Big(\sum_{j=1}^{l+u}\lambda_j K(x_i,x_j) + b\Big) + q_i - 1\right) \end{aligned} \tag{2.25}$$

The KKT conditions for the dual to pass are $\frac{\partial}{\partial b} = \frac{\partial}{\partial q} = \frac{\partial}{\partial h} = \frac{\partial}{\partial \lambda} = 0$, thus we have:

$$\frac{\partial}{\partial b} = 0 \implies pb + \sum_{i=1}^{l} \alpha_i \ y_i - \sum_{i=1}^{l} \alpha_{i+l} \ y_i = 0 \implies b = \frac{1}{p} \sum_{i=1}^{l} (\alpha_{l+i} - \alpha_i) y_i \tag{2.26}$$

$$\frac{\partial}{\partial h} = 0 \implies h - \sum_{i=1}^{l} \alpha_i = 0 \implies h = \sum_{i=1}^{l} \alpha_i \tag{2.27}$$

$$\frac{\partial}{\partial q_i} = 0 \implies \frac{Cq_i}{l} - \sum_{i=1}^{l} \alpha_{i+l} = 0 \implies q_i = l \sum_{i=1}^{l} \alpha_{i+l} \tag{2.28}$$

Substituting, the above obtained results in 2.7 to obtain the reduced Lagrangian, L^R :

$$L^R = h\Big(\frac{h}{2} - \sum_{i=1}^{l} \alpha_i\Big) + \sum_{i=1}^{l} q_i \Big(\frac{Cq_i}{2l} - \sum_{i=1}^{l} \alpha_{i+l}\Big) + \sum_{i=1}^{l} \alpha_{l+i} + \frac{\gamma_A}{2} \lambda^T K \lambda + \frac{\gamma_I}{2} \lambda^T K L K \lambda$$
$$+ b\Big(\frac{p\,b}{2} + \sum_{i=1}^{l} (\alpha_{l+i} - \alpha_i) y_i\Big) - \sum_{i=1}^{l} (\alpha_{l+i} - \alpha_i) \ y_i \Big(\sum_{j=1}^{l+u} \lambda_j K(x_i, , x_j)\Big)$$

using the values from 2.26, 2.27 and 2.28 the reduced Lagrangian, L^R becomes:

$$L^R = -\frac{1}{2}\Big(\sum_{i=1}^{l} \alpha_i\Big)^2 - \frac{l}{2C}\Big(\sum_{i=1}^{l} \alpha_{l+i}\Big)^2 + \sum_{i=1}^{L} \alpha_{l+i} + \frac{\gamma_A}{2} \lambda^T K \lambda + \frac{\gamma_I}{2(l+u)^2} \lambda^T K L K \lambda$$
$$- \frac{1}{2p}\Big(\sum_{i=1}^{l} (\alpha_{l+i} - \alpha_i) y_i\Big)^2 - \lambda^T K J^T Y (\alpha^l - \alpha^0)$$

Proceeding in a way similar to what was followed in the previous section,

$$L^R = -\frac{1}{2}\Big(\sum_{i=1}^{l} \alpha_i\Big)^2 - \frac{l}{2C}\Big(\sum_{i=1}^{l} \alpha_{l+i}\Big)^2 + \sum_{i=1}^{L} \alpha_{l+i} + \lambda^T K \Big\{ \Big(\frac{\gamma_I}{(l+u)^2} LK + \gamma_A I \Big) \lambda - J^T Y (\alpha^l - \alpha^0) \Big\}$$
$$- \frac{1}{2p}\Big(\sum_{i=1}^{l} (\alpha_{l+i} - \alpha_i) y_i\Big)^2 \tag{2.29}$$

Now, further using the last KKT condition i.e, $\frac{\partial L}{\partial \lambda} = 0$ we get the same result we obtained for constrained LapMCM i.e,

$$\lambda = \Big(\frac{\gamma_I}{(l+u)^2} LK + \gamma_A I \Big)^{-1} J^T Y (\alpha^l - \alpha^0) \tag{2.30}$$

Now substituting the above obtained result (2.30) in 2.29 to get,

$$L^R = -\frac{1}{2}\Big(\sum_{i=1}^{l}\alpha_i\Big)^2 - \frac{l}{2C}\Big(\sum_{i=1}^{l}\alpha_{l+i}\Big)^2 + \sum_{i=1}^{L}\alpha_{l+i} - \frac{1}{2}J^TY(\alpha^l - \alpha^0) - \frac{1}{2p}\Big(\sum_{i=1}^{l}(\alpha_{l+i}-\alpha_i)y_i\Big)^2 \tag{2.31}$$

Now, again as we did for constrained LapMCM the symmetric M such that, $\lambda^T KJ^TY(\alpha^l - \alpha^0 = (\alpha^l-\alpha^0)^T M(\alpha^l-\alpha^0)$ has the following value,

$$M = YJK\Big(\frac{\gamma_I}{(l+u)^2}LK + \gamma_A I\Big)^{-1}J^TY$$

Thus, 2.31 becomes a quadratic in α as follows,

$$L^R = -\frac{1}{2}\Big(\sum_{i=1}^{l}\alpha_i\Big)^2 - \frac{l}{2C}\Big(\sum_{i=1}^{l}\alpha_{l+i}\Big)^2 + \sum_{i=1}^{l}\alpha_{l+i} - \frac{1}{2}(\alpha^l-\alpha^0)^TM(\alpha^l-\alpha^0) - \frac{1}{2p}\Big(\sum_{i=1}^{l}(\alpha_{l+i}-\alpha_i)y_i\Big)^2 \tag{2.32}$$

Now, for further simplification consider the following term,

$$(\alpha^l-\alpha^0)^TM(\alpha^l-\alpha^0) + \frac{1}{p}\Big(\sum_{i=1}^{l}(\alpha_{l+i}-\alpha_i)y_i\Big)^2 = \sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_{l+i}-\alpha_i)\ m_{ij}\ (\alpha_{l+j}-\alpha_j) + \frac{1}{p}\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_{l+i}-\alpha_i)\ y_i\ y_j\ (\alpha_{l+j}-\alpha_j) \tag{2.33}$$

where, $M = \{m_{ij}\} = YJK\Big(\frac{\gamma_I}{(l+u)^2}LK + \gamma_A I\Big)^{-1}J^TY = YRY$ i.e, $M = YRY$ and Y is a diagonal matrix we can write, $m_{ij} = y_i\ r_{ij}\ y_j$ where,

$$R = \{r_{ij}\} = K\Big(\frac{\gamma_I}{(l+u)^2}LK + \gamma_A I\Big)^{-1}J^T \tag{2.34}$$

Thus, the equation 2.33 becomes,

$$\begin{aligned}(\alpha^l-\alpha^0)^TM(\alpha^l-\alpha^0) + \frac{1}{p}\Big(\sum_{i=1}^{l}(\alpha_{l+i}-\alpha_i)y_i\Big)^2 \\ = \sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_{l+i}-\alpha_i)\ y_i\ r_{ij}\ y_j\ (\alpha_{l+j}-\alpha_j) + \frac{1}{p}\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_{l+i}-\alpha_i)\ y_i\ y_j\ (\alpha_{l+j}-\alpha_j) \\ = \sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_{l+i}-\alpha_i)\ y_i\ (r_{ij}+\frac{1}{p})\ y_j\ (\alpha_{l+j}-\alpha_j) = \sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_{l+i}-\alpha_i)\ p_{ij}\ (\alpha_{l+j}-\alpha_j)\end{aligned} \tag{2.35}$$

where, we can have a matrix, P such that $P = \{p_{ij}\} = \Big\{y_i\ (r_{ij} + \frac{1}{p})\ y_j\Big\}$. As, Y is a diagonal matrix we can further have,

$$P = Y\Big(R + \frac{I}{p}\Big)Y \tag{2.36}$$

where, I is the identity matrix. Now, substituting 2.34 in 2.33 to get the following,

$$L^R = -\frac{1}{2}\Big(\sum_{i=1}^{l}\alpha_i\Big)^2 - \frac{l}{2C}\Big(\sum_{i=1}^{l}\alpha_{l+i}\Big)^2 + \sum_{i=1}^{l}\alpha_{l+i} - \frac{1}{2}\Big\{\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_{l+i} - \alpha_i)\ p_{ij}\ (\alpha_{l+j} - \alpha_j)\Big\}$$

$$L^R = -\frac{1}{2}\Big(\sum_{i=1}^{l}\alpha_i\Big)^2 - \frac{l}{2C}\Big(\sum_{i=1}^{l}\alpha_{l+i}\Big)^2 + \sum_{i=1}^{l}\alpha_{l+i} - \frac{1}{2}(\alpha^l - \alpha^0)^T P(\alpha^l - \alpha^0)$$

Now, substituting all the values in terms of α as we did for the constrained LapMCM. The resulting reduced Lagrangian becomes,

$$L^R = -\frac{1}{2}\alpha^T \begin{bmatrix} 1_{l\times l} & 0_{l\times l} \\ 0_{l\times l} & 0_{l\times l} \end{bmatrix} \alpha - \frac{l}{2C}\alpha^T \begin{bmatrix} 0_{l\times l} & 0_{l\times l} \\ 0_{l\times l} & 1_{l\times l} \end{bmatrix} \alpha + \sum_{i=1}^{l}\alpha_{l+i} - \frac{1}{2}\alpha^T \begin{bmatrix} -I_{l\times l} \\ I_{l\times l} \end{bmatrix} P \begin{bmatrix} -I_{l\times l} & I_{l\times l} \end{bmatrix} \alpha$$

Thus, the reduced Lagrangian can be expressed as,

$$L^R = e^T\alpha - \frac{1}{2}\alpha^T Q\alpha \tag{2.37}$$

where e is given by 2.20 and Q in terms of P(2.36, 2.34) is as follows,

$$Q = \begin{bmatrix} 1_{l\times l} & 0_{l\times l} \\ 0_{l\times l} & 0_{l\times l} \end{bmatrix} + \frac{l}{C} \begin{bmatrix} 0_{l\times l} & 0_{l\times l} \\ 0_{l\times l} & 1_{l\times l} \end{bmatrix} + \begin{bmatrix} -I_{l\times l} \\ I_{l\times l} \end{bmatrix} P \begin{bmatrix} -I_{l\times l} & I_{l\times l} \end{bmatrix} \tag{2.38}$$

Incorporating the non-negative nature of the Lagrangian multipliers the dual of the problem becomes,

$$\alpha^* = \arg\max \quad e^T\alpha - \frac{1}{2}\alpha^T Q\alpha \tag{2.39}$$
$$\text{such that, } \alpha \geq 0$$

where values of Q, e and A are given by 2.38, 2.20 and 2.19 respectively.

2.3.3 Solving the unconstrained LapMCM using Newton's method

Consider the objective function of the problem, $E(\alpha) = \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$. To solve the problem we must minimize E(α) and it can be done by iteratively changing each of the component of the α i.e, α_k while keeping all other components, α_i fixed.
The update rule at the $(r+1)^{th}$ iteration for the K^{th} component will be,

$$\alpha_k(r+1) = \alpha_k(r) - \frac{\left(\partial E/\partial\alpha_k\right)}{\left(\partial^2 E/\partial\alpha_k^2\right)}$$

$$\implies \quad \alpha_k(r+1) = \alpha_k(r) + \frac{(1 - \sum\limits_{j=1}^{l} Q_{kj}\alpha_j)}{Q_{kk}}$$
$$\forall k = 1, 2, 3, ..l$$

The inequality constraint have to be checked after each iteration and the the component of α must be clipped to the follow the constraint i.e,

$$if(\alpha_k(r+1) < 0): \quad \alpha_k \longleftarrow 0$$

As, each component can be updated independently, we performed the updates for each of the component parallely on a GPU, to considerable reduce the training time for the algorithm.

The summarised algorithm with all the steps for training a LapMCM model can be found at Algorithm 1:

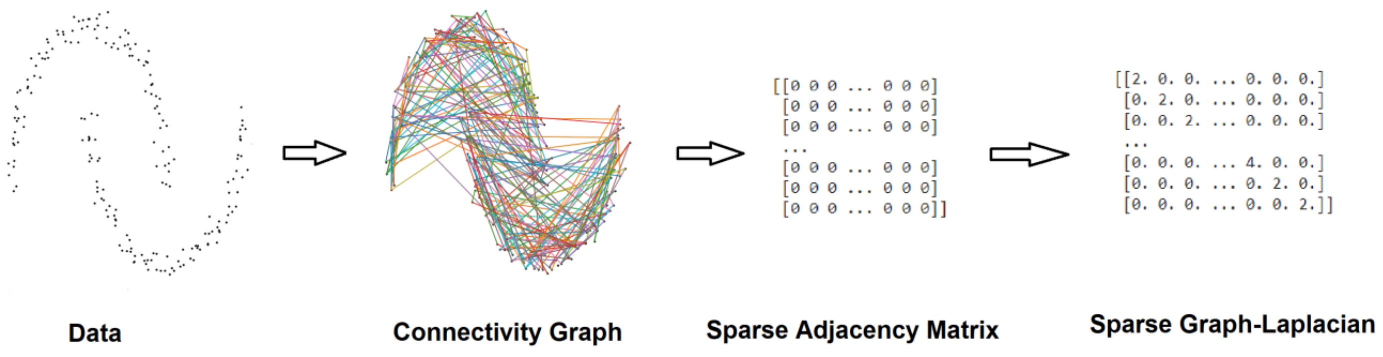

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```
```
[[2. 0. 0. ... 0. 0. 0.]
 [0. 2. 0. ... 0. 0. 0.]
 [0. 0. 2. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 4. 0. 0.]
 [0. 0. 0. ... 0. 2. 0.]
 [0. 0. 0. ... 0. 0. 2.]]
```

Figure 2.1: Constructing Graph Laplacian from the data

Algorithm 1: Laplacian Minimal Complexity Machines

```
Input: $l$ labelled samples $\{(x_i, y_i)\}_{i=1}^{l}$ and, $u$ un-labelled samples $\{x_j\}_{j=l+1}^{l+u}$
Output: $f(x) = \sum_{j=1}^{l+u} \lambda_j K(x, x_j) + b : \mathcal{R}^n \rightarrow \mathcal{R}$
1  Using the data construct a distance or connectivity graph, using which compute the
     Graph-Laplacian through the adjacency matrix as shown in 2.1
2  Choose hyper-parameters and compute the Gram matrix $K_{ij}$ such that
     $K_{ij} = K(x_i, x_j)$ (K: Kernel function)
3  Compute various helper matrices and $Q$ using 2.38 and other equations
4  Minimize $\frac{1}{2}\alpha^T Q\alpha - e^T\alpha$ by calling the optimize function
5  Function optimize($Q$, numItr):
6      Randomly Initialize the vector $\alpha$
7      for $k$ in length($\alpha$) do
8          for itr in range(numItr) do
9              $\alpha_k(itr+1) \longleftarrow \alpha_k(itr) + \frac{(1-Q[k,:]^T\alpha)}{Q[k,k]}$ if $\alpha_k(itr+1) < 0$ then
10                 $\alpha_k(itr+1) \longleftarrow 0$
11                 break
12             end
13         end
14     end
15     return $\alpha$
16 End Function
17 Compute EFS vector, $\lambda$ from $\alpha$ using 2.30
18 $f(x) = \sum_{j=1}^{l+u} \lambda_j K(x, x_j) + b$ and the predicted class, $y = sgn(f(x))$
```

The preliminary sanctity of the proposed algorithm can be assured by testing it on simple artificial datasets. We also used the well established Laplacian Support Vector Machines (LapSVM) proposed in [BNS06], as a contemporary to asses our proposed algorithm. The LapMCM and LapSVM algorithms were thus, tested on artificial two-moons, two-circles and the two-spiral datasets. Each datset consisted of 200 samples out of which a pair of samples from complementary classes were treated as labelled, while the other 198 were treated as unlabelled, while training the algorithm, The results are shown in the figure 2.2.

The results clearly show better generalization of LapMCM, with tighter decision boundary of the resulting hyperplane classifier.

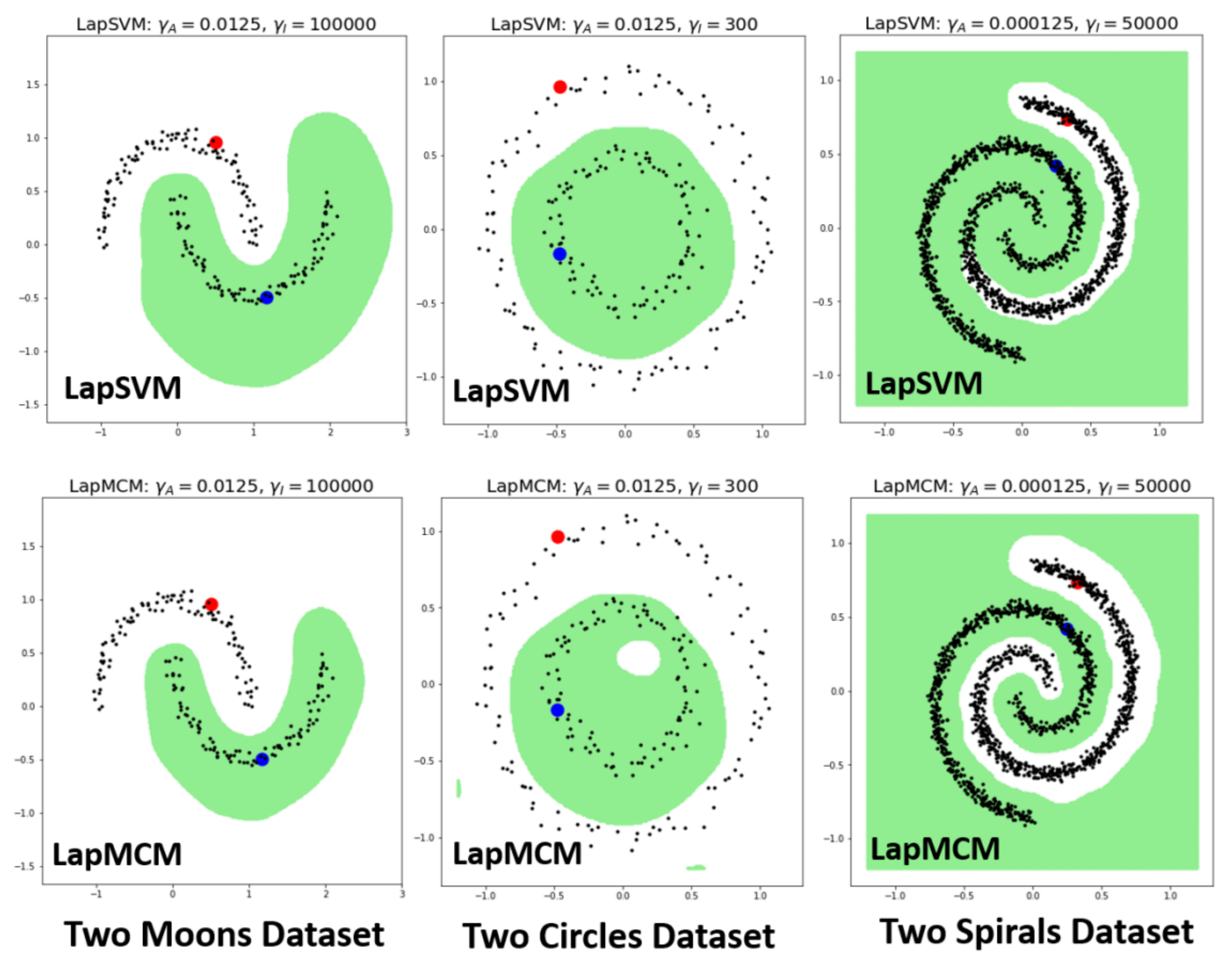


Figure 2.2: Performance of LapMCM on artificial datasets

In the following section, we propose another novel formulation of LapMCM.

Chapter 3

Manifold Regularization through Graph Trend Filtering

3.1 Introduction to Trend filtering

[WSST16] proposed the novel technique of trend filtering on graphs, which can adapt to local in-homogeneites as compared to l-2 norm based regularizations. It is also computationally efficient and falls into analysis framework of estimators thus, can easily yield complex extensions of the basic estimator by mixing penalties.

For a graph, $G = (V, E)$ with vertices $V = 1, 2, ..., n$ and undirected edges $E = e_1, e_2, ..., e_m$, if the variables observed over the node are $y = (y_1, y_2, ..., y_n) \in \mathcal{R}^n$ and if the input variables are assumed to be evenly spaced, [WSST16] defined k^{th} order graph trend filtering (GTF) estimate, $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2, ..., \hat{\beta}_n)$ as follows,

$$\hat{\beta} = \arg\min \frac{1}{2}||y - \beta||_2^2 + \lambda||D^{k+1}\beta||_1$$

In the above estimate, the penalty matrix D^{k+1} is $(k+1)^{th}$ order graph difference operator (GDO) such that,

$$D^{(k+1)T}D^{(k+1)} = L \tag{3.1}$$

where L is the Graph-Laplacian. Further, [WSST16] defines the 1^{st} ordered GDO as $D \in \{-1, 0, 1\}^{m \times n}$ as the orient incidence matrix i.e, for each edge $e_l = (i, j)$, the l^{th} row of D is:

$$D_l = (0, ..., \underset{\substack{\downarrow \\ i}}{1}, ..., \underset{\substack{\downarrow \\ j}}{-1}, ..., 0)$$

The higher order, GDOs are related to D as, $D^{k+1} = DD^k$. The L-1 norm penalty term in the objective of the GTF, while considering the first order GDO can be expressed as,

$$||D\beta||_1 = \sum_{\{i,j\} \in E} |\beta_i - \beta_j| \tag{3.2}$$

In the present work we extend the above definition of the GDO and propose graph trend filtering (GTF) as a contemporary approach to the Laplacian based manifold regularization, as shown in the next section.

3.2 L1-norm Manifold regularized MCM through Trend filtering - TF MCM

3.2.1 Defining the Graph Difference Operator (GDO)

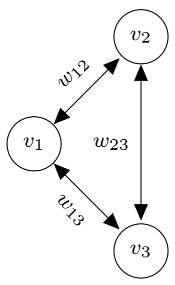
Manifold regularization is based on a penalty term which can help account for the intrinsic geometry of the marginal. Conventionally, [BNS06] proposed using Graph-Laplacian based penalty term as means to regularize the classifier over marginal's manifold. We in the present work, propose a graph trend filtering (GTF) based penalty term as means for manifold regularization. The choice of GTF over Graph-Laplacian based regularizer is due to the inherent advantages of GTF as described in the above section.
Inspired by [WSST16], for a graph $G = (V, E)$ we novely define the Graph Difference Operator(GDO) as $\Delta \in \{\{-w_{ij}^{1/2}\}, 0, \{w_{ij}^{1/2}\}\}^{m\times n}$ where $W = \{w_{ij}\}$ is the weighted adjacency matrix of the graph. Thus, for complete definition of Δ, for each edge $e_l = (i, j)$, with corresponding weight w_{ij} the l^{th} row of Δ is defined to be:

$$\Delta_l = (0, ..., \underset{\substack{\downarrow \\ i}}{w_{ij}^{1/2}}, ..., \underset{\substack{\downarrow \\ j}}{-w_{ij}^{1/2}}, ..., 0)$$

The above defined GDO is related to the Graph-Laplacian as follows,

$$\Delta^T \Delta = L \tag{3.3}$$

We further illustrate above definition of the GDO with an example of the following graph,



$$\implies \Delta = \begin{bmatrix} w_{13}^{1/2} & 0 & -w_{13}^{1/2} \\ w_{12}^{1/2} & -w_{12}^{1/2} & 0 \\ 0 & w_{23}^{1/2} & -w_{23}^{1/2} \end{bmatrix}$$

The functionality of this definition lies in the fact that the l1-norm penalty term in the objective of the GTF, using the our Δ while taking equation 3.2 as a reference, gives us:

$$||\Delta\beta||_1 = \sum_{\{i,j\}\in E} w_{ij}^{1/2} |\beta_i - \beta_j| \tag{3.4}$$

3.2.2 Manifold Regularization using Graph Trend Filtering

The general semi-supervised Manifold regularized learning framework as proposed by [BNS06] is as follows,

$$f^* = \underset{f \in \mathcal{H}_K}{\arg\min} \ \frac{1}{l}\sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A ||f||_A^2 + \gamma_I ||f||_I^2$$

where, f is a function in the reproducing kernel Hilbert space, $\mathcal{H}_K$. The penalty, $||f||_A^2$ is the ambient norm and the penalty, $||f||_I^2$ is the manifold regularizer. The manifold regularizer can be estimated using the Graph-Laplacian as, $||f||_I^2 = \sum_{i,j \ \in \ E} w_{ij}(f_i - f_j)^2 = f^T L f$ where, $W = \{w_{ij}\}$ is the weight adjacency matrix and L is the Graph-Laplacian. Using the Representor theorem the solution of the above posed optimization problem is, $f(x) = \sum_{j=1}^{l+u} \lambda_j K(x, x_j)$ [BNS06] where, K is the gram matrix.

$$\text{Now, } f = \begin{bmatrix} f(x_i) \\ \dots \\ f(x_{l+u}) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{l+u} \lambda_j K(x_i, x_j) \\ \dots \\ \sum_{j=1}^{l+u} \lambda_j K(x_{l+u}, x_j) \end{bmatrix} = K\lambda$$

$$\implies ||f||_I^2 = \sum_{i,j \ \in \ E} w_{ij}(f_i - f_j)^2 = f^T L f = \lambda^T K L K \lambda$$

where, E is the set of all edges of the graph. Now, Substituting 3.3 in above,

$$||f||_I^2 = \lambda^T K L K \lambda = \lambda^T K \Delta^T \Delta K \lambda = (\lambda K \Delta)^T \Delta K \lambda = ||\Delta K \lambda||^2$$

$$\implies ||f||_I = \sum_{i,j \ \in \ E} w_{ij}^{1/2} |f_i - f_j| = (f^T L f)^{1/2} = (\lambda^T K L K \lambda)^{1/2} = ||\Delta K \lambda||$$

Also, can be directly seen by, equation 3.4

$$\implies ||f||_I = ||\Delta K \lambda|| = ||\Delta f|| \tag{3.5}$$

We can also try to expand and see the $||f||_I$ in terms of EFS vectors,

$$||f||_I = \sum_{i,j \in E} w_{ij}^{1/2} || \sum_{s=1}^{l+u} \lambda_s K(x_i, x_s) - \sum_{s=1}^{l+u} \lambda_s K(x_i, x_s) ||$$

$$= \sum_{i,j \in E} w_{ij}^{1/2} | \sum_{s=1}^{l+u} \lambda_s \{K(x_i, x_s) - K(x_i, x_s)\} |$$

Now, inspired by the Graph trend filtering, we formulate the problem just by using the l-1 norm penalty term replacing the quadratic Graph-Laplacian based l-2 norm term. This, can also be done as the $||f||_I^2$ (laplacian term) is minimized in the objective and, for it's small values $||f||_I$ acts as it's upper bound and thus minimizing $||f||_I$ also minimizes $||f||_I^2$.
The resulting manifold-regularized semi-supervised learning framework based on Graph Trend filtering will thus be (using 3.6),

$$f^* = \underset{f \in \mathcal{H}_K}{\arg\min} \; \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A ||f||_A^2 + \frac{\gamma_I}{l+u} ||\Delta f||_1 \tag{3.6}$$

3.2.3 Trend Filtered MCM - TFMCM

Using the framework we derived in the above section(3.6) the Trend Filtered MCM (TFMCM) can be formulated as linear programming problem as follows,

$$\min_{h,q,b,\alpha} \quad h + \frac{1}{l}\sum_{i=1}^{l} q_i + \frac{\gamma_I}{u+l}||\Delta K \lambda||_1 \tag{3.7}$$

$$\text{such that ,}$$

$$h \geq y_i \Big(\sum_{j=1}^{l+u} \lambda_j \times K_{ij} + b \Big)$$

$$y_i \Big(\sum_{j=1}^{l+u} \lambda_j \times K_{ij} + b \Big) + q_i \geq 0$$

$$q_i \geq 0, \quad h \geq 1$$

$$\forall i = 1, 2, 3, , ..., l$$

The optimal solution is given by $f^*(x) = \sum_{j=1}^{l+u} \lambda_j^* K(x, x_j)$. The above formulation is a linear problem and have all the advantages of GTF viz. computational efficiency, can easily lead to complex extensions and thus, can have better generalized performance.
As a preliminary test, we tested the algorithm on artificial datasets and compared the obtained decision boundary with that in LapSVM and LapMCM. The datasets constructed were two-moons and two-circles, each with 200 samples out of which 198 were considered as unlabelled and a pair was considered labelled. The results are in the figure 3.1.
The results clearly show a tighter decision boundary of the obtained hyperplane classifier in the case of LapMCM and TFMCM. The results for TFMCM have low smoothness, which can be attributed to the use of first order GDO. Still, better complex extensions obtained by TFMCM as visible by small humps, in order to incorporate all the unlabelled points.
Graph-Laplacian based Manifold Regularization have an inherit disadvantage, when the manifold to be learned is non-smooth or the conditional varies non-smoothly over the

geodesics of the marginal, as written in [BNS06]. Thus, LapSVM and LapMCM may not be able to learn a hyperplane classifier conforming to complex extensions with sharp non-differentiabilities, while the TFMCM may be better immune to such disadvantage and thus, is expected to be a more robust algorithm.

We further in Chapter 6, use this hypothesis to formulate a TFMCM based regressor to approximate functions with complex manifolds.

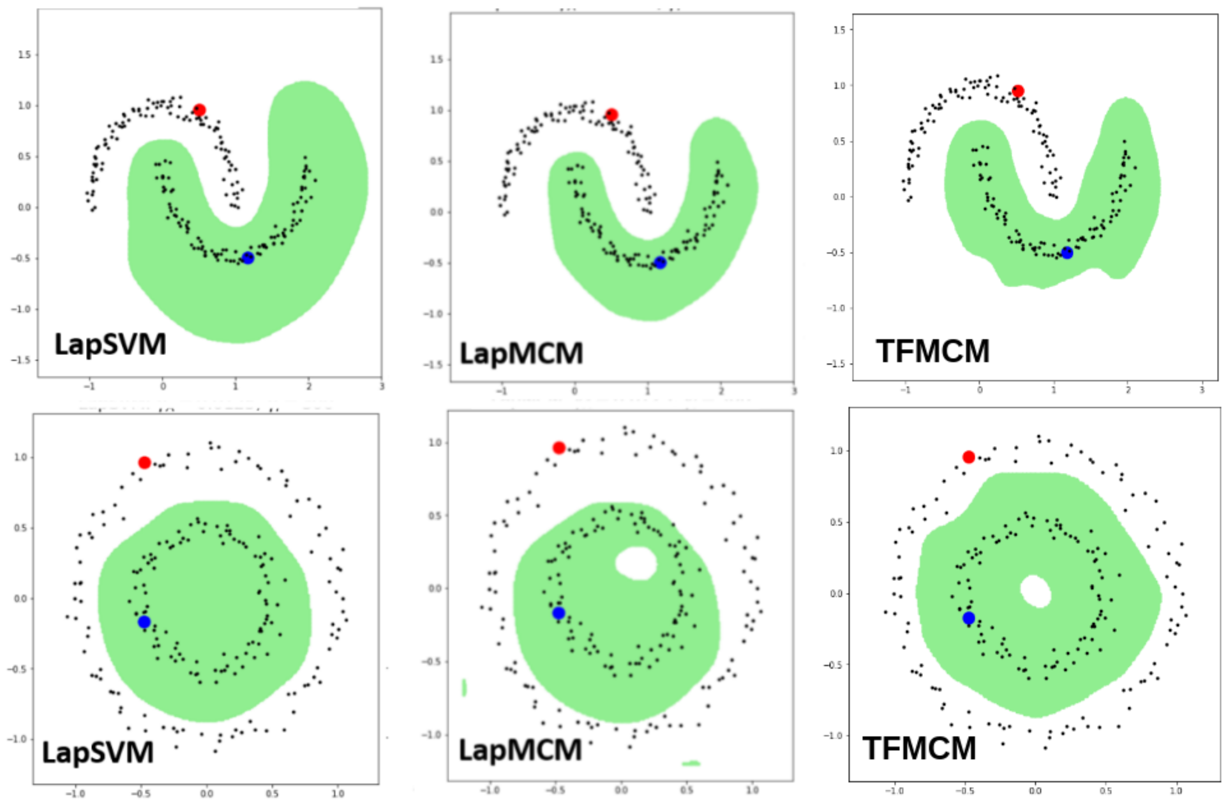


Figure 3.1: Performance of TFMCM on artificial datasets

In the next sections we explain and give the results of various experiments we performed involving LapMCM.

Chapter 4

Experimental Results and Discussions

In this section we give various experimental evidences, which confirm to better performance of LapMCM over the contemporary LapSVM. All the experiments were performed on Google's Colaboratory (Python 3 Google Compute Engine backend) with a 16GB RAM.

4.1 Exp-1: Role of the hyper-parameter "C" in LapMCM

LapMCM have a hyper-parameter "C", which determines the trade-off between the minimization of VC dimension and classifier accuracy. Lower the value of "C", tighter will be the bound on the VC dimension (smaller the value of h) and, higher the value of "C", more insignificant the h term becomes and the classifier approaches the LapSVM. Also, the value of "C" cannot be very small as for very small "C", the classifier becomes a hard margin classifier and thus, gives lower classification-accuracy.
We tried to illustrate and verify the above hypothesis through an experiment. We considered the UCI's German dataset and trained a LapMCM model. 400 of the samples were considered in the training set, out of which 20% were considered labelled samples, equally from each class. The values of the hyper-parameter were kept as, $\gamma_A = 10^{-4}$, $\gamma_I = 10^{-4}$, $p = 100$ and the Kernel chosen was RBF kernel which is described in the equation below:

$$K(x_i, x_j) = e^{-\frac{||x_i - x_j||^2}{2q^2}} \tag{4.1}$$

The kernel parameter was kept at a value $q = 2$ and models were trained for the values of "C" ranging from 10^{-6} to 10^{7}. The 5-fold cross validation accuracy, number of support vector and the optimal value of h were recorded and plotted. The resulting plot is shown in figure 4.1.
The plot cleary verifies the hypothesis, as the value of the tight bound on the VC dimension, h increases with the value of "C" thus, giving a more complex classifier (shown in red in figure 4.1). Also, the number of support vectors increase with the increase in the value of "C" (shown in blue in figure 4.1), which again points to the increasing complexity of the classifier. The accuracy (shown in green in figure 4.1) is found to be almost consistent, but decreases sharply, when the value of "C" is very small (which can be attributed to the objective becoming hard margin).
Thus, the value of the hyper-parameter, "C" must be chosen such that we get maximum

accuracy along with least number of support-vectors. This will ensure that the classifier obtained have minimal complexity along with good performance.

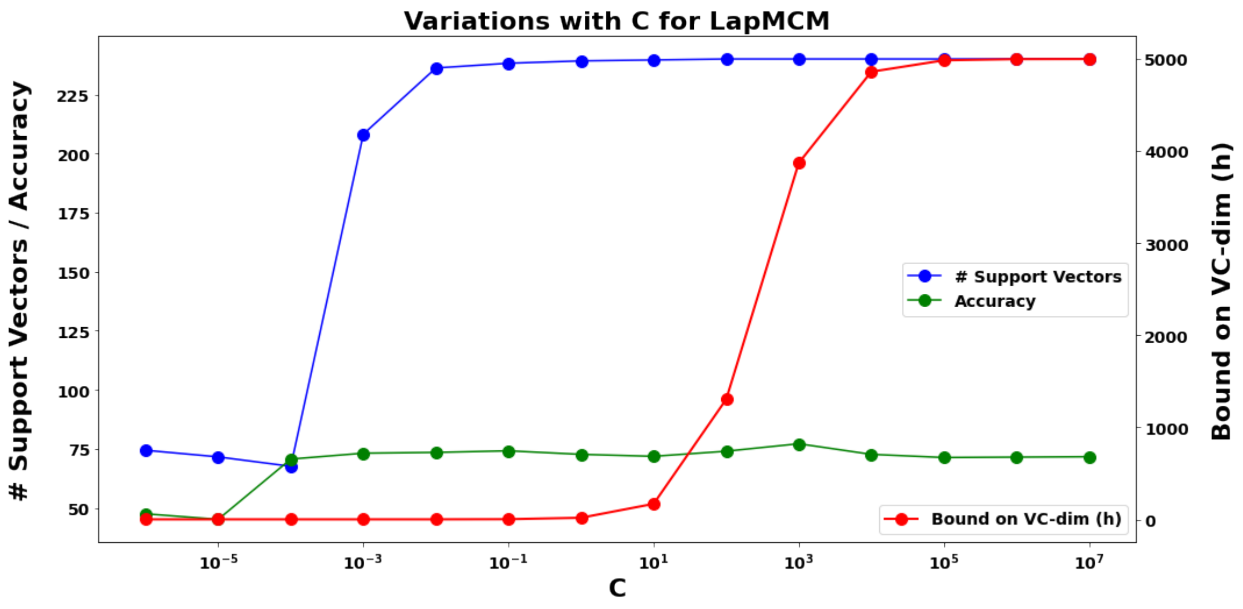


Figure 4.1: Role of the hyper-parameter "C" in LapMCM

4.2 Exp-2: Complexity of LapMCM Vs LapSVM

For this experiment we again took UCI's Australian and German datasets and trained various LapSVM and LapMCM models. The total number of data-points or samples were slowly added sequentially (i.e, more points were added slowly to the existing points) and models were trained to understand the effect of increasing data-points on the complexity of the classifier. 40% of the total data-points were considered labelled with equal number from each of the 2 binary classes. Rest of the 60% samples were considered unlabelled. The models were trained and the 5 fold cross validation accuracies and the number of support vectors were plotted, for increasing number of data points thus, giving curves of accuracies vs. data-points and number of support-vectors vs. data-points.

The hyper-parameter tuning was done using grid-search where for LapSVM, the values of both γ_A and γ_I were chosen from 10^{-6} to 10^{6} (to see the hyper-parameters in LapSVM, refer to [BNS06]). For LapMCM, the values of γ_A and γ_I were again chosen from 10^{-6} to 10^{6}, the value of the unconstraining parameter b was kept high at 2000 and, the value of "C" was chosen from 10^{-5} to 1. The Kernel used in both the cases were RBF, and the Kernel parameters was also tuned using grid-search with values ranging from 2^{-4} to 2^{4}. The hyper-parameters offering the maximum sum of accuracy and the number of support-vectors were chosen from the search-grid.

The resulting plot of the accuracy vs. data-points for Australian data-set is shown in figure 4.2. The accuracy is almost constant with the increasing number of data-points, and the accuracy for LapMCM is observed to be close to LapSVM, with slight improvement.

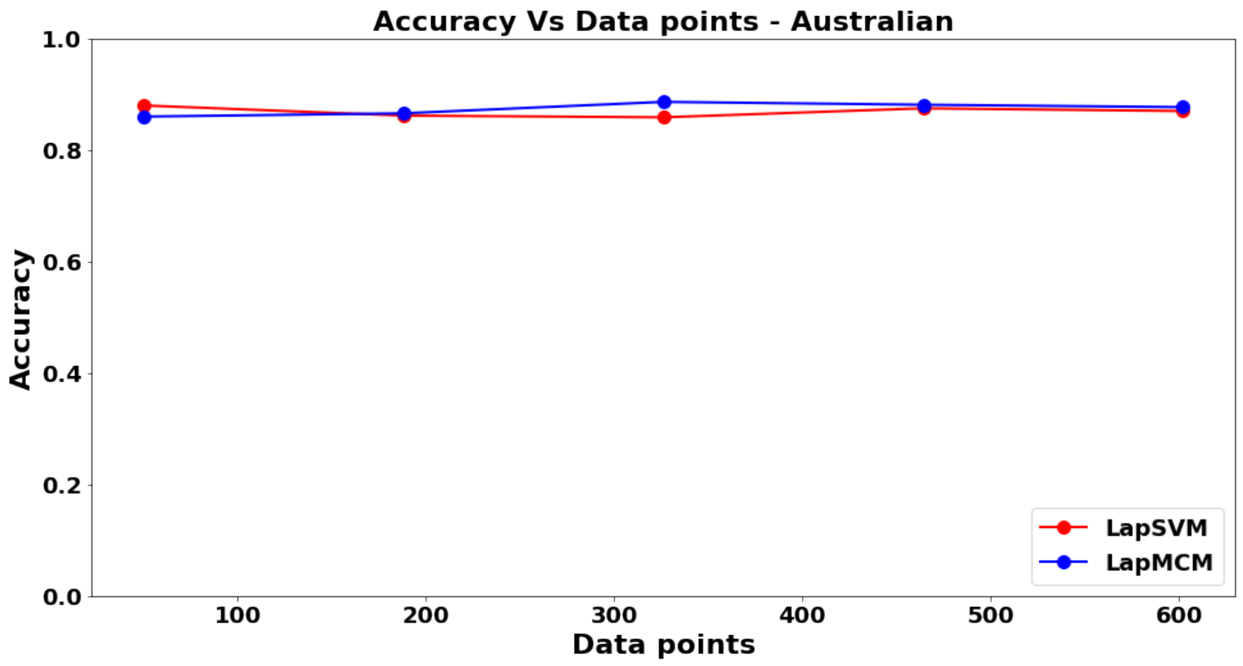


Figure 4.2: Accuracy vs datapoints for LapSVM and LapMCM on Australian dataset

For the same Australian dataset, the number of support-vectors with increasing number of data-points are shown in the figure 4.3. Clearly, the support-vectors from LapMCM are easy few than for LapSVM even when the accuracies of the two algorithm are very close. This, verifies the minimally complex nature of LapMCM attributed to the implicit minimization of the tight bound on the VC dimension.

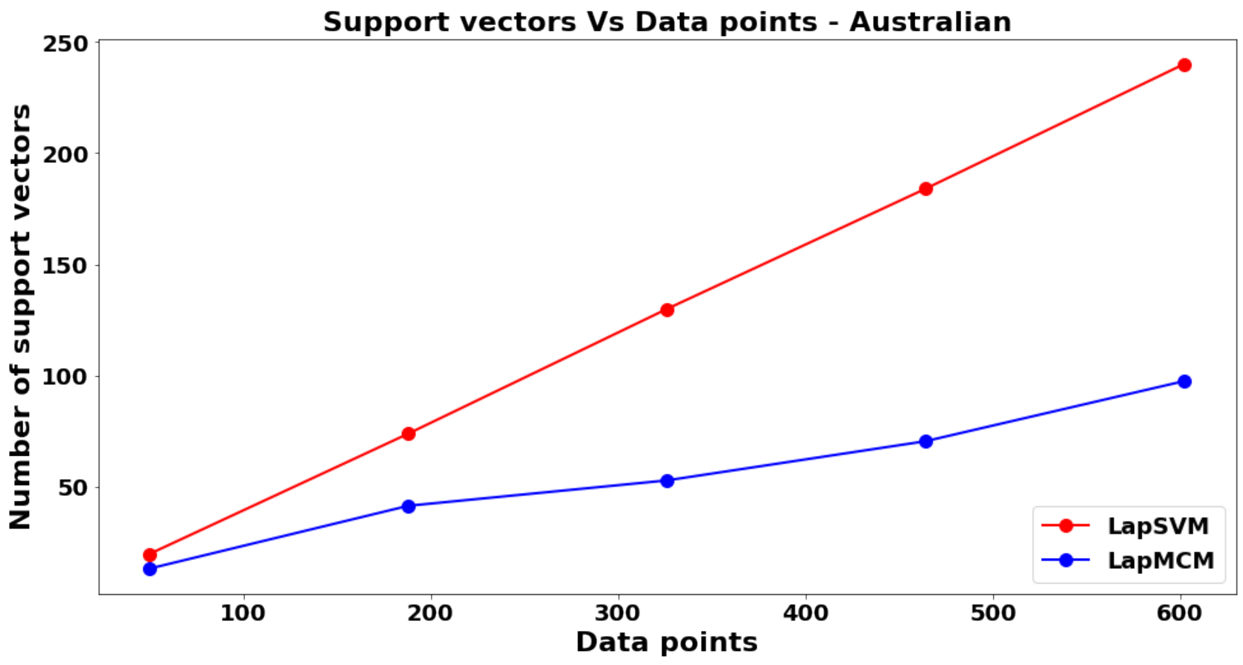


Figure 4.3: No. of support vectors vs datapoints - LapSVM & LapMCM - Australian

To further see the variations of the number of support-vectors with increasing data-points, the same experiment was performed on the German dataset (which have more number of training samples). The number of data-points were sequentially increased, by slowly adding more points to the existing points. The hyper-parameters were tuned in the same way as for the Australian dataset and the number of support-vectors were recorded and plotted, giving the plot shown in figure 4.4.

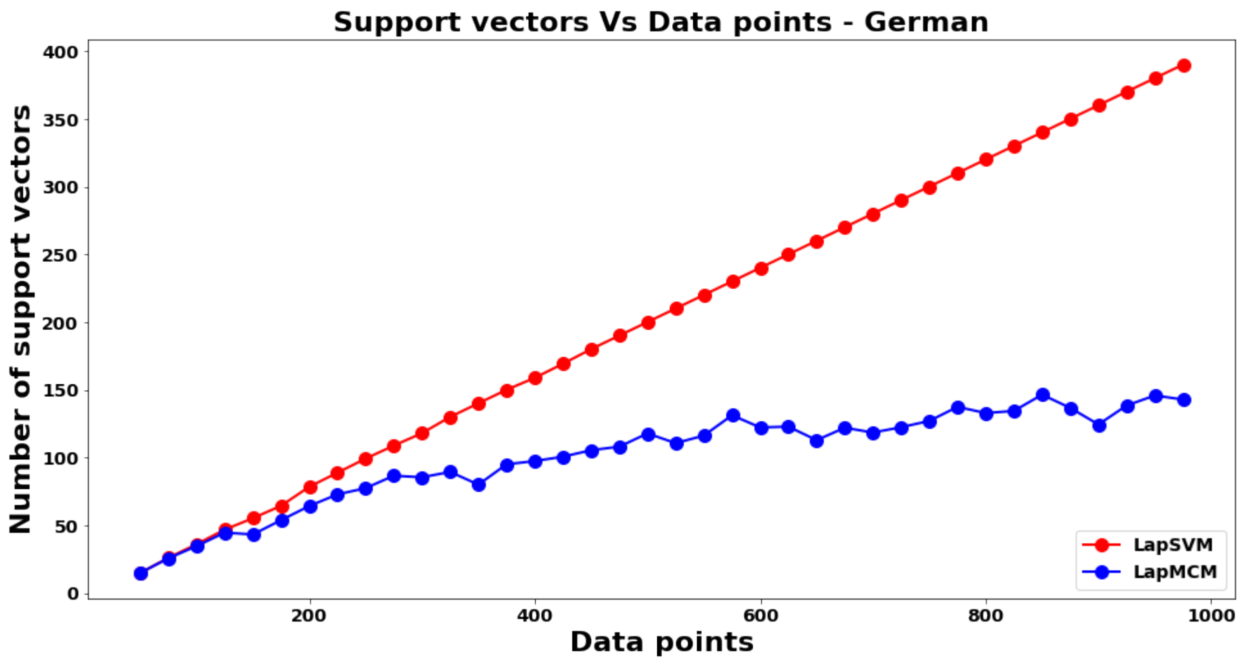


Figure 4.4: No. of support vectors vs datapoints - LapSVM & LapMCM - German

The above figure, again verifies the minimal complex nature of LapMCM. The number of support vectors are way fewer for LapMCM compared to LapSVM and they tend to saturate when the number of data-points become too large whereas for LapSVM, the number of the support-vectors increases continuously with increasing number of data-points.

4.3 Exp-3: LapMCM on UCI classification datasets

For this experiment, 4 datasets were taken from the UCI data repository viz. Australian, German, Ionosphere and Heart. All of the 4 datasets are binary classification datasets. We trained the LapSVM and LapMCM models on each of the dataset following the below described approach.

40% of the total samples in the dataset were considered as labelled, divided equally from both of the binary classes, and the rest 60% considered unlabelled. The hyper-parameters were tuned using grid-search and the values were varied in the same range as described in the section 4.2. The resulting 5 fold validation accuracies on the 4 datasets are described in the table 4.1.

Dataset	LapSVM	LapMCM
Australian (690×14×2)	0.869 ± 0.042	**0.875 ± 0.035**
German (1000×24×2)	0.700 ± 0.04 5	**0.725 ± 0.015**
Ionosphere (351×34×2)	0.878 ± 0.077	**0.909 ± 0.066**
Heart (270×13×2)	0.811 ± 0.032	**0.833 ± 0.031**

Table 4.1: Performance of LapMCM on UCI datasets with 40% labelled samples

The above results clearly illustrates the better performance of our LapMCM over LapSVM.

4.4 Exp-4: Effect of labelled samples on the performance

For this experiment, we again trained various LapSVM and LapMCM models with increasing number of labelled samples. The total samples were kept fixed but the number of labelled samples were slowly incremented. The hyper-parameters were tuned using grid-search with values in the same range as described in section 4.2. The 5 fold cross validation error-rate was plotted with the increasing number of labelled samples and the resulting plots for various UCI datasets are as follows:

4.4.1 For Australian dataset

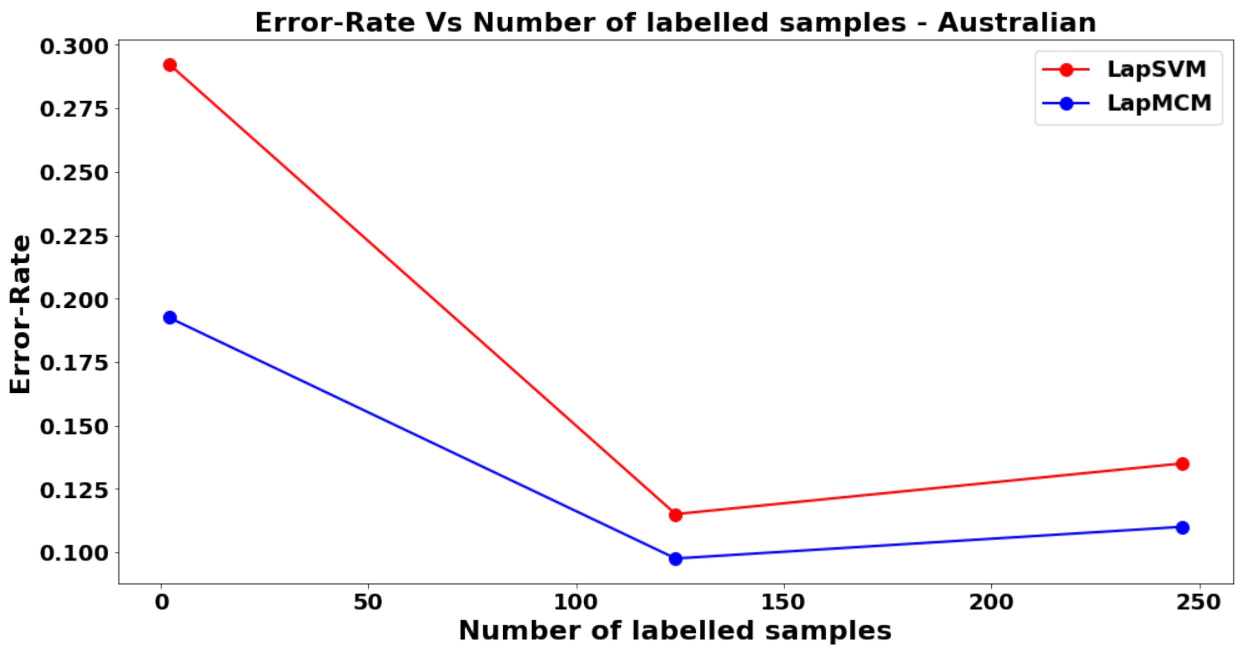


Figure 4.5: Error-rate vs No. labelled samples - LapSVM & LapMCM - Australian

4.4.2 For Heart dataset

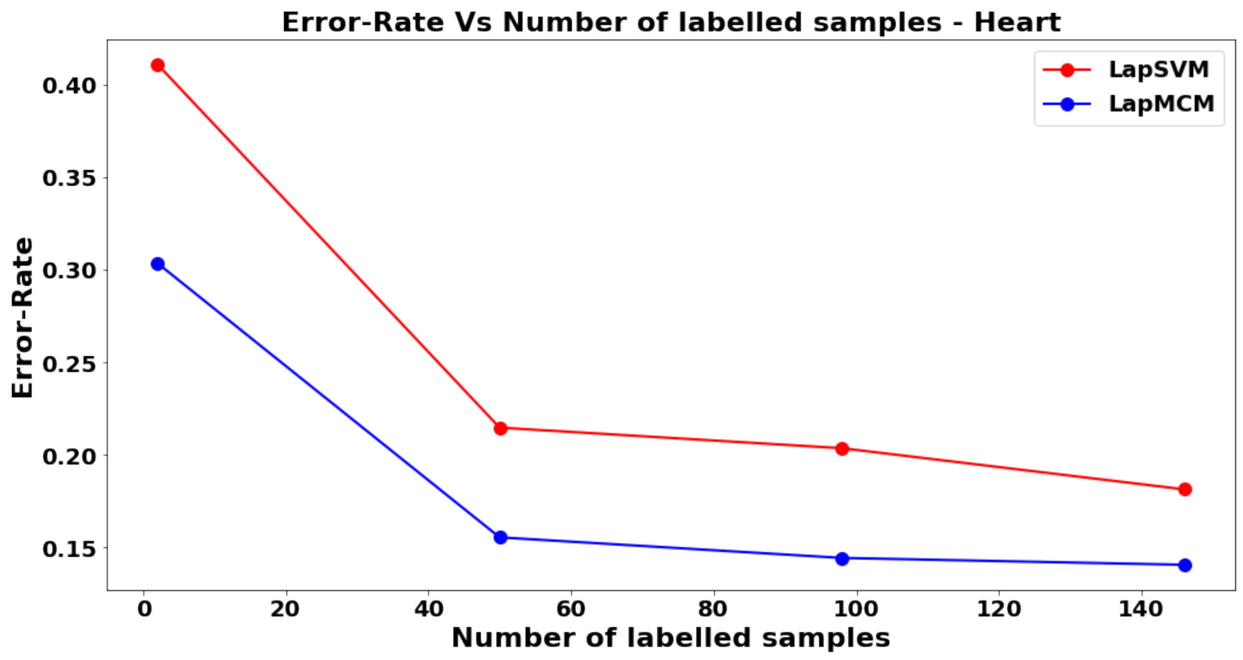


Figure 4.6: Error-rate vs No. labelled samples - LapSVM & LapMCM - Heart

4.4.3 For Ionosphere dataset

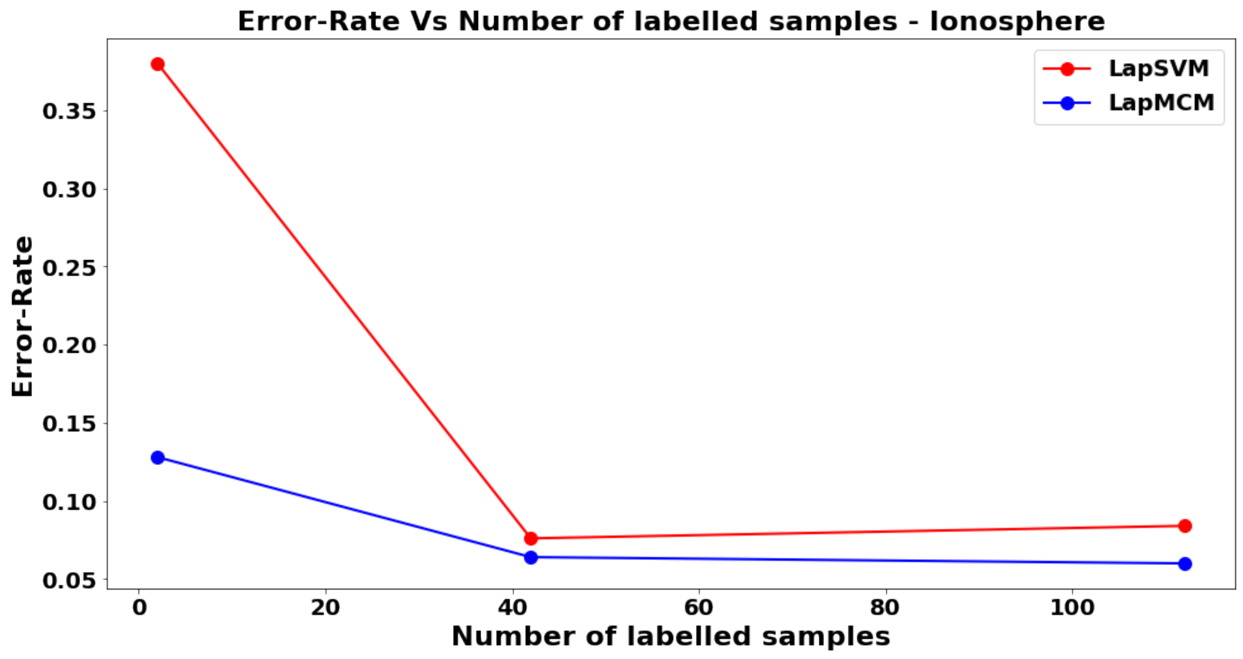


Figure 4.7: Error-rate vs No. labelled samples - LapSVM & LapMCM - Ionosphere

the above plots in figures 4.5, 4.6 and 4.7 clearly illustrates better performance of LapMCM over LapSVM for smaller percentage of labelled samples. The curves also confirms to the expected increase in classification-accuracies with increasing number of labelled samples.

In the following sections we introduce novel applications of our proposed algorithm over the above presented results, in order to illustrate competent real world application over and above better performance and complexity.

Chapter 5

Application - Feature Selection using Laplacian MCM

[Jay14] proposed feature selection as an application of hyperplane classifiers which tend to minimize the VC dimension. Our proposed LapMCM also tends to minimize the tight bound on VC dimension thus, we expanded this application of feature selection to LapMCM.
The number of features are also a tight bound on VC dimension thus, minimizing the tight bound on VC dimension in turn offers a minimal set of features. These features must easily be generalized thus, to asses the competence of the selected features, a Support Vector Machine (SVM) model can be trained on these features and the resulting performance can be a potential metric.
In order to perform such feature selection, we trained a LapMCM model with linear Kernel and computed the resulting hyperplane weights. The features corresponding to non-zero weights were selected and further employed in training a SVM model. Further, LapMCM is a semi-supervised learning algorithm thus, unlabelled samples are also considered in the above described process of feature selection thus, giving us a novel application of feature selection using unlabelled samples through LapMCM

5.1 Results of feature selection on Gene expression datasets

We chose to test the proposed process of feature selection on gene expression datasets which have a large number of features with relatively fewer samples. We employed our LapMCM and used only a single pair as labelled samples which considering all the other samples as unlabelled. The labelled and unlabelled samples were trained on a linear LapMCM and were used to select features, corresponding to non-zero weights. As described in the above process, we further trained a SVM model on the samples with above selected features and report the 5 fold cross validation accuracies. The results are in the table 5.1.

In the below table, ReliefF, FCBF and MCM results are borrowed from [Jay14] and thus the table does not present an absolute comparison. Still, the table verifies the feature selection application of LapMCM as, LapMCM is able to identify way fewer features than methods like ReliefF and FCBF while giving similar or better classification-accuracy (on SVM). Futher, the novelety of the approach lies in the fact that the the feature-selection is mainly using unlabelled features.

Datasets	**Features**				**Accuracies**			
(samples X dimensions)	**LapMCM**	**MCM**	**ReliefF**	**FCBF**	**LapMCM**	**MCM**	**ReliefF**	**FCBF**
Alon (62 × 2000)	25	41	896	1984	87%	83.8%	82.2%	82.1%
Shipp (77 × 7129)	35	51	3196	7129	97%	96.1%	93.5%	93.5%
Golub (72 × 7129)	67	47	2271	7129	96%	95.8%	90.3%	95.8%
Singh (102 × 12600)	66	81	5650	11619	91%	91.2%	89.2%	92.5%
Christensen (198 × 1413)	198	98	633	1413	99%	99.5%	99.5%	99.5%

Table 5.1: LapMCM based feature selection

In the following chapter, we propose another novel application of LapMCM and to illustrate the competence of our proposed TFMCM we employ the Trend Filtered version of LapMCM, i.e, TF MCM for the next application.

Chapter 6

Application - Function approximation using TFMCM

The problem we formulated was for binary classification but, this can easily be extended to a regression problem using the approach given by [BP03]. Consider the samples $\{x_i, y_i\}_{i=1}^{l}$ as l training samples with the regression output y_is. To formulate the regression problem as a corresponding binary classification problem, we considered a parameter ϵ and introduce a new feature column in the samples viz. $y_i - \epsilon$ and $y_i + \epsilon$. So, the labelled samples belonging to class -1 becomes, $\{(x_i, y_i - \epsilon)\}$ and the samples belonging to class 1 becomes, $\{(x_i, y_i + \epsilon)\}$. Learning a hyperplane classifier on these newly constructed samples corresponds to an equivalent regression sample, where the parameter ϵ determines the extent of fit of the regression.

[JCSS16] proposed a regression problem corresponding to MCM, we employ the same approach and propose a regression problem corresponding to TFMCM, which is given as follows:

$$\min_{h,q,b,\alpha} \quad h + \frac{C}{l}\sum_{i=1}^{l}(q_i^+ + q_i^-) + \frac{\gamma_I}{u+l}||\Delta K\lambda||_1 \tag{6.1}$$

$$\text{such that ,}$$

$$h \geq 1 \times \Big[(\sum_{j=1}^{l+u}\lambda_j \times K_{ij} + b) + \eta(y_i{+}\epsilon)\Big]$$

$$1 \times \Big[(\sum_{j=1}^{l+u}\lambda_j \times K_{ij} + b) + \eta(y_i{+}\epsilon)\Big] + q_i^+ \geq 1$$

$$h \geq -1 \times \Big[(\sum_{j=1}^{l+u}\lambda_j \times K_{ij} + b) + \eta(y_i{-}\epsilon)\Big]$$

$$-1 \times \Big[(\sum_{j=1}^{l+u}\lambda_j \times K_{ij} + b) + \eta(y_i{-}\epsilon)\Big] + q_i^- \geq 1$$

$$q_i^+, q_i^- \geq 0, \quad h \geq 1$$

$$\forall i = 1, 2, ..., l$$

where η is a new hyper-parameter and other symbols have usual meaning. The regressor function i.e, the approximated function is given by,

$$y = -\frac{1}{\eta}(\sum_{j=1}^{l+u} \lambda_j \times K_{ij} + b)) \tag{6.2}$$

Regression involves learning complex functions and manifold thus, using the hypothesis of complex generalization which we introduced in the chapter on TFMCM, we employed TFMCM to formulate the above regressor, instead of the graph-laplacian.

A sample result verifying the competence of our regression is as follows, we tried fitting a sine curve using our algorithm, as shown in figure 6.1. The blue points in the figure 6.1, refers to the labelled points and the orange points are predicted points, when we trained our TFMCM regressor using intermediate values of input as unlabelled samples.

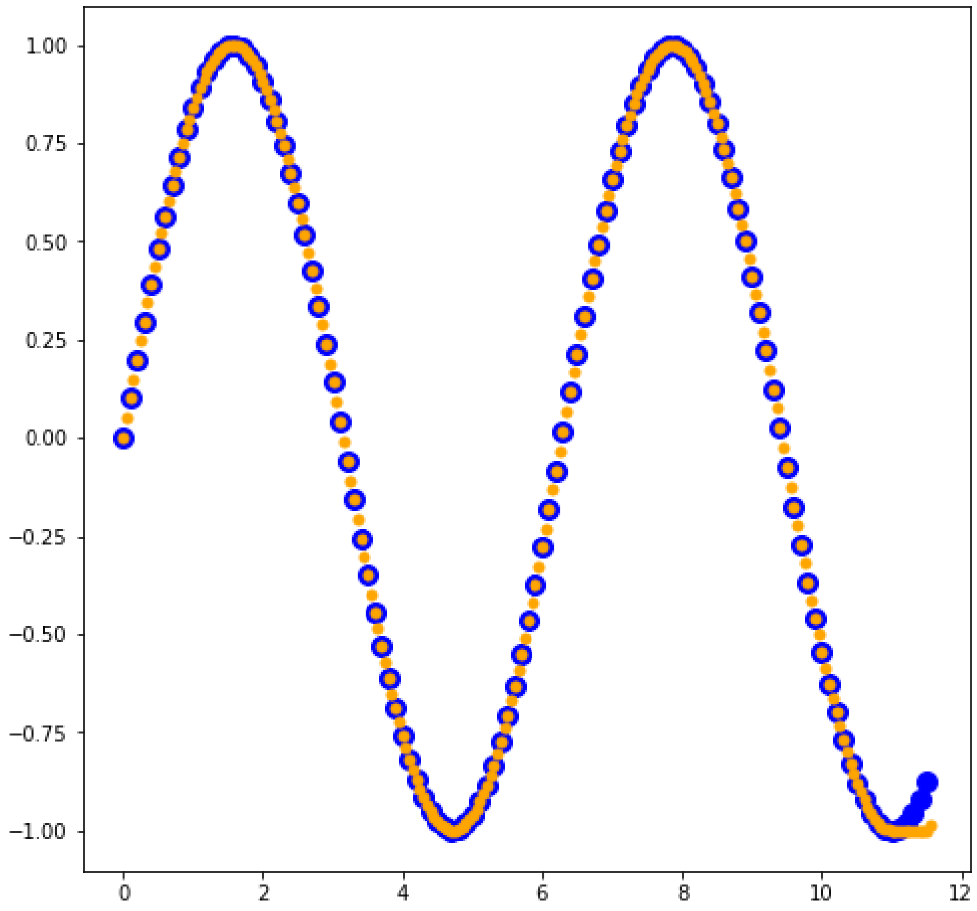


Figure 6.1: Results of TFMCM regressor on sine curve

6.1 Novel Application - Super-resolution using TFMCM

Super-resolution is a problem of constructing higher resolution images from low-resolution images. An image can be represented as a function $I(x, y)$, where (x, y) is the coordinate of the image pixel and $I(x, y)$ represents the intensity at the pixel with coordinates (x, y). Approximating the function $I(x, y)$ gives us a way to resolve low-resolution images, as the approximated function can be used to predict intensities at intermediate pixels. Conventionally, Deep learning based approaches had been applied to the problem of super-resolution but we present a novel approach of super-resolution by manifold approximation using unlabelled samples.
Consider an $p \times q$ image, then the sample set , $X_l = \{x_i, x_j\}$ where, $0 \leq i \leq p$ and $0 \leq j \leq q$. The regressor output at these samples are in the set $Y_L = \{I(x_i, x_j)\}$. We further introduce intermediate pixels as unlabelled samples i,e, $X_u = \{x_i \pm 0.5, x_j \pm 0.5\}$. We then train TFMCM regressor on these samples and compute the regressing hyperplane, which we use to predict the intensitites for intermediate pixel locations.
The process of super-resolution using TFMCM is described in the figure 6.2. We first take the original low-resolution image with original 3D manifold shown in the 2^{nd} part. We use the image pixels along with unlabelled intermediate pixels to learn a manifold which is shown as a scatter plot in the 3^{rd} part of the image. This learned manifold is further used to predict intermediate pixel intensities and to resolve the image.

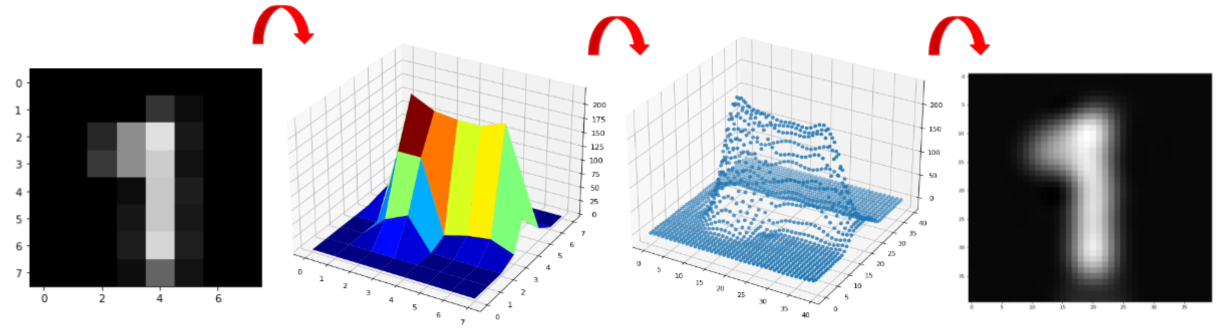


Figure 6.2: Process of super-resolution

6.2 Super-resolution results

Our algorithm is easily able to construct continuous high-resolution manifolds from discrete low-resolution manifolds as shown in figure 6.3.

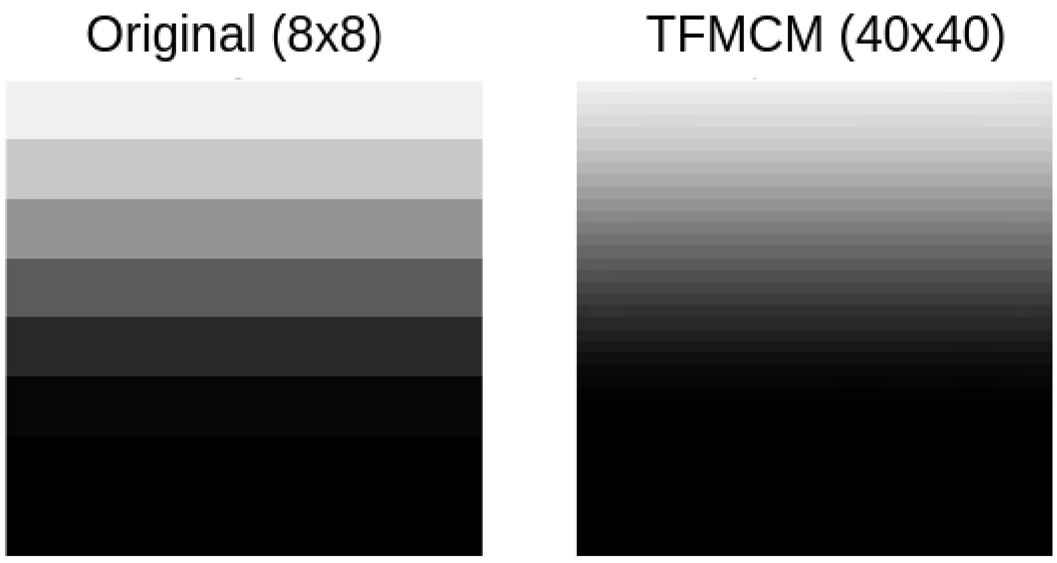


Figure 6.3: Results of super-resolution using TFMCM - 8x8 to 40x40

Our algorithm even gives better resolved generalizations as compared to methods like bilinear interpolation. Interpolation methods result into more diffused images while our algorithm gives more sharper results as illustrated in figures 6.4 and 6.5.

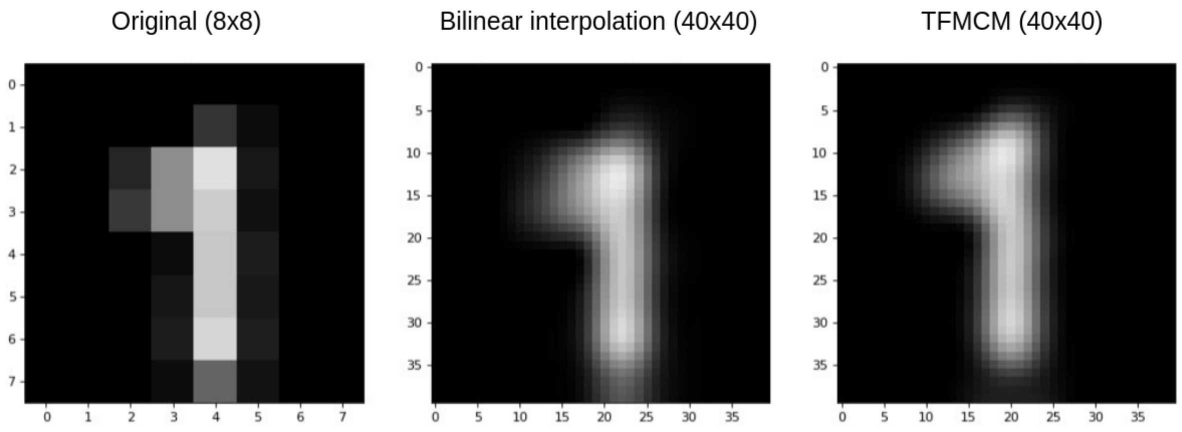


Figure 6.4: Comparison with Bilinear interpolation

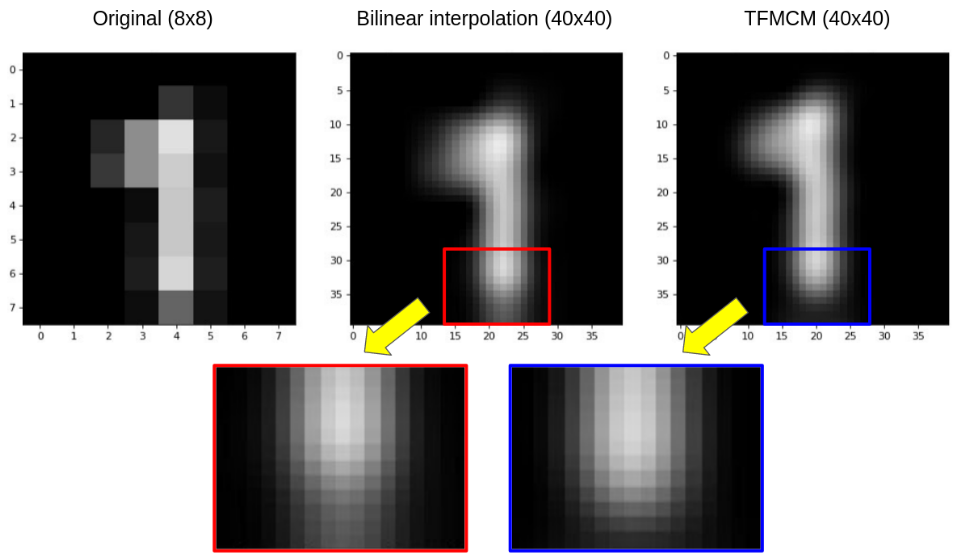


Figure 6.5: Better generalization in super-resolution using TFMCM

We also used the algorithm on 48 by 48 image and resolved it 5 times to a 240 by 240 image. Due to high computational cost and limited RAM of Google Colab, the images were super-resolved in parts of size 8 by 8. The image was divided into distinct frames of size 8 by 8 and each frame was parallely super-resolved to the size of 40 by 40. The resulting image too gives a better generalization and enhances features of the original image. The results are shown in figure 6.6.

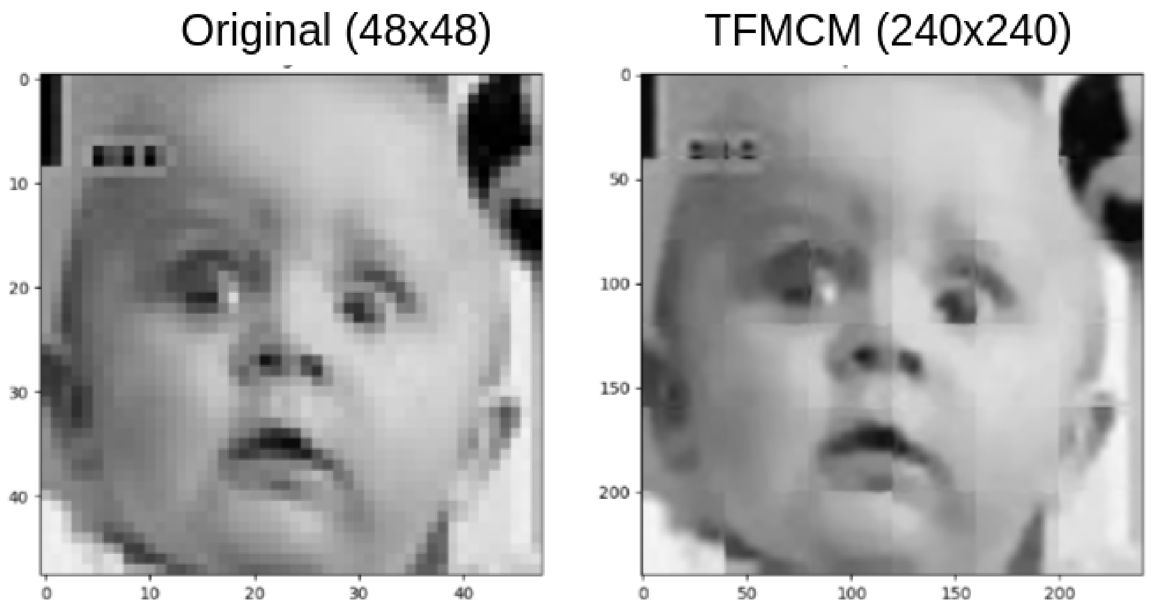


Figure 6.6: Super-resolution using TFMCM on 48x48 image

The algorithm also works flawlessly on colored images and enhances features tremendously.

For colored images each of the R, B and the G channels were super-resolved parallelly and were then cascaded and normalized (to avoid overflow), to create a colored image; figure 6.7 illustrates the same.

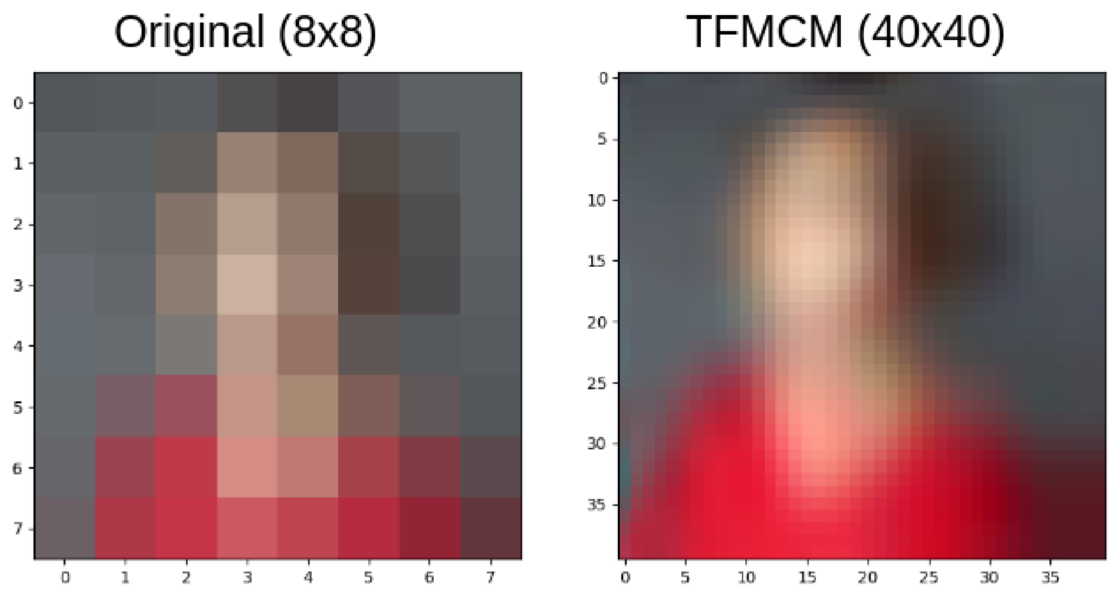


Figure 6.7: Super-resolution using TFMCM on colored images

Chapter 7

Conclusion and Future Extension of Work

We proposed a semi-supervised classification algorithm which minimizes the tight-bound on the VC dimesion. We also proposed a novel Graph Trend Filtering based framework for semi-supervised learning and Manifold regularization which offers better computational efficiency and ability to yield complex extensions or generalization. We verified the better generalized performance and minimal complexity of the proposed algorithm through various experiments. We further proposed novel real world applications of out algorithm and justified them with experimental evidences. The proposed applications include the novel feature-selection from unlabelled samples.
In future, we aim to publish the current work and further explore the introduced applications. We plan to develop an iterative algorithm, in order to train the TFMCM in the primal form thus, extending work to large scale. We also plan on making Structured Graph Learning(SGL) adaptive and the following are the details of the problem.

7.1 Adaptive Structured Graph Learning

7.1.1 Literature Review

Extracting information from raw data into a graphical representation has a multitude of applications in practical life, and hence is an extensively researched field in Graph Learning. The Graph Learning algorithm that is the focus of this study is the Structured Graph Learning (SGL) algorithm.

Structured Graph Learning

[KYdMCP20] formulated a versatile algorithm to learn from a variety of graph structures such as multi-component, bipartite and other regular graphs. The SGL algorithm employs the identifying properties of given graphs by imposing constraints on the general graph structure.
As per the spectral graph theory, the structural constraints for the concerned graph can be exerted by applying spectral constraints from the Graph Adjacency Matrix (SGA), or the Graph Laplacian Matrix (SGL), or from both (SGLA). So SGL reduces the graph-learning problem to an optimization problem the spectral constraints are imposed on the Graph Laplacian Matrix Θ:

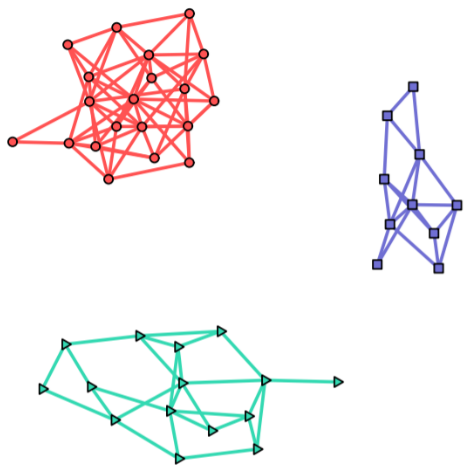
$$\begin{aligned} \max_{\Theta} \quad & \log \operatorname{gdet}(\Theta) - \operatorname{tr}(\Theta S) - \alpha h(\Theta) \\ \text{s.t.} \quad & \Theta \in \mathcal{S}_{\Theta},\ \lambda(\mathcal{T}(\Theta)) \in \mathcal{S}_{\mathcal{T}} \end{aligned} \tag{7.1}$$

where S is the sample covariance matrix (SCM), tr() is the operator for matrix's trace, gdet() is the generalized determinant, that is the product of non-zero eigenvalues, and $\mathcal{S}_{\mathcal{T}}$ is the set of spectral constraints imposed on the Graph Laplacian Matrix's eigen values to constrain the output graph structure.

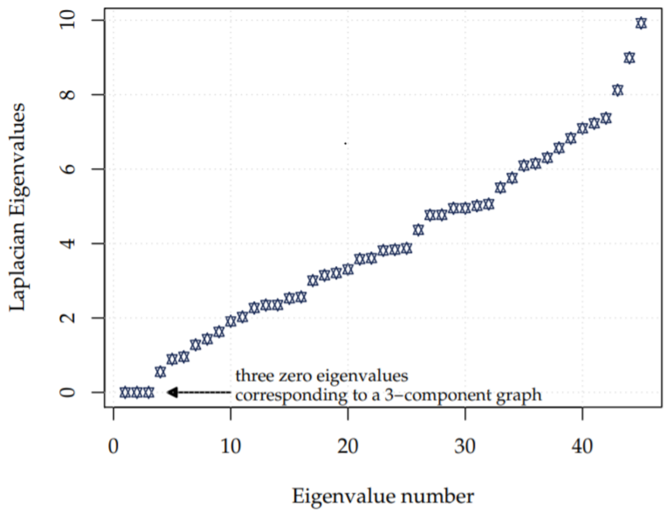
Performance on Multi-component Graphs

A graph is said to have k components if the nodes can be separated into k disjoint subsets, i.e. no two nodes from different clusters are networked to one another by an edge. The structural properties of any given k-component graph is embedded in its Graph Laplacian Matrix.

The Graph Laplacian Matrix for a multi-component graph will have precisely k non-zero eigenvalues for k components. Hence, by subjecting Θ to the constraint of having precisely k non-zero eigenvalues, a strict k-component graph will be imposed on the optimization problem. Fig 7.1 demonstrates how the first three eigenvalues of Θ being zero provides information for spectral constraints pertaining to the SGL algorithm's optimization problem.



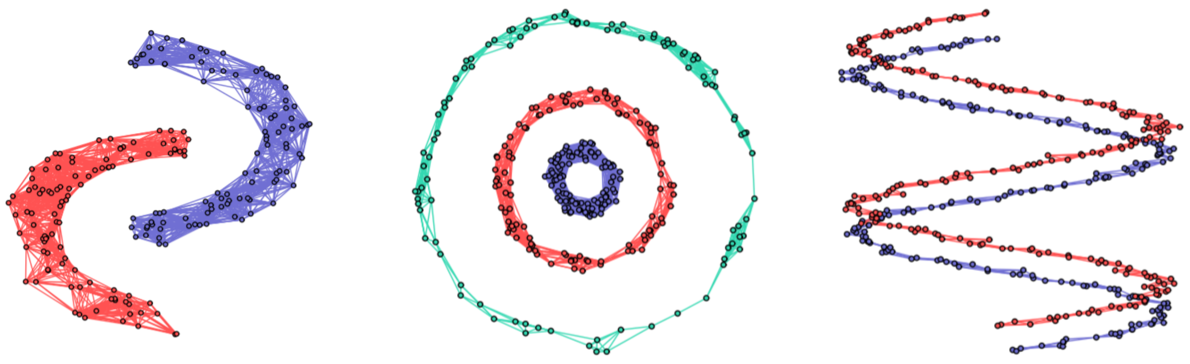
(a) A 3-component graph.



(b) Eigenvalue distribution of a 3-component graph.

Figure 7.1: 3-component graph and corresponding eigenvalue distribution [KYdMCP20]

SGL has better clustering performance than other similar methods such as K-means clustering with cleaner outputs as compared to GLasso [KYdMCP19]. The performance of SGL with renowned datasets of low dimensions are exhibited in Figure 7.2.



(a) 2 moons (b) 3 circles (c) 2 helices

Figure 7.2: Near accurate clustering performance of SGL on different graph structures

Adaptive Graph Learning

Structured Graph Learning can be implemented by imposition of Laplacian spectral constraints. For this formulation, we introduce the Graph Laplacian Operator :

Definition: : $\mathbf{w} \in R_{+}^{p(p-1)/2} \rightarrow \mathbf{w} \in R^{p\times p}$ such that:

$$|\mathbf{w}_{ij}| = \begin{cases} -w_{i+d_j} & if\ i > j \\ |\mathbf{w}_{ji}| & if\ i < j \\ \Sigma_{i\neq j}|\mathbf{w}_{ij}| & if\ i = j \end{cases}$$

where $d_j = -j + \frac{j-1}{2}(2p - j)$.

The optimization problem in (7.1) can now be written as the given Laplacian spectral constrained optimization problem [KYdMCP19]:

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\lambda},U} \quad & -\log \operatorname{gdet}(U\operatorname{Diag}(\boldsymbol{\lambda})U^T) + \operatorname{tr}(K\mathbf{w}) + \frac{\beta}{2}||\mathbf{w} - U\operatorname{Diag}(\boldsymbol{\lambda})U^T||_F^2 \\ \text{s.t.} \quad & \mathbf{w} \geq 0,\ \ \boldsymbol{\lambda} \in \mathcal{S}_\lambda, U^TU = I \end{aligned} \tag{7.2}$$

where $\mathbf{w}$ is the Laplacian Matrix allowing the decomposition $\mathbf{w} = U\operatorname{Diag}(\boldsymbol{\lambda})U^T$. The penalty term enforces spectral information for the desired graph structure. The above optimization problem can be solved by the block successive upper-bound minimization framework (BSUM) [RHL12], that updates each block in a sequential manner without disturbing the other blocks.

Update of w

For the $(t+1)^{th}$ iteration, treating $\mathbf{w}$ as variable with other parameters fixed, the subproblem for $\mathbf{w}$ can be simplified to:

$$\min_{\mathbf{w}\geq 0} \quad f(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||_F^2 - \mathbf{c}^T\mathbf{w} \tag{7.3}$$

where $c = \ * (U\text{Diag}(\boldsymbol{\lambda})U^T - \beta^{-1}K$, making it a strictly convex problem. Due to the non-negativity contrainst on $\mathbf{w}$, a majorized function can be derived which will give the optimal solution as follows, after applying KKT optimality conditions.

$$\mathbf{w}^{t+1} = \left(\mathbf{w}^t - \frac{1}{2p}\nabla f(\mathbf{w}^t)\right)^+ \quad , \ where\ (a)^+ = max(a,0) \tag{7.4}$$

Update of U

Keeping U as variable with other parameters constant, the sub-problem for U becomes:

$$\max_{U} \quad \text{tr}(U^T\mathbf{w}U\text{Diag}(\boldsymbol{\lambda})) \quad \text{s.t. } U^TU = I_q \tag{7.5}$$

Applying KKT optimality conditions to (7.5), the solution comes out to be, (where the $n-k$ eigenvectors of $\mathbf{w}$ are in increasing order of the magnitudes of their eigenvalues):

$$U^{t+1} = \text{eigenvectors}(\mathbf{w})[k+1:p] \tag{7.6}$$

Update of λ

For $\boldsymbol{\lambda}$, the optimization sub-problem can be rewritten as

$$\min_{c_1\leq\lambda_1\leq\ldots\leq\lambda_q\leq c_2} \quad -\sum_{i=1}^{q}\text{log}\lambda_i + \frac{\beta}{2}||\boldsymbol{\lambda} - \mathbf{d}||_2^2 \tag{7.7}$$

where $\boldsymbol{\lambda} = [\lambda_1, ..., \lambda_q]^T$ and $\mathbf{d} = [d_1, ..., d_q]^T$, d_i being the i^{th} diagonal element of $\text{Diag}(U^T(\mathbf{w})U)$. The solution to this can be derived by applying KKT optimality conditions, as (7.7) is a convex optimization problem.

Thus, the SGL algorithm can be summarized as follows [KYdMCP19]:

Algorithm 2: SGL

```
  Input: SCM $\mathbf{S}$, $k, c_1, c_2, \beta$
1 $t \leftarrow 0$
2 while Stopping criteria not met do
3     Update $\mathbf{w}^{t+1}$ as given in (7.4
4     Update $U^{t+1}$ as given in (7.6)
5     Update $\boldsymbol{\lambda}^{t+1}$ as discussed in (7.7)
6     $t \leftarrow t + 1$
7 end
  Output: $\widehat{\Theta} = \mathbf{w}^{t+1}$
```

The complete update step has a complexity of $O(p^3)$, which can be inefficient for adding new nodes to large graph structures. We aim to reduce this time complexity to provide a more robust version of SGL that incorporates traits of Adaptive Graph Learning as well. As the optimization problem involved here is an eigen-decomposition problem, we seek to emulate results of out-of-sample extensions for semi-unsupervised learning algorithms by representing the SGL algorithm as learning eigen-functions of a data-dependent kernel [BPV+04].

Appendix A

Code and Derivations

A simple implementation of LapMCM is appended below:

```python
import numpy as np
import torch
import pandas as pd
from scipy.spatial.distance import cdist
from sklearn.neighbors import kneighbors_graph
from scipy import sparse
import time

class LapMCM(object):
    def __init__(self,opt):
        self.opt=opt

    def fit(self,X,Y,X_u):
        #construct graph
        self.X=np.vstack([X,X_u])
        Y=np.diag(Y)
        if self.opt['neighbor_mode']=='connectivity':
            W = kneighbors_graph(self.X, self.opt['n_neighbor'], mode='connectivity',include_self=False)
            W = (((W + W.T) > 0) * 1)
        elif self.opt['neighbor_mode']=='distance':
            W = kneighbors_graph(self.X, self.opt['n_neighbor'], mode='distance',include_self=False)
            W = W.maximum(W.T)
            W = sparse.csr_matrix((np.exp(-W.data**2/4/self.opt['t']),W.indices,W.indptr),shape=(self.X.shape[0],self.X.shape[0]))
        else:
            raise Exception()

        # Computing Graph Laplacian
        L = sparse.diags(np.array(W.sum(0))[0]).tocsr() - W
        L = L/np.max(L)

        # Computing K with k(i,j) = kernel(i, j)
        K = self.opt['kernel_function'](self.X,self.X,**self.opt['kernel_parameters'])

        l=X.shape[0]
        u=X_u.shape[0]

        # Creating matrix J [I (l x l), 0 (l x (l+u))]
        J = np.concatenate([np.identity(l), np.zeros(l * u).reshape(l, u)], axis=1)

        # Computing "almost" alpha
        almost_alpha = np.linalg.inv(2 * self.opt['gamma_A'] * np.identity(l + u) + ((2 * self.opt['gamma_I']) / (l + u) ** 2) * L.dot(K)).dot(J.T).dot(Y)

        # Computing Q
        R = J.dot(K.dot(np.linalg.inv(2 * self.opt['gamma_A'] * np.identity(l + u) + ((2 * self.opt['gamma_I']) / (l + u) ** 2) * L.dot(K)).dot(J.T)))
        P = Y.dot((R+(1/self.opt['p'])*np.identity(l)).dot(Y))
        ol = np.ones((l,l))
        zl = np.zeros((l,l))
        M1 = np.concatenate((np.concatenate((ol, zl), axis=1), np.concatenate((zl, zl), axis=1)), axis=0)
        M2 = np.concatenate((np.concatenate((zl, zl), axis=1), np.concatenate((zl, ol), axis=1)), axis=0)
        N = np.concatenate((-1*np.identity(l), np.identity(l)), axis=1)
        Q = M1 + (l/self.opt['C'])*M2 + (N.T).dot(P.dot(N))
        Q = (Q+Q.T)/2
        Q = Q/np.max(Q)
        Q = self.eig_transform(Q)

        del W, L, K, J, M1, M2, zl, ol, P, R

        #Sequentially updating on a GPU
        def optimize(Q, l, itr):
            u = torch.zeros(2*l, dtype=torch.float32)
            device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
            u.to(device)
            Q_ = torch.tensor(Q, dtype=torch.float32)
            Q_.to(device)

            for k in range (0,2*l):
                for i in range (0, itr):
                    print(f"\rFor k: {k}/{2*l}; Iteration : {i}/{itr}", end="")
                    if k<l:
                        u[k] = u[k] + (-1*Q_[k,:].T.dot(u))/Q_[k,k]
                        if (u[k]>1): u[k] = 1
                    else:
                        u[k] = u[k] + (1-Q_[k,:].T.dot(u))/Q_[k,k]
                        if (u[k]>1): u[k] = 1
```

```
                            if (u[k]<0): u[k] = 0
                return u.detach().numpy()

        # ===== Solving =====

        tick = time.time()
        beta_hat = optimize(Q, 1, 500)
        tock = time.time()
        #print("Time taken in optimization: ", round(tock-tick, 6), "sec")
        #print()

        # Computing final alpha
        self.alpha = almost_alpha.dot(N.dot(beta_hat))
        del almost_alpha, Q

        self.b=(1/self.opt['p'])*np.sum(Y.dot(N.dot(beta_hat)))

        return self.alpha

    def decision_function(self,X):
        new_K = self.opt['kernel_function'](self.X, X, **self.opt['
   kernel_parameters'])
        f = np.squeeze(np.array(self.alpha)).dot(new_K)
        return f+self.b

    def eig_transform(self, Q):
        w,_ = np.linalg.eig(Q)
        if(np.sum(w<0)!=0):
            Q = Q-np.real((np.min(w))*np.identity(Q.shape[0]))
        return Q

def rbf(X1,X2,**kwargs):
    return np.exp(-np.square(cdist(X1,X2))/(2*kwargs['p']**2))

def lin(X1,X2,**kwargs):
    return np.dot(X1.T, X2)
```

Bibliography

[BNS06] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[BP03] Jinbo Bi and Kristin P.Bennett. A geometric approach to support vector regression. *Neurocomputing*, 55:79–108, 2003.

[BPV$^+$04] Yoshua Bengio, Jean-françcois Paiement, Pascal Vincent, Olivier Delalleau, Nicolas Roux, and Marie Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.

[Bur98] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[Jay14] Jayadeva. Feature selection through minimization of the vc dimension. *Preprint submitted to ArXiV*, 2014.

[Jay15] Jayadeva. Learning a hyperplane classifier by minimizing an exact bound on the vc dimension. *Neurocomputing*, 149:683–689, 2015.

[JCSS16] Jayadeva, Suresh Chandra, Sanjit S.Batra, and Siddarth Sabharwala. Learning a hyperplane regressor through a tight bound on the vc dimension. *Neurocomputing*, 171:1610–1616, 2016.

[JJRC12] Sachindra Joshi, Jayadeva, Ganesh Ramakrishnan, and Suresh Chandra. Using sequential unconstrained minimization techniques to simplify svm solvers. *Neurocomputing*, 77:253–260, 2012.

[KYdMCP19] Sandeep Kumar, Jiaxi Ying, Jose Vinicius de Miranda Cardoso, and Daniel Palomar. Structured graph learning via laplacian spectral constraints. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[KYdMCP20] Sandeep Kumar, Jiaxi Ying, JosÃ© VinÃcius de M. Cardoso, and Daniel P. Palomar. A unified framework for structured graph learning via spectral constraints. *Journal of Machine Learning Research*, 21(22):1–60, 2020.

[RHL12] Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23, 09 2012.

[Vap98] V Vapnik. *A Tutorial on Support Vector Machines for Pattern Recognition*, volume 2. 1998.

[WSST16] Yu-Xiang Wang, James Sharpnack, Alexander J. Smola, and Ryan J. Tibshirani. Trend filtering on graphs. *Journal of Machine Learning Research*, 17:1–41, 2016.