A Graph-Theoretic Clustering Algorithm based on the Regularity Lemma and Strategies to Exploit Clustering for Prediction

Shubhendu Trivedi

Submitted in partial fulfillment of the

requirements for the degree

of Master of Science

in the Department of Computer Science

Worcester Polytechnic Institute

2012

Thesis Approved:

Dr. Neil T. Heffernan (Thesis Advisor)

Dr. Gábor N. Sárközy (Thesis Advisor)

Dr. Sonia Chernova (Thesis Reader)

Dr. Craig E. Wills (Dept. Head)

The mind can be highly delighted in two ways: by perception and conception. But the former demands a worthy object, which is not always at hand, and a proportionate culture, which one does not immediately attain. Conception, on the other hand, requires only susceptibility: it brings its subject-matter with it, and is itself the instrument of culture.

Dichtung und Wahrheit

Dedicated to Ritika

Acknowledgments

I would like to express my deepest gratitude for all who have been directly or indirectly responsible for this work. To start with I would like to thank my great advisers: Professor Neil Heffernan and Professor Gábor Sárközy for their constant support and encouragement. Professor Heffernan has had a big hand by indirectly moulding what I worked on by containing my overtly enthusiastic disposition so that it could be channelled into something more useful and productive. Professor Sárközy for the wise words and the constant counselling about research and life in general. Thanks to my Thesis Reader Professor Sonia Chernova for being so prompt and helpful. A very special thanks to Prof. Glynis Hamel for her help in getting me subjects that I could manage to TA for and all the Professors that I assisted in some course (Prof. Joshua Guttman, Prof. Sárközy, Prof. Hugh Lauer and Prof. Heffernan). I am also thankful to Prof. Hugh Lauer for his kind words of encouragement about my work and not to mention - making the experience of TAing for a Systems course (and yet maintaining the tempo for research) so enjoyable. Thanks to Zach Pardos for the innumerable conversations about work and persevering with my countless digressions. I am also grateful to Fei Song for many insightful discussions on some of this work and for his both direct and indirect help in improving my understanding of many ideas. I am also grateful to Professor Stanley Selkow for many sharp insights and meetings that often changed my way of looking at things. Thanks to Professor Carolina Ruiz for her support in harder times.

A special thanks to Mike Voorhis and John Leveillee for always being there with help for both my research computing issues and those with courses that I was assigned to assist. I am also grateful to Diane Baxter for her help in getting a lot of paper work done quickly and making it fun. I am also grateful to all the other faculty members in the CS Department for making the experience and the time that went into this work so much fun. Thanks to

my room-mates Adinath and Tanmay for putting up with my weird work schedules and sleeping hours.

Further afield I would like to thank my school physics teacher Mrs Bhawana Danak for discussing some issues back in high school that got me interested in spectral stuff to start with.

Lastly, but more importantly, thanks to my parents, my brother and sister and all my friends for their constant love and support. Finally, I would like express my deepest gratitude to Ritika, without whom none of this could have been possible.

ABSTRACT

The fact that clustering is perhaps the most used technique for exploratory data analysis is only a semaphore that underlines its fundamental importance. The general problem statement that broadly describes clustering as the identification and classification of patterns into coherent groups also implicitly indicates it's utility in other tasks such as supervised learning. In the past decade and a half there have been two developments that have altered the landscape of research in clustering: One is improved results by the increased use of graph theoretic techniques such as spectral clustering and the other is the study of clustering with respect to its relevance in semi-supervised learning i.e. using unlabeled data for improving prediction accuracies. In this work an attempt is made to make contributions to both these aspects. Thus our contributions are two-fold: First, we identify some general issues with the spectral clustering framework and while working towards a solution, we introduce a new algorithm which we call "Regularity Clustering" which makes an attempt to harness the power of the Szemerédi Regularity Lemma, a remarkable result from extremal graph theory for the task of clustering. Secondly, we investigate some practical and useful strategies for using clustering unlabeled data in boosting prediction accuracy. For all of these contributions we evaluate our methods against existing ones and also apply these ideas in a number of settings.

Table of Contents

Introduction **1**

0.1 Contributions and Overview 2

I Preliminaries **5**

1 Preliminaries **6**

1.1 Clustering 6

1.2 Spectral Clustering 7

1.2.1 Self-Tuning Spectral Clustering 12

II Regularity Clustering **14**

2 Motivation to Regularity Clustering **15**

2.1 Introduction 15

2.2 Clustering 17

2.3 Prior Attempts to Use the Regularity Lemma for Practical Applications . . 18

3 The Szemerédi Regularity Lemma **20**

3.1 Definitions and Notation 20

3.2 Original Statement 22

3.3 Algorithmic Versions of the Regularity Lemma 22

3.3.1 Alon-Duke-Lefmann-Rödl-Yuster Version 22

3.3.2 Frieze-Kannan Version 26

3.4 Using the Regularity Lemma 27

4 Regularity Clustering **28**

4.1 Modifications to the Constructive Version 28

4.2 A Two Phase Strategy for Clustering 30

4.3 Empirical Validation 31

4.3.1 Datasets and Metrics Used 31

4.3.2 Case Study 33

4.3.3 Clustering Results on Benchmark Datasets 39

4.4 Discussion and Future Directions 39

III Clustering for Prediction **44**

5 Clustering for Prediction **45**

5.1 PAC-MDL Bounds and Clustering as Classification 46

5.2 A Simple Bagging Strategy 48

5.2.1 k as a Tunable Parameter 50

5.3 Combining Predictions 51

5.3.1 An Information Theoretic View of Clustering 52

5.3.2 Ensemble Learning 53

5.3.3 Methodology for Combining Predictions 54

5.4 Similarity with Existing Methods 54

5.5 Empirical Validation 55

5.5.1 Algorithms 55

5.5.2 Datasets 56

5.5.3 Methodology 57

5.5.4 Results 59

5.6 Future Work 67

6 Improving Student Test Score Prediction **70**

6.1 Motivation and Literature Survey 70

6.2 Data and Metrics . . . 72
6.3 Methodology . . . 73
6.4 Results using k-means Clustering . . . 74
6.5 Results Using Spectral Clustering . . . 75
6.6 Contributions and Future Work . . . 77

7 Improving Knowledge Tracing **80**
7.1 Motivation and Background on Knowledge Tracing . . . 80
7.1.1 Bayesian Knowledge Tracing . . . 82
7.2 Clustered Knowledge Tracing . . . 83
7.3 Empirical Validation . . . 84
7.3.1 Datasets . . . 84
7.3.2 Results . . . 85
7.4 Discussion . . . 87

8 Co-Clustering for Prediction **89**
8.1 Why Co-Cluster? . . . 90
8.2 Co-Clustering . . . 92
8.2.1 Notation and Definitions . . . 92
8.2.2 Spectral Co-Clustering . . . 94
8.3 Co-Clustering for Bootstrapping . . . 97
8.3.1 Blending Predictions . . . 98
8.4 Experimental Validation . . . 98
8.4.1 Dataset Description and Context . . . 99
8.4.2 Experimental Results . . . 101
8.5 Discussion and Future Work . . . 102

IV Concluding Remarks **105**

9 Conclusions **106**
9.1 Future Work . . . 107

V Bibliography **110**

Bibliography **111**

List of Figures

1.1 Results of using k-means on two synthetic datasets. 9

1.2 Results of using Spectral Clustering on a synthetic dataset. 12

4.1 A Two Phase Strategy for Clustering 31

4.2 Accuracy Landscape for Regularity Clustering on the Red Wine Dataset for different values of ε and refinement size l (with k = 6 on the left and k = 3 on the right). The Plane cutting through in blue represents accuracy by running self-tuned spectral clustering using the fully connected similarity graph. . . 35

4.3 Accuracy Landscape on the Red Wine Dataset (with k = 6 on the left and k = 3 on the right) when the most irregular pair is considered in each refinement. The Plane cutting through in blue represents accuracy by running self-tuned spectral clustering using the fully connected similarity graph. 38

5.1 A "Prediction Model". A "Prediction Model" is composed of k cluster models (PM_k). It should be noted that any other method for regression could be used in place of Linear Regression 49

5.2 Mapping a test point to a cluster to make a prediction on it 50

5.3 Generation of multiple prediction models by using k as a free parameter. Each of these prediction models will make a prediction on the test set. These predictions can then be combined together by a nave ensemble to get a final prediction. 51

5.4 The error profiles for all 11 datasets for stepwise linear regression. The x-axis represents the number of prediction models averaged from 1. The bar marked in red indicates the one that has been chosen by the first heuristic as the final prediction. In many cases we notice that this is clearly a sub-optimal choice. The chosen value and the lowest value in the error profile for each dataset should be contrasted with the value of CVk mentioned in the table. Since the number of prediction models to average chosen is different in each fold by the second method, it has not been represented in the graph. 62

5.5 The error profiles for all 11 datasets for Random Forests (for regression). The x-axis represents the number of prediction models averaged from 1. The bar marked in red indicates the one that has been chosen by the first heuristic as the final prediction. In many cases we notice that this is clearly a sub-optimal choice. The chosen value and the lowest value in the error profile for each dataset should be contrasted with the value of CVk mentioned in the table. Since the number of prediction models to average chosen is different in each fold by the second method, it has not been represented in the graph. . . . 63

6.1 Results on the ASSISTments Data using Spectral Clustering 76

6.2 Visualization of Clustering on the ASSISTments Data using k-means (L) and Spectral Clustering (R). Top row is a 3-D Visualization while the lower rows are planar views of the same. 78

7.1 The Bayesian Knowledge Tracing Model 83

7.2 Results on the Algebra (L) and the Bridge to Algebra (R) datasets with spectral clustering when all the features are considered. The red line shows the ensembled results after averaging from PM_1 to PM_K while the black one shows the results for each Prediction Model (PM_K). 85

7.3 Algebra (L) and the Bridge to Algebra (R) with k-means clust. considering all features. 86

7.4 Algebra (L) and the Bridge to Algebra (R) with k-means clust. considering user features. 87

8.1 Finding a Prediction Model, PM_{kl} with k row clusters and l column clusters 96
8.2 Ordering the Co-Cluster Prediction Models, PM_{kl} 99
8.3 Performance on the 2004-05 Set 101
8.4 Performance on the 2005-06 Set 102

List of Tables

4.1 Clustering Results on Red Wine Dataset by Other Methods 34

4.2 Reduced Graph Sizes. Original Affinity Matrix size : 1599 × 1599 36

4.3 Illustration of Increase in Potential 36

4.4 Regular Partitions with required number of regular pairs and actual number present 37

4.5 Clustering Results on UCI Datasets with No Ordering of Vertices 40

4.6 Compression Obtained on the UCI Datasets 40

5.1 Predictions Using Linear Regression and Clustering 64

5.2 Predictions using Stepwise Linear Regression and Clustering 65

5.3 Predictions using Random Forests and Clustering 65

6.1 Prediction errors by different Prediction Models on the ASSISTments Data with k-means Clustering. 75

6.2 Prediction errors by different Prediction Models Averaged on the ASSISTments Data with k-means Clustering. The subscripts refer to the models whose predictions were used in averaging. 76

8.1 Comparison of predictions based on k-means and Co-Clustering for the ASSISTments 2004-05 Dataset. Figures in bold indicate significance over the baseline on paired t-test. Numbers are Mean Absolute Errors. Also note that Pred. Model 1 corresponds to the baseline 103

8.2 Comparison of predictions based on k-means and Co-Clustering for the ASSISTments 2005-06 Dataset. Figures in bold indicate significance over the baseline on paired t-test. Numbers are Mean Absolute Errors. Also note that Pred. Model 1 corresponds to the baseline 104

Introduction

This thesis makes an attempt to make two humble contributions to the area of clustering in general. Before getting into the specifics of the work we find it appropriate to mention the rise of graph theoretic ideas in machine learning and related disciplines.

In the past decade and a half, two important developments in research in unsupervised learning especially clustering have significantly altered research in the same. One is the increased use of graph theoretic methods for clustering itself leading to improved performance [34], [88], [79], [3], [19], [60], [78], [71] and the second is semi-supervised learning [105] where the main goal is to understand how completely unlabeled data (many times by clustering) can help improve prediction accuracies in supervised settings. Incidentally many of such techniques are also graph-theoretic. There are important reasons for both these developments that we briefly touch upon in this section.

The extensive use of representations based on graphs seems obvious in machine learning and pattern recognition due to their power and flexibility. While such use has been prevalent for a while, there has been renewed interest in the same recently. Unsurprisingly this renewed interest is in sync with the proverbial information explosion and the need to develop better algorithms to understand the massive amounts of data being generated in various settings. The necessity for this need can be summarized in a pithy observation by the noted physicist Freeman J. Dyson of the *Institute for Advanced Study*.

> "The year 2000 was essentially the point at which it became cheaper to collect information than to understand it." [38].

This fresh interest can be understood by the fact that abstracting problems into graph theoretic terms allows them to be crystallized in settings with very strong theoretical un-

derpinnings. These strong foundations allows the use of many graph algorithms making manipulation easier and precise thus facilitating better understanding. The areas of Graphical Models [66] and Deep Learning [13] are examples of this phenomenon within machine learning. There are numerous examples outside (but related to) machine learning too, some striking examples include a significant leap in understanding social and biological networks by using and building upon the arsenal of ideas from the theory of random graphs [7], a rapid evolution of web-search algorithms based on link analysis such as PageRank [24] (and its precursor the HITS algorithm due to Kleinberg [62]) which borrow from spectral graph theory.

One of the two central points of this thesis, Spectral Clustering (the other being the Szemerédi Regularity Lemma) is also a good example of the above in Machine Learning. Spectral clustering is based on ideas from spectral graph theory[22] and has generated a lot of interest in recent years, significantly improving the state of art in clustering in atleast some sense of thw word and also raising new and interesting questions about the very nature of un-supervised and semi-supervised learning.

0.1 Contributions and Overview

Despite various advantages of spectral clustering, one major problem is that for large datasets it is very computationally intensive. In the form of spectral clustering that is originally stated there are two main bottlenecks: First, the finding of the affinity matrix ($O(n^2d)$, where d is the dimensionality of the data) of the pairwise distances between data-points, and second, once we have the affinity matrix (and the Laplacian) the finding of the eigendecomposition, which is $O(n^3)$. Understandably this has received a lot of attention recently. Many ways have been suggested to solve these problems. One approach is not to use an all-connected graph but a k-nearest neighbour graph in which each data-point is typically connected to $\log n$ neighbouring data-points(where n is the number of data-points). This considerably speeds up the process of finding the affinity matrix, however it has a drawback that by taking nearest neighbors we might miss something interesting in the global structure of the data. A method to remedy this is the Nyström method which

takes a random sample (1-3 %) of the entire dataset (thus preserving the global structure in a sense) and then involves doing spectral clustering on this much smaller sample. The results are then extended to all other points in the data set [46], [8].

One contribution of our work is that we propose a method to substantially ease the second bottleneck (i.e once the affinity matrix is given). For doing so, we introduce an approach that is quite different from existing approaches to this problem. This approach can be considered to be a new clustering algorithm that we call *Regularity Clustering*. It harnesses the power of an extremely useful tool from extremal graph theory - The Regularity Lemma of Szémeredi [92] which roughly states that any graph could be partitioned into a bounded number of pseudo-random graphs. We use this decomposition to construct a reduced representation of the original dataset and then work with this to cluster the dataset.

While clustering is an extremely useful tool for exploratory data analysis, it's role in semi-supervised settings has received a lot of interest recently [105],[12]. We too are explicitly interested in investigating how useful clustering of unlabelled data is in boosting prediction accuracy and propose a simple strategy that could make use of any advantage of the same. This interest is motivated by the following:

The statement of the supervised learning problem in machine learning could be roughly stated as: Given a training set consisting of ordered pairs of feature vectors and their associated labels (which might be discrete or continuous), the task of a learning algorithm is to learn a functional map from the feature space to label space. A learning algorithm is said to be more powerful if it is able to learn mappings such that it can generalize well and make correct predictions on test data-points on which it was not trained. Since the functional map under consideration might be highly non-linear, learning algorithms that output only a single mapping (frequently referred to as the hypothesis) might suffer from statistical, computational and representation issues that restrict them from learning good mappings. One way of solving this problem is to transform the feature space into a more suitable and "richer" representation such that learning using this new representation gives much better functional maps as compared to the original representation. This is the motivation behind deep learning methods which have caused a new wave of excitement in the machine learning community since 2006 [13]. Another way of solving this problem atleast partly, is

by using ensemble learning methods [35],[36],[14]. The basic idea behind ensemble methods is that they involve running a "base learning algorithm" multiple times, each time with some change in the representation of the input (e.g. only considering a subset of features in each run) so that a number of diverse predictions (or maps) could be obtained. This diversity in prediction is then exploited to get better predictions. Thus ensemble methods approach the said problem by both trying to learn multiple functional maps and also by learning a more distributed and hence "richer" representation of the input space at the same time. In this work we also propose to use clustering for the task of bootstrapping and some work has shown that this strategy in spite of it's simplicity is quite powerful. This idea is quite unlike other bagging methods which use a random subset to bootstrap. Thus, it has the potential advantage that the subsets used to bootstrap could be more interpretable and actionable.

To talk about these contributions, the document is organized as follows: Part One I is on preliminaries and has one chapter (Chapter 1) that reviews some basics on Spectral Clustering. Part Two II talks about the Regularity Clustering method and is divided into three chapters: Chapter 2 covers the motivation to Regularity Clustering while Chapter 3 discusses the Regularity Lemma of Szemerédi, both the original statement and constructive versions by Alon *et al.* and Frieze and Kannan. Chapter 4 discusses the actual algorithm that we introduce and discusses modifications to a constructive version of the Regularity Lemma. Part III talks of Clustering for Prediction and is divided into four chapters. Out of the four, Chapter 5 talks of a bagging strategy for using clustering for prediction and reports results on a number of benchmark datasets for the same. The other chapters in this part are applications of this strategy with the last one being an extension to Co-Clustering. Part IV reviews some contributions and talks about some future directions of work.

Part I

Preliminaries

Chapter 1

Preliminaries

The central goal of this chapter is to review the basic notion of clustering, the back bone of this thesis. In particular, we review a popular spectral clustering algorithm and a variant that makes it more useful in practice.

1.1 Clustering

The way humans interact with the world and form intuitions about it involves the idea of clustering in a big way. For instance, how we view different people and objects and place them in rough groups involves some kind of clustering. Indeed, it has been remarked that a mathematical precise notion of clustering of sensory data could help approximate solving problems as done by the brains in some sense [91]. The goal of clustering - that is to group data in a meaningful and coherent way such that objects that are more similar to each other are placed in the same cluster, is perhaps one of the most compelling and intuitive. Yet, clustering is one of the most ill defined machine learning tasks considering how widely used it is. This ill defined nature coupled with its wide applicability make it an area of fruitful investigation both in terms of new ideas and algorithms and work in clustering theory. Clustering is now perhaps the most widely used tool for exploratory data analysis and knowledge discovery by means of cluster analysis. Applications involve areas such as general Artificial Intelligence, Business Intelligence, Customer Analytics, Recommender Systems, Medicine, Bioinformatics and Genomics etc. Clustering is critical to data mining

because of its ability to summarize the data into a more manageable form, performing a kind of lossy compression on the data.

There are various approaches to clustering, each of which have their unique characteristics. Clustering algorithms can be top-down or bottom-up. In top-down approaches, the entire dataset is considered to be one cluster and then the other clusters are formed by recursive partitioning. On the other hand, the bottom-up approaches involve considering each data point to be one cluster and then merging them successively to get fewer clusters. On the basis of cluster membership, clustering could be classified as hard or fuzzy. Hard clustering induces a flat partitioning into non-overlapping groups while fuzzy clustering gives a probability for cluster membership for each point. Out of the various modern clustering algorithms, Spectral Clustering has become one of the most popular. In the next section we review some of the background and basics on Spectral Clustering.

1.2 Spectral Clustering

The spectral clustering "gold-rush" in the Machine Learning community started with the works of Shi and Malik [88], Meila and Shi[73], Ng, Jordan and Weiss [78] around 2000-01, eventhough work in Spectral Clustering is much older. It was first suggested by Donath [37] that graph partitions based on the eigenvectors of the adjacency matrix were feasible. In the same year Fiedler [43] suggested using the second eigenvalue for a bipartition of a graph. Since then, there has been a steady stream of work that used spectral clustering in different guises before it was picked up by the machine learning community at large.

Spectral Clustering is a graph theoretic technique for metric modification such that it gives a much more global notion of similarity between data points as compared to other clustering methods such as k-means. It thus represents data in such a way that it is easier to find meaningful clusters on this new representation. It is especially useful in complex datasets where traditional clustering methods would fail to find groupings.

To understand the weakness of methods such as k-means, a useful way of looking at clustering is the following: Consider a set of K distributions, $\mathcal{D} = \{D_1, D_2 \dots D_K\}$ such that each of these distributions has an associated weight, the collection of which is given by

$\{w_1, w_2 \dots w_K\}$ such that $\sum_i w_i = 1$. Suppose a dataset is generated by sampling these K distributions, such that a point in this dataset might be picked from distribution D_i with probability w_i. Under the partitional paradigm, the objective of clustering methods is to identify these K distributions given a dataset. Methods such as k-means and Expectation Maximization (EM) are based on estimating explicit models of the data. While k-means finds the clusters by assuming that the set of distributions $\mathcal{D}$ that generated the data was a set of spherical Gaussians, EM algorithms in general learn a mixture of Gaussians with arbitrary shapes. More formally, k-means finds the clusters by minimizing the distortion function:

$$J(c, \mu) = \sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

Where μ_c is the cluster centroid to which a point x has been assigned. In spite of the great popularity of the k-means algorithm very few theoretical guarantees on its performance are known. In practice however, k-means performs well on data that at least approximately follows its assumption of being generated by a mixture of well-separated spherical Gaussians [20]. This, coupled with its simplicity makes it a handy tool for a data-miner. However, k-means performs poorly when these assumptions of data generation are not met, which is usually the case in real world datasets. Figure 1.1 shows the results of using k-means on two synthetic datasets. k-means is unable to identify clusters when the data is distributed in concentric groups (left figure in Figure 1.1), while it clearly finds the clusters in well separated and tight spherical Gaussians (right). The clusters identified are indicated by different colors. Both sets have 600 points. Spectral Clustering makes no such assumptions for data generation. It instead finds groupings by analyzing the top eigenvectors of the affinity matrix and hence usually returns better results.

Note that the broad idea of clustering is essentially to group points that are similar in one cluster and points that are dissimilar into different clusters. The notion of similarity that is employed in k-means is the Euclidean distance between data points and the cluster centroids to which they are assigned to (which get updated in each iteration). In a sense, the idea of similarity used in k-means restricts what could be known about the geometry of the data. In k-means we work with the data directly, in spectral clustering however, we work with a representation of the data that gives a more global (and hence better) encoding

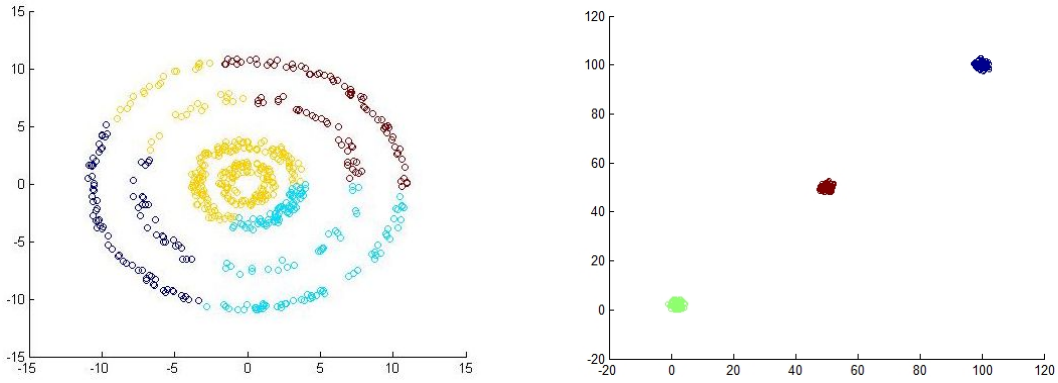

Figure 1.1: Results of using k-means on two synthetic datasets.

of the similarities between points. This similarity in spectral clustering is represented in the form of a graph called the similarity graph, represented by $\mathcal{G} = (V, E)$ where V is the set of vertices and E is the set of edges. The idea is that points in the dataset can be represented by a graph with each data point as a vertex of the graph $\mathcal{G}$ and the edges connecting them encoding a notion of similarity $w_{ij} > 0$ between them. Two points are connected in the graph if the similarity or weight between them is either non-zero or above some threshold. The clustering problem can then be re-stated using information from the similarity graph as: We want to find partitions of this graph such that weights between points in the same group are high and those between points in different groups are low. Before talking how we cluster using this representation, we introduce some notation and discuss how the graph is used to represent the dataset.

Given the similarity graph $\mathcal{G}$ of n data points $\{x_1, x_2 \dots x_n\}$, there are essentially two things about it that tell us something about the global structure of the data:

1. The degree of a vertex (a data-point in our case): The degree of a vertex tells us the sum of weights of all the edges that originate from a vertex i to all other vertices j. It is given by:

$$d_i = \sum_{j=1}^{n} w_{ij}$$

This definition is somewhat non-standard but more general. The standard definition for degree of a vertex is only defined for $w_{ij} = \{0, 1\}$, and thus is only the count of vertices a given vertices is connected to. Given this definition, the degree matrix of

the similarity graph is the diagonal matrix $\mathcal{D}$ with the degrees d_i on the diagonal.

$$D = \begin{pmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{pmatrix}$$

Intuitively the degree matrix of a graph tells us how many points each point is connected to (we could connect all points, or choose to connect k-nearest neighbors of each point) and by how much (hence the summation of the weights).

2. The weighted similarity matrix or the affinity matrix of the similarity graph, W on the other hand is a representation of similarity between all the points. Each element in the affinity matrix is given by $w_{i.j}$, which is the weight or edge between two points i and j. A common way of representing the weight is:

$$w_{ij} = exp\left(\frac{-\left\|x_i - x_j\right\|^2}{2\sigma^2}\right)$$

Notice that $w_{i,j}$ is simply the exponentiated Euclidean distance between two points (points in $\mathcal{R}^n$) scaled by a parameter called the scaling or weighing parameter σ . This parameter is to be tuned and varying it changes the weight between points. A point to note is that if all the points are connected then all $w_{i,j}$ such that $i \neq j$ will be non-zero values. If points are connected to only their k-nearest neighbors and not every other point, then most of the matrix W will be populated with zeroes.

The matrices W and D tell us something about the global structure of the data, but we dont work with them directly. We instead work with the graph Laplacian matrix given by

$$L = D - W$$

The above is the un-normalized version of the Laplacian. There are two normalized versions that are represented as:

$$L_{sym} = D^{-1/2} W D^{-1/2}$$

and

$$L_{rw} = D^{-1} W$$

The first is called the symmetric Laplacian while the second is called the random-walk Laplacian. The Laplacian in a way combines both the degree and the affinity matrix and also has some mathematically interesting properties (such as being positive semi-definite) that make it easier to work with [75]. Since the Laplacian is a representation of the similarity between the data-points, we can now work with it to find groups in the data. Given the above background, clusters in a dataset can be found by the following method[78]:

A Spectral Clustering Algorithm due to Ng, Jordan, Weiss [78]:

1. *For the dataset having n data points, construct the similarity graph $\mathcal{G}$. The similarity graph can be constructed in two ways: by connecting each data point to the other $n-1$ data points or by connecting each data point to its k-nearest neighbors. A rough estimate of a good value of the number of nearest neighbors is $log(n)$. The similarity between the points is given by $w_{ij} = exp\left(\frac{-\left\|x_i - x_j\right\|^2}{2\sigma^2}\right)$. This will give the matrix W.*

2. *Given the similarity graph, construct the degree matrix D.*

3. *Using D and W find L_{sym}.*

4. *Let K be the number of clusters to be found. Compute the first K eigenvectors of L_{sym}. Sort the eigenvectors according to their eigenvalues.*

5. *If $u_1, u_2 ... u_K$ are the top eigenvectors of L_{sym}, then construct a matrix U such that $U = \{u_1, u_2 \dots u_K\}$. Normalize rows of matrix U to be of unit length.*

6. *Treat the rows in the normalized matrix U as points in a K dimensional space and use k-means to cluster these.*

7. *If $c_1, c_2 \dots c_K$ are the K clusters, Then assign a point in the original dataset s_i to cluster c_K if and only if the i^{th} row of the normalized U is assigned to cluster c_K.*

It is noteworthy that we dont cluster the original dataset directly. We first transform it to find its top K eigenvectors. These being the most important eigenvectors of L, encode the maximum information about it. At the same time, this reduces the dimensionality which without throwing away much information which makes the task of clustering much

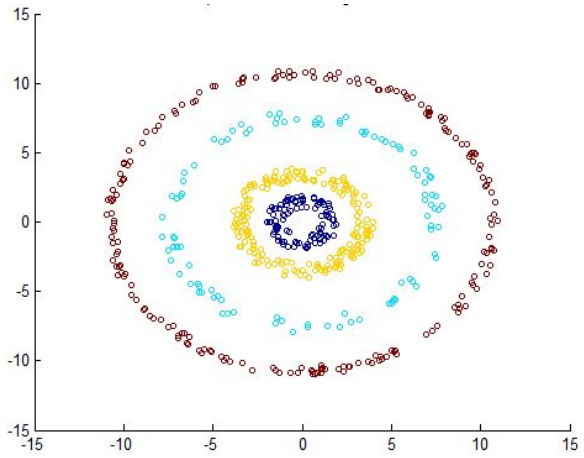

Figure 1.2: Results of using Spectral Clustering on a synthetic dataset.

easier. To illustrate the power given by this change of representation, we demonstrate it on a toy dataset (Figure 1.2). A detailed tutorial that explains various spectral clustering algorithms and some point of views on why it works is by Luxburg[71].

1.2.1 Self-Tuning Spectral Clustering

One major issue with the Spectral Clustering algorithms suggested before 2005 was that they were usually not suitable for handling multi-scale data. The associated parameter that handled scaling had to be learned by cross-validation. This parameter is the parameter σ in the equation below:

$$w_{ij} = exp\left(\frac{-\left\|x_i - x_j\right\|^2}{2\sigma^2}\right)$$

Many times, for cluster analysis true cluster labels are not known which restricted the utility of spectral clustering algorithms. An important development was the suggestion of using Local Scaling for the selection of an appropriate σ by Zelnik-Manor and Perona [103]. Choosing σ by cross-validation had many issues - Data which had clusters at various scales would usually have a different σ associated and hence using the same for the entire dataset would result in bad results overall. At the same time, since σ had to be learned by cross-validation, other than the increased time and computational resources needed for the same, the initial guess for searching it had to be done manually. [103] suggested using local scaling in the following way to over come this problem: Instead of selecting a single

scaling parameter for the entire dataset they proposed calculating a scaling parameter σ_i for each data point x_i What this would imply that for the a distance between two points could be represented asymmetrically. The distance to point x_j from point x_i would be given as $\frac{d(x_i,x_j)}{\sigma_i}$ and the distance to point x_i from x_j would be given by $\frac{d(x_j,x_i)}{\sigma_j}$. The squared distance could then be appropriately written as:

$$\frac{d^2(x_i, x_j)}{\sigma_i \sigma_j}$$

. Thus the affinity between two points could be written as

$$w_{ij} = exp\left(\frac{-d^2(x_i,x_j)}{\sigma_i \sigma_j}\right)$$

The good thing about this formulation is that each σ could be calculated based on the local statistics of each point, completely eliminating the need of finding the same by cross validation. Thus, the algorithm due to Ng., Jordan, and Weiss could be modified with this change in calculating the affinity matrix. Experimental results using Self-Tuning generally returns better results.

Part II

Regularity Clustering

Chapter 2

Motivation to Regularity Clustering

2.1 Introduction

The Regularity lemma of Szemerédi [92] has proved to be a very useful tool in graph theory. It was initially developed as an auxiliary lemma to prove a long standing conjecture of Erdős and Turán on arithmetic progressions [39] , which stated that sequences of integers with positive upper density must contain arbitrarily long arithmetic progressions. Now the Regularity Lemma by itself has become an important tool and found numerous other applications (see [69]). Based on the Regularity Lemma and the Blow-up Lemma [67], [68] the Regularity method has been developed that has been quite successful in a number of applications in graph theory (e.g. [56], [57]).

However, one major disadvantage of these applications and the Regularity Lemma is that they are mainly theoretical, they work only for astronomically large graphs as the Regularity Lemma can be applied only for such large graphs. Indeed, to find the ε-regular partition in the Regularity Lemma, the number of vertices must be a tower of 2's with height proportional to ε^{-5}. Furthermore, Gowers demonstrated [52] that a tower bound is necessary.

The basic content of the Regularity Lemma could be described by saying that every graph can, in some sense, be partitioned into random graphs. Since random graphs of

a given edge density are much easier to treat than all graphs of the same edge-density, the Regularity Lemma helps us to carry over results that are trivial for random graphs to the class of all graphs with a given number of edges. We are especially interested in harnessing the power of the Regularity Lemma for clustering data. Graph partitioning methods for clustering and segmentation have become quite popular in the past decade because of the ease of representations and interactions with graphs and the theoretical underpinnings for clustering provided by spectral graph theory. The Regularity Lemma is a result that comes with important theoretical implications and thus importing it into machine learning for clustering could potentially have many advantages. An earlier attempt to apply the Regularity Lemma in real-world applications can be found in [90], but the authors do not provide too many details.

In this part we propose a general methodology to make the Regularity Lemma more useful in practice. To make it truly applicable, instead of constructing a provably regular partition we construct an *approximately* regular partition. This partition behaves just like a regular partition (especially for graphs appearing in practice) and yet it does not require the large number of vertices as mandated by the original Regularity Lemma. We will call this Practical Regularity Lemma. Then this approximately regular partition is used for performing clustering and image segmentation. We call the resulting new clustering technique *Regularity clustering*. We present comparisons with standard pairwise clustering methods such as k-means and spectral clustering and the results are very encouraging

To present our results, this part is organized as follows. In Section 2.2 we discuss clustering in general and also present a popular spectral clustering algorithm that is used later in the reduced graph. We also point out what are the possible ways to improve its running time. In Section 3.1 we give some definitions and general notation. In Chapter 3 we present the Regularity Lemma, first the original form (which was non-constructive) and two well known constructive versions that were proved much later. Furthermore, in this section we point out the various problems arising when we attempt to apply the lemma in real-world applications and a general outline of how the Regularity Lemma is used in establishing theoretical results. In Section 4.1 we discuss how the constructive Regularity Lemma could be modified to make it truly applicable for real-world problems where the graphs typically

are much smaller, say have a few thousand vertices only. In Section 4.2 we show how this Practical Regularity Lemma can be applied to develop a new clustering technique called Regularity clustering. In Section 4.3, we present an extensive empirical validation of our method and also present comparisons with methods such as spectral clustering. Section 4.4 is spent in discussing the various possible future directions of work.

2.2 Clustering

It is fair to say that at least some part of our understanding of the world is by a semi-supervised learning process that involves some sort of clustering. Not surprisingly it has been often pointed out that a mathematically precise understanding of clustering could go a long way in helping us solve problems at least approximately as solved by the brain [91]. Clustering is perhaps the most used data mining tool for exploratory data analysis. It's uses are primarily in finding meaningful groups, for feature extraction and summarizing and using it for making data-driven inferences. An useful view of clustering is the following: Given a space X, clustering could be thought of as a partitioning of this space into K parts i.e. $f : X \longmapsto \{1, \ldots, K\}$. Usually this partitioning is obtained by optimizing some internal criteria such as the inter-cluster distances, etc.

Out of the various modern clustering techniques, spectral clustering has become one of the most popular. This has happened due to not only its superior performance over the traditional clustering techniques, but also due to the strong theoretical underpinnings in spectral graph theory and its ease of implementation. It has many advantages over the more traditional clustering methods such as k-means and expectation maximization (EM). The most important is its ability to handle datasets that have arbitrary shaped clusters. Methods such as k-means and EM are based on estimating explicit models of the data. Such methods fail spectacularly when the data is organized in very irregular and complex clusters. Spectral clustering on the other hand does not work by estimating explicit models of the data. It analyzes the spectrum of the graph Laplacian obtained from the pairwise similarities of the data points (similarity matrix). This is useful as the top few eigenvectors can unfold the data manifold to form meaningful clusters [105].

In this work we employ spectral clustering on the reduced graph, even though any other pairwise clustering method could be used. The algorithm that we employ is due to Ng, Jordan and Weiss [78]. Despite various advantages of spectral clustering, one major problem is that for large datasets it is very computationally intensive. And understandably this has received a lot of attention recently. In the form of spectral clustering that is originally stated there are two main bottlenecks: First, the finding of the affinity matrix of the pairwise distances between datapoints, and second, once we have the affinity matrix the finding of the eigendecomposition. Many ways have been suggested to solve these problems. One approach is not to use an all-connected graph but a k-nearest neighbour graph in which each data point is typically connected to $\log n$ neighboring datapoints(where n is the number of data-points). This considerably speeds up the process of finding the affinity matrix, however it has a drawback that by taking nearest neighbors we might miss something interesting in the global structure of the data. A method to remedy this is the Nyström method which takes a random sample of the entire dataset (thus preserving the global structure in a sense) and then doing spectral clustering on this much smaller sample. The results are then extended to all other points in the data set [46], [8].

Our work is quite different from such methods. The speed-up is primarily in the second stage where eigendecomposition is to be done. The original graph is represented by a reduced graph which is much smaller and hence the resulting affinity matrix is much smaller reducing the computational load. Further work on a practical variant of the sparse Regularity Lemma could be useful in a speed-up in the first stage, too.

2.3 Prior Attempts to Use the Regularity Lemma for Practical Applications

Given the astronomical bounds associated with the Regularity Lemma and moreover since Gowers established that they are necessary [52], the practical applications of the Regularity Lemma have been non-existent. Indeed, many times some indirect applications are alluded to. A recent exposition on the work of Endre Szemerédi by Gowers [53] gives an instance of an indirect practical application that could be particularly useful. This is in relation to

the Probably-Approximately-Correct or simply PAC formalism advanced by Leslie Valiant [100], a very useful way of thinking about machine learning. However, no results concerning the Regularity Lemma and it's application to the PAC Learning formalism are known.

Another recent line of investigation that might eventually be relevant to Machine Learning is due to Csiszár-Rejtö-Tusnády [27]. In this work they investigate the utility of the Regularity Lemma with respect to statistical inference on random structures and also give an interpretation of the Regularity Lemma for the statistician.

In more practical applications a recent work by Nepusz *et al.* [76] involves prediction of yet uncharted connections in the large scale network of the cerebral cortex. They give an interesting way to solve this problem by a method that claims inspiration from the Regularity Lemma. In another paper by Nepusz and Baszó, the Regularity Lemma is again used as an inspiration as in [76] for developing an algorithm for clustering of Directed Graphs [77]. Pehkonen and Reittu [82] also use the Regular Partition promised by the Regularity Lemma as an inspiration to develop a strategy to cluster peer-to-peer streaming data. However, note that none of the above applications use the Regularity Lemma directly, they only use the Regularity Lemma as an inspiration to come up with a similar working methodology.

The only practical application of the Regularity Lemma to the best of our understanding remains by Sperotto & Pelillo [90], where they use the Regularity Lemma as a pre-processing step. They give some interesting ideas on how the Regularity Lemma might be used, however they don't give too many details on the same. In subsequent chapters we give an attempt to harness the power of the Regularity Lemma for the practical task of clustering. We begin with some background on the Regularity Lemma in the next chapter.

Chapter 3

The Szemerédi Regularity Lemma

The Regularity Lemma is a cornerstone result in Extremal Graph theory with wide ranging implications and applications. It's fundamental nature can be understood by considering that it has an important place in linking many area of mathematics, such as analysis and combinatorics. The Regularity Lemma was originally advanced as a tool in the original proof in 1975 by Endre Szemerédi of an old conjecture of Erdős and Turán on Ramsey Properties of arithmetic progressions [39].

The purpose of this chapter is to introduce the Regularity Lemma. It must be noted that the statement of the Lemma as originally stated is an existential, non-constructive one. If we have to work towards making the Regularity Lemma truly applicable to settings in clustering, then we first need an algorithmic version and then make changes to that. Here we review the original statement of the Regularity Lemma and it's constructive versions. However, before doing so, we introduce some definitions and notations.

3.1 Definitions and Notation

Let $G = (V, E)$ denote a graph, where V is the set of vertices and E is the set of edges. When A, B are disjoint subsets of V, the number of edges with one endpoint in A and the other in B is denoted by $e(A, B)$. When A and B are nonempty, we define the *density* of edges between A and B as

$$d(A, B) = \frac{e(A, B)}{|A||B|}.$$

The most important concept is the following.

Definition 1. *The bipartite graph $G = (A, B, E)$ is ε-regular if for every $X \subset A$, $Y \subset B$ satisfying*

$$|X| > \varepsilon|A|,\ |Y| > \varepsilon|B|$$

we have

$$|d(X, Y) - d(A, B)| < \varepsilon,$$

otherwise it is ε-irregular.

Roughly speaking this means that in an ε-regular bipartite graph the edge density between *any* two relatively large subsets is about the same as the original edge density. In effect this implies that all the edges are distributed almost uniformly.

Definition 2. *A partition P of the vertex set $V = V_0 \cup V_1 \cup \ldots \cup V_k$ of a graph $G = (V, E)$ is called an equitable partition if all the classes $V_i, 1 \leq i \leq k$, have the same cardinality. V_0 is called the exceptional class.*

Thus note that the exceptional class V_0 is there only for a technical reason, namely to guarantee that the other classes have the same cardinality.

Definition 3. *For an equitable partition P of the vertex set $V = V_0 \cup V_1 \cup \ldots \cup V_k$ of $G = (V, E)$, we associate a measure called the index of P (or the potential) which is defined by*

$$ind(P) = \frac{1}{k^2} \sum_{s=1}^{k} \sum_{t=s+1}^{k} d(C_s, C_t)^2.$$

This will measure the progress towards an ε-regular partition.

Definition 4. *An equitable partition P of the vertex set $V = V_0 \cup V_1 \cup \ldots \cup V_k$ of $G = (V, E)$ is called ε-regular if $|V_0| < \varepsilon|V|$ and all but εk^2 of the pairs (V_i, V_j) are ε-regular where $1 \leq i < j \leq k$.*

With these definitions we are now in a position to state the Regularity Lemma.

3.2 Original Statement

The Regularity Lemma essentially states that every large enough graph could be approximated by the union of a constant number of bipartite graph. A detailed exposition to some statements and applications of the Regularity Lemma can be found here [69].

More specifically this lemma claims that every (dense) graph could be partitioned into a bounded number of pseudo-random bipartite graphs and a few leftover edges. Since random graphs of a given edge density are much easier to treat than all graphs of the same edge-density, the Regularity Lemma helps us to translate results that are trivial for random graphs to the class of all graphs with a given number of edges.

Theorem 1 (Szemerédi Regularity Lemma [92]). *For every positive $\varepsilon > 0$ and positive integer t there is an integer $T = T(\varepsilon, t)$ such that every graph with $n > T$ vertices has an ε-regular partition into $k + 1$ classes, where $t \leq k \leq T$.*

3.3 Algorithmic Versions of the Regularity Lemma

The original proof of the regularity lemma [92] does not give a method to construct a regular partition but only shows that one must exist. To apply the regularity lemma in practical settings, we need a constructive version. Alon *et al.* [2] were the first to give an algorithmic version. Since then a few other algorithmic versions have also been proposed [48], [64]. Below we present the details of the Alon *et al.* algorithm.

3.3.1 Alon-Duke-Lefmann-Rödl-Yuster Version

Theorem 2 (Alon *et al.* Algorithmic Regularity Lemma [2]). *For every $\varepsilon > 0$ and every positive integer t there is an integer $T = T(\varepsilon, t)$ such that every graph with $n > T$ vertices has an ε-regular partition into $k+1$ classes, where $t \leq k \leq T$. For every fixed $\varepsilon > 0$ and $t \geq 1$ such a partition can be found in $O(M(n))$ sequential time, where $M(n)$ is the time for multiplying two n by n matrices with $0, 1$ entries over the integers. The algorithm can be parallelized and implemented in NC^1.*

This result is somewhat surprising from a computational complexity point of view since as it was proved in [2] the corresponding decision problem (checking whether a given partition is ε-regular) is $co-NP$-complete. Thus the search problem is easier than the decision problem.

Theorem 3 *[2] The Following decision problem is $co-NP$-complete: Given a Graph G, an integer $k \geq 1$ and a parameter $\varepsilon > 0$, and a partition of the set of vertices of G in $k+1$ parts. Decide is the given partition is ε-regular.*

To describe the algorithm underlined in Theorem 2 and how it is an algorithm in the first place, we first introduce a couple of definitions:

Definition 5 (Average Degree).

If H is a bipartite graph with classes of equal size $|A| = |B| = n$. Then, the average degree of H may be given as:

$$d = \frac{1}{2n} \sum_{i \in A \cup B} deg(i)$$

Definition 6 (Neighborhood Deviation) *[2]Let y_1 and y_2 be two distinct vertices, such that $y_1, y_2 \in B$. The neighborhood deviation of y_1, y_2 is defined as:*

$$\sigma(y_1, y_2) = |N(y_1) \cap N(y_2)| - \frac{d}{n^2}$$

Where N(.) gives the neighborhood of a vertex. For some subset $Y \subseteq B$, the deviation of this set *Y is defined as:*

$$\sigma(Y) = \frac{\sum_{y_1, y_2 \in Y} \sigma(y_1, y_2)}{|Y|^2}$$

Now given that $0 < \varepsilon < \frac{1}{16}$ and if $Y \subseteq B$ and is sufficiently large i.e $|Y| > \varepsilon n$ and such that $\sigma(Y) \geq \frac{\varepsilon^3 n}{2}$, then one the following conditions holds:

1. $d < \varepsilon^3 n$ i.e. H is ε-irregular
2. There exists in B a set of atleast $\frac{1}{8}\varepsilon^4 n$. The degrees of this set deviate from d by at least $\varepsilon^4 n$
3. There are subsets $A' \subseteq A, B' \subseteq B, |A'| \geq \frac{\varepsilon^4}{4} n, |B'| \geq \frac{\varepsilon^4}{4} n$ and $|d(A', B') - d(A, B)| \geq \varepsilon^4$.

It is using these cases that the algorithm mentioned in Theorem 2 can be derived. To describe the actual algorithm we need a couple of more Lemmas.

Lemma 1 (Alon *et al.* [2]). *Let H be a bipartite graph with equally sized classes $|A| = |B| = n$. Let $2n^{-1/4} < \varepsilon < \frac{1}{16}$. There is an $O(M(n))$ algorithm that verifies that H is ε-regular or finds two subset $A' \subset A$, $B' \subset B$, $|A'| \geq \frac{\varepsilon^4}{16}n$, $|B'| \geq \frac{\varepsilon^4}{16}n$, such that $|d(A,B) - d(A',B')| \geq \varepsilon^4$. The algorithm can be parallelized and implemented in NC^1.*

This lemma basically says that we can either verify that the pair is ε-regular or we provide certificates that it is not. The certificates are the subsets A', B' and they help to proceed to the next step in the algorithm. The next lemma describes the procedure to do the refinement from these certificates.

Lemma 2 (Szemerédi [92]). *Let $G = (V, E)$ be a graph with n vertices. Let P be an equitable partition of the vertex set $V = V_0 \cup V_1 \cup \ldots \cup V_k$. Let $\gamma > 0$ and let k be a positive integer such that $4^k > 600\gamma^{-5}$. If more than γk^2 pairs (V_s, V_t), $1 \leq s < t \leq k$, are γ-irregular then there is an equitable partition Q of V into $1 + k4^k$ classes, with the cardinality of the exceptional class being at most*

$$|V_0| + \frac{n}{4^k}$$

and such that

$$ind(Q) > ind(P) + \frac{\gamma^5}{20}.$$

This lemma implies that whenever we have a partition that is not γ-regular, we can refine it into a new partition which has a better index (or potential) than the previous partition. The refinement procedure to do this is described below.

Refinement Algorithm: *Given a γ-irregular equitable partition P of the vertex set $V = V_0 \cup V_1 \cup \ldots \cup V_k$ with $\gamma = \frac{\varepsilon^4}{16}$, construct a new partition Q.*
For each pair (V_s, V_t), $1 \leq s < t \leq k$, we apply Lemma 1 with $A = V_s$, $B = V_t$ and ε. If (V_s, V_t) is found to be ε-regular we do nothing. Otherwise, the certificates partition V_s and V_t into two parts (namely the certificate and the complement). For a fixed s we do this for all $t \neq s$. In V_s, these sets define the obvious equivalence relation with at most 2^{k-1} classes,

namely two elements are equivalent if they lie in the same partition set for every $t \neq s$. The equivalence classes will be called atoms. Set $m = \lfloor \frac{|V_i|}{4^k} \rfloor$, $1 \leq i \leq k$. Then we choose a collection Q of pairwise disjoint subsets of V such that every member of Q has cardinality m and every atom A contains exactly $\lfloor \frac{|A|}{m} \rfloor$ members of Q. The collection Q is an equitable partition of V into at most $1 + k4^k$ classes and the cardinality of its exceptional class is at most $|V_0| + \frac{n}{4^k}$.

Now we are ready to present the main algorithm.

Regular Partition Algorithm: *Given a graph G and ε, construct a ε-regular partition.*

1. **Initial partition:** *Arbitrarily divide the vertices of G into an equitable partition P_1 with classes $V_0, V_1, \ldots, V_b$, where $|V_1| = \lfloor \frac{n}{b} \rfloor$ and hence $|V_0| < b$. Denote $k_1 = b$.*

2. **Check regularity:** *For every pair (V_s, V_t) of P_i, verify if it is ε-regular or find $X \subset V_s, Y \subset V_t, |X| \geq \frac{\varepsilon^4}{16}|V_s|, |Y| \geq \frac{\varepsilon^4}{16}|V_t|$, such that $|d(X,Y) - d(V_s,V_t)| \geq \varepsilon^4$.*

3. **Count regular pairs:** *If there are at most εk_i^2 pairs that are not verified as ε-regular, then halt. P_i is an ε-regular partition.*

4. **Refinement:** *Otherwise apply the Refinement Algorithm and Lemma 2, where $P = P_i, k = k_i, \gamma = \frac{\varepsilon^4}{16}$, and obtain a partition Q with $1 + k_i 4_i^k$ classes.*

5. **Iteration:** *Let $k_{i+1} = k_i 4_i^k, P_{i+1} = Q, i = i + 1$, and go to step 2.*

Since the index cannot exceed 1/2, the algorithm must halt after at most $\lceil 10\gamma^{-5} \rceil$ iterations (see [2]). Unfortunately, in each iteration the number of classes increases to $k4^k$ from k. This implies that the graph G must be indeed astronomically large (a tower function) to ensure the completion of this procedure. As mentioned before, Gowers [52] proved that indeed this tower function is necessary in order to guarantee an ε-regular partition for *all* graphs. The size requirement of the algorithm above makes it impractical for real world situations where the number of vertices typically is a few thousand.

In the next section we describe the constructive version of the Regularity Lemma due to Frieze and Kannan [48].

3.3.2 Frieze-Kannan Version

To describe this algorithm we need Lemma 2, the definitions in Section 3.1 and another lemma that we state below.

Lemma 3 (Frieze-Kannan [48]) *Let W be an $R \times C$ matrix with $|R| = p$ and $|C| = q$ and $\left\|W\right\|_\infty$ and let γ be a positive real.*

a If there exists $S \subseteq R, T \subseteq C$ such that $|S| \geq \gamma p, |T| \geq \gamma q$ and $|W(S,T)| \geq \gamma |S||T|$ then $\sigma_1(W) \geq \gamma^3 \sqrt{pq}$. Where σ_1 is the first singular value.

b If $\sigma_1(W) \geq \gamma\sqrt{pq}$ then there exist $S \subseteq R, T \subseteq C$ such that $|S| \geq \gamma' p, |T| \geq \gamma' q$ and $W(S,T) \geq \gamma'|S||T|$, where $\gamma' = \frac{\gamma^3}{108}$. Also S, T can be constructed in Polynomial time by the method of conditional expectation [83], [89]

Combining the Lemmas 2 and 3, we get an algorithm for finding an epsilon regular partition, quite similar to the Alon *et al.* version [2], which we present below:

Regular Partition Algorithm (Frieze-Kannan): *Given a graph G and ε, construct a ε-regular partition.*

1. **Initial partition:** *Arbitrarily divide the vertices of G into an equitable partition P_1 with classes $V_0, V_1, \ldots, V_b$, where $|V_1| = \lfloor \frac{n}{b} \rfloor$ and hence $|V_0| < b$. Denote $k_1 = b$.*

2. **Check regularity:** *For every pair (V_s, V_t) of P_i, compute $\sigma_1(W_{r,s})$. If the pair (V_r, V_s) are not ε-regular then by Lemma 3 we obtain a proof that they are not not $\gamma = \varepsilon^9/108$-regular.*

3. **Count regular pairs:** *If there are at most εk_i^2 pairs that produce proofs of non γ-regularity, then halt. P_i is an ε-regular partition.*

4. **Refinement:** *Otherwise apply the Refinement Algorithm and Lemma 2, where $P = P_i, k = k_i, \gamma = \frac{\varepsilon^9}{108}$, and obtain a partition P' with $1 + k_i 4_i^k$ classes.*

5. **Iteration:** *Let $k_{i+1} = k_i 4_i^k, P_{i+1} = P', i = i + 1$, and go to step 2.*

This algorithm is guaranteed to finish in at most ε^{-45} steps with an ε-regular partition.

In the next section we outline some basic ideas that are used to apply the Regularity Lemma.

3.4 Using the Regularity Lemma

In applications of the Regularity Lemma the concept of the *reduced graph* plays an important role.

Definition 7 *Given an ε-regular partition of a graph $G = (V, E)$ as provided by Theorem 1, we define the reduced graph G^R as follows. The vertices of G^R are associated to the classes in the partition and the edges are associated to the ε-regular pairs between classes with density above d.*

The most important property of the reduced graph is that many properties of G are inherited by G^R. Thus G^R can be treated as a representation of the original graph G albeit with a much smaller size, an "essence" of G. Then if we run any algorithm on G^R instead of G we get a significant speed-up.

An instance of using the Regularity Lemma via the concept of a reduced graph is the so called Regularity Lemma-Blow-Up Lemma Method developed by Komlós-Sárközy-Szemerédi [67]. The rough outline of the method involves first applying the Regularity Lemma to the graph and obtaining the reduced graph. Given that it has properties as outlined above, knowing the density conditions in the reduced graph we can find nice objects (these depend on the particular application and density condition, triangles, cliques, etc.) in it by applying usually well-known or easy extremal graph theoretical results. The goal is to reduce the embedding problem in the original graph to embedding into these regular objects in the partitioning. For this purpose sometimes we have to do some technical steps, such as connecting these objects (if for example we are looking for a Hamiltonian cycle) or eliminating the various exceptional vertices, etc. For the last step of the method, for the embedding into the regular objects, the Blow-up Lemma [67] is used.

However, the use of the reduced graph is mainly in establishing theoretical results. In the next chapter we suggest some changes to the Regular Partition algorithm described in the previous section so that it could be made more applicable to smaller graphs.

Chapter 4

Regularity Clustering

4.1 Modifications to the Constructive Version

The work in this section is also explicated in this paper[93].

To make the regularity lemma applicable we first needed a constructive version that we stated above. But we see that even the constructive version is not directly applicable to real world scenarios. We note that the above algorithm has such restrictions because it's aim is to be applicable to *all* graphs. Thus, to make the regularity lemma truly applicable we would have to give up our goal that the lemma should work for *every* graph and should be content with the fact that it works for *most* graphs. To ensure that this happens, we modify the Regular Partition Algorithm so that instead of constructing a regular partition, we find an *approximately* regular partition. Such a partition should be much easier to construct. We have the following 3 major modifications to the Regular Partition Algorithm.

Modification 1: We want to decrease the cardinality of atoms in each iteration. In the above Refinement Algorithm the cardinality of the atoms may be 2^{k-1}, where k is the number of classes in the current partition. This is because the algorithm tries to find all the possible ε-irregular pairs such that this information can then be embedded into the subsequent refinement procedure. Hence potentially each class may be involved with up to $k-1$ ε-irregular pairs. One way to avoid this problem is to bound this number. To do so, instead of using all the ε-irregular pairs, we only use some of them. Specifically, in this paper, for each class we consider at most one ε-irregular pair that involves the given class.

By doing this we reduce the number of atoms to at most 2. We observe that in spite of the crude approximation, this seems to work well in practice.

Modification 2: We want to bound the rate by which the class size decreases in each iteration. As we have at most 2 atoms for each class, we could significantly increase m used in the Refinement Algorithm as $m = \frac{|V_i|}{l}$, where a typical value of l could be 3 or 4, much smaller than 4^k. We call this user defined parameter l the refinement number.

Modification 3: Modification 2 might cause the size of the exceptional class to increase too fast. Indeed, by using a smaller l, we risk putting $\frac{1}{l}$ portion of all vertices into V_0 after each iteration. To overcome this drawback, we "recycle" most of V_0, i.e. we move back most of the vertices from V_0. Here is the modified Refinement Algorithm.

Modified Refinement Algorithm: *Given a γ-irregular equitable partition P of the vertex set $V = V_0 \cup V_1 \cup \ldots \cup V_k$ with $\gamma = \frac{\varepsilon^4}{16}$ and refinement number l, construct a new partition Q.*

For each pair (V_s, V_t), $1 \leq s < t \leq k$, we apply Lemma 1 with $A = V_s$, $B = V_t$ and ε. For a fixed s if (V_s, V_t) is found to be ε-regular for all $t \neq s$ we do nothing, i.e. V_s is one atom. Otherwise, we select one ε-irregular pair (V_s, V_t) randomly and the corresponding certificate partitions V_s into two atoms. Set $m = \lfloor \frac{|V_i|}{l} \rfloor$, $1 \leq i \leq k$. Then we choose a collection Q' of pairwise disjoint subsets of V such that every member of Q' has cardinality m and every atom A contains exactly $\lfloor \frac{|A|}{m} \rfloor$ members of Q'. Then we unite the leftover vertices in each V_s, we select one more subset of size m from these vertices and add these sets to Q' resulting in the partition Q. The collection Q is an equitable partition of V into at most $1 + lk$ classes.

Now we present our modified Regular Partition Algorithm. There are three main parameters to be selected by the user: ε, the refinement number l and h the minimum class size when we must halt the refinement procedure. h is used to ensure that if the class size has gone too small then the procedure should not continue.

Modified Regular Partition Algorithm (or the Practical Regularity Lemma):

Given a graph G and parameters ε, l, h, construct an approximately ε-regular partition.

1. **Initial partition:** *Arbitrarily divide the vertices of G into an equitable partition P_1 with classes $V_0, V_1, \ldots, V_l$, where $|V_1| = \lfloor \frac{n}{l} \rfloor$ and hence $|V_0| < l$. Denote $k_1 = l$.*

2. **Check size and regularity:** *If $|V_i| < h$, $1 \leq i \leq k$, then halt. Otherwise for every pair (V_s, V_t) of P_i, verify if it is ε-regular or find $X \subset V_s, Y \subset V_t, |X| \geq \frac{\varepsilon^4}{16}|V_s|, |Y| \geq \frac{\varepsilon^4}{16}|V_t|$, such that $|d(X,Y) - d(V_s,V_t)| \geq \varepsilon^4$.*

3. **Count regular pairs:** *If there are at most εk_i^2 pairs that are not verified as ε-regular, then halt. P_i is an ε-regular partition.*

4. **Refinement:** *Otherwise apply the Modified Refinement Algorithm, where $P = P_i, k = k_i, \gamma = \frac{\varepsilon^4}{16}$, and obtain a partition Q with $1 + lk_i$ classes.*

5. **Iteration:** *Let $k_{i+1} = lk_i, P_{i+1} = Q, i = i + 1$, and go to step 2.*

4.2 A Two Phase Strategy for Clustering

To make the regularity lemma applicable in clustering settings, we adopt the following two phase strategy (Figure 4.1):

1. **Application of the Practical Regularity Lemma:** In the first stage we apply the practical regularity lemma as described in the previous section to obtain an approximately regular partition of the graph representing the data. Once such a partition has been obtained, the reduced graph as described in Definition 7 could be constructed from the partition.

2. **Clustering the Reduced Graph:** The reduced graph as constructed above would preserve most of the properties of the original graph (see [69]). This implies that any changes made in the reduced graph would also reflect in the original graph. Thus, clustering the reduced graph would also yield a clustering of the original graph. We apply spectral clustering (though any other pairwise clustering technique could be used) on the reduced graph to get a partitioning and then project it back to the higher dimension. Recall that vertices in the exceptional set V_0, are leftovers from the refinement process and must be assigned to the clusters obtained. Thus in the end

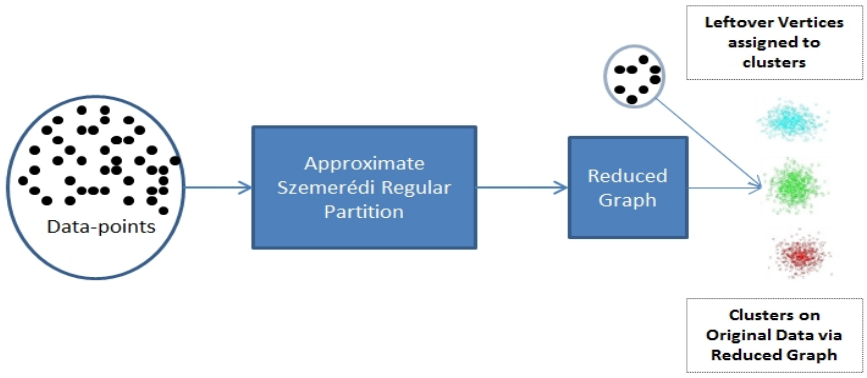

Figure 4.1: A Two Phase Strategy for Clustering

these leftover vertices are redistributed amongst the clusters using k-nearest neighbour classifier to get the final grouping.

4.3 Empirical Validation

In this section we present extensive experimental results to indicate the efficacy of this approach both by employing it for clustering on a number of benchmark datasets and for the task of image segmentation. We also compare the results with spectral clustering (both for the fully connected graph and k-nearest neighbour graphs) in terms for accuracy. We also report some results that indicate the amount of compression obtained by constructing the reduced graph.

As discussed later, the results also directly point to a number of promising directions of future work. We first review the datasets considered and the metrics used for comparisons.

4.3.1 Datasets and Metrics Used

The datasets considered for empirical validation were taken from the University of California, Irvine machine learning repository [47]. A total of 12 datasets were used for validation. We considered datasets with real valued features and associated labels or ground truth. In some datasets (as described below) that had a large number of real valued features, we removed categorical features to make it easier to cluster. Unless otherwise mentioned, the number of clusters was chosen so as to equal the number of classes in the dataset (i.e. if

the number of classes in the ground truth is 4, then the clustering results are for k = 4 etc). An attempt was made to pick a wide variety of datasets i.e. with integer features, binary features, synthetic datasets and of course real world datasets with both very high and small dimensionality.

The following datasets were considered: (1) Red Wine (R-Wine) and (2) White Wine (W-Wine) are two datasets having 1599 and 4898 datapoints respectively, each having 11 features. The target measures wine quality on a scale of 0-10. Though both are ten class problems, they only contain labels for 6 and 7 classes respectively. (3) The Arcene dataset (Arcene) has data for the task of distinguishing cancer from normal patterns from mass spectroscopic data. Thus it is a 2-class problem and was used in the NIPS 2003 feature selection challenge. The data consists of a train set with 100 points, a validation set with 100 points and a test set with 700 points (the test set does not come with labels). However, since we are not making any prediction as such we can combine the train and validation sets here and use it as one dataset. Thus, this dataset has 200 datapoints with each data instance described by 10000 features. Given this very high dimensionality, this should be an interesting dataset to experiment on. (4) The Blood Transfusion Dataset (Blood-T) has 748 data-instances with 4 features each. The task is to predict whether a person donated blood in a certain month (March, 2007) and hence is a two-class problem. (5) The Ionosphere dataset (Ionos) has 351 data instances each of which is 34 dimensional feature vector having information about radar returns from the Ionosphere. The task is to classify the radar returns as "good" i.e. those showing some structure in the ionosphere and "bad" i.e. those returns that do not.

(6) The Wisconsin Breast cancer dataset (Cancer) has 699 datapoints and 9 attributes. The task is classifying a point as benign or malignant. Some rows having missing values were deleted so the actual number of datapoints considered is 683. All the features in this dataset are integer valued. (7) The Pima Indian diabetes dataset (Pima) is a standard dataset provided by the National Institute of Diabetes and Digestive and Kidney diseases. It has 8 attributes for 768 patients (all female of the Pima Indian heritage). The 2-class task for this dataset is to predict whether or not a patient has diabetes. (8) The Vertebral Column dataset (Vertebral-1) has data for 310 orthopaedic patients with 6 bio-mechanical

features. The task is to classify patients into either normal, disk hernia or spondilolysthesis and alternately as normal and abnormal. The second task (9) (Vertebral-2) is considered as another dataset. (10) The Steel Plates Faults Dataset (Steel) is a 7-class dataset having 1941 instances and 27 attributes. The goal is to recognize faults of seven different types. (11) The Musk 2 (Musk) dataset has information about a set of 102 molecules of which 39 are judged by human experts to be musks and the rest judged to be non-musks. Thus, this is a two class problem in which the goal is to predict whether a new molecule will be a musk or not. However, considering all the possible conformations, this dataset has 6598 examples,each with 168 features. Two of which are deleted. (12) Haberman's Survival (Haberman) has data from a study conducted on the survival of patients who had undergone surgery for breast cancer. It only has three features and 306 points, the task is to predict if the patient (described by three features each) survived for more than five years or not after surgery, thus being a two class problem.

Now that we have briefly discussed all the datasets considered in the study we now discuss the metric used for comparison with other clustering algorithms. For evaluating the quality of clustering, we follow the approach of [102] and use the cluster accuracy as a measure. This is an interesting combinatorial measure that relies on the confusion matrix. The measure is defined as:

$$Accuracy = 100 * \left(\frac{\sum_{i=1}^{n} \delta(y_i, map(c_i))}{n} \right)$$

Where, n is the number of data-points considered, y_i represents the true label (ground truth) while c_i is obtained cluster label of data-point x_i. The function $\delta(y, c)$ equals one if the true and the obtained labels match ($y = c$) and 0 if they don't. The function map is basically a permutation function that maps each cluster label to the true label. An optimal match can be found by using the Hungarian Method for the assignment problem [70].

In the next section we report some experiments and results on one of the above datasets as a case study.

4.3.2 Case Study

Before reporting comparative results on benchmark datasets, we first consider one dataset as a case study. While experiments reported in this case study were carried on all the bench-

Table 4.1: Clustering Results on Red Wine Dataset by Other Methods

Clustering Method	k = 6	k = 3
Self Tuned Spectral (k-nearest neighbor graph)	26.0163	40.6504
Self Tuned Spectral (fully connected graph)	25.8286	37.3984
k-means	23.8899	37.0857

mark datasets considered, the purpose here is to illustrate the investigations conducted at each stage of application of the regularity lemma. An auxiliary purpose is also to underline a set of guidelines on what changes to the practical regularity lemma proved to be useful. Some of the points that we extracted from such experiments have also pointed to some very interesting directions for future work and refinement that are outlined in Section **??**.

For this task we consider the Red Wine dataset which has 1599 instances with 11 attributes each. For the Red Wine dataset, the number of classes involved is six. It must be noted though that the class distribution in this dataset is pretty skewed (with the various classes having 10, 53, 681, 638, 199 and 18 datapoints respectively), this makes clustering this dataset quite difficult when k = 6. We however consider both k = 6 and k = 3 to compare results with spectral clustering.

Recall that our method has two meta-parameters that need to be user specified (or estimated by cross-validation) - ε and l. The first set of experiments thus explore the accuracy landscape of regularity clustering spanned over these two parameters. Care has to be taken that ε isn't too large or small, so we consider 25 linearly spaced values of ε between 0.15 and 0.50. The "next refinement size", l as noted in Section 4.1 can not be too large. Since it can only take integer values, we consider six values from 2 to 7. For the sake of comparison, we also obtain clustering results on the same dataset with spectral clustering with self tuning [103] (both using all connected and k-nearest neighbour graph versions) and k-means clustering. We pick the variant of spectral clustering that is known to return the best results to make for a good comparison. Figure 4.2 gives the accuracy of the regularity clustering on a grid of ε and l. Even though this plot is only for exploratory purposes, it shows that the accuracy landscape is in general much better than the accuracy obtained by Spectral Clustering for this dataset. In this particular dataset it appears that

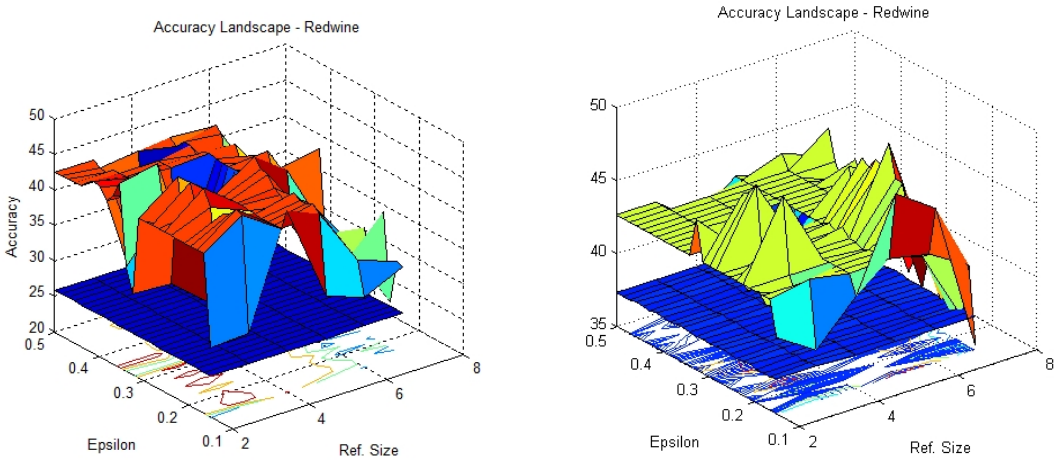

Figure 4.2: Accuracy Landscape for Regularity Clustering on the Red Wine Dataset for different values of ε and refinement size l (with k = 6 on the left and k = 3 on the right). The Plane cutting through in blue represents accuracy by running self-tuned spectral clustering using the fully connected similarity graph.

the better performance of regularity clustering is not really too dependent on the choice of ε and l. We summarize results obtained by other methods on the Red Wine dataset in Table 4.1.

An important aspect of the regularity clustering method is that by using a modified constructive version of the regularity lemma we obtain a much reduced representation of the original data. This reduced graph is often much smaller than the original graph representing the dataset. The size of the reduced graph depends both on ε and l. However, in our observation it is more sensitive to changes to l and understandably so. From the grid for ε and l we take three rows to illustrate the obtained sizes of the reduced graph (more precisely, the dimensions of the affinity matrix of the reduced graph). We compare these numbers with the original dataset size. The compression obtained is quite striking. As we note in the results over the benchmark datasets in section 4.3.3 , this compression is quite big in larger datasets.

In the proof of the regularity lemma, Szemerédi used an iterative greedy method and a clever potential function. We defined this function earlier in definition 3. This function is also called the index of a partition and measures how close the partition is to being ε regular.

Table 4.2: Reduced Graph Sizes. Original Affinity Matrix size : 1599 × 1599

ε \ l	**2**	**3**	**4**	**5**	**6**	**7**
0.15	16 × 16	27 × 27	27 × 27	27 × 27	36 × 36	49 × 49
0.33	49 × 49	49 × 49	66 × 66	66 × 66	66 × 66	66 × 66
0.50	66 × 66	66 × 66	66 × 66	66 × 66	66 × 66	66 × 66

Table 4.3: Illustration of Increase in Potential

(ε,l) \ $ind(P)$	$ind(P_1)$	$ind(P_2)$	$ind(P_3)$	$ind(P_4)$
0.15, 2	0.1966	0.2892	0.3321	0.3539
0.33, 2	0.1966	0.2883	0.3321	0.3683
0.50, 2	0.1965	0.2968	0.3411	0.3657

Refining a partition can only increase the potential of the new partition. If an irregular pair (V_i, V_j) is split then the potential would increase significantly. Like mentioned in Section **??**, each class given a partition of k classes, could potentially be ε-irregular with $k-1$ other classes. If all such pairs are considered then the number of elements in each refinement increase exponentially. In Section 4.1 we approximated this by considering for each class at most one ε-irregular pair. While this seems like a crude approximation, the results for final clustering are pretty good nonetheless. By results such as those illustrated in Table 4.3, we give a more direct evidence that even taking at most one ε-irregular pair still leads to a good increase in the potential function in each refinement. Another interesting thing that we note in such observations is that if we take ε sufficiently high, we do get a regular partition in just a few iterations. This is rather surprising even if the ε is large. By these set of experiments it was noticed that a small l tended to get more regular partitions with a somewhat large ε. There were a few cases where regular partitions were observed for even larger l. A few examples where this was noticed in the Red Wine dataset are mentioned in Table 4.4.

It is mentioned above that for refinement we only consider one ε-irregular pair for each

Table 4.4: Regular Partitions with required number of regular pairs and actual number present

(ε, l)	# for ε-regularity	# of Reg. Pairs	# Iterations
0.6, 2	1180	1293	6
0.7, 6	352	391	2
0.7, 7	506	671	2

class. Strategies for picking this irregular pair were also investigated and compared. Two natural strategies were tried: Picking a random irregular pair from the set of all irregular pairs and picking the most irregular pair. Intuitively, the second strategy should yield better results, but it was observed that this was rarely the case. It should be noted that the accuracy results reported earlier were based on choosing a random irregular pair. Figure 4.3 depicts the accuracy versus the meta-parameters when the most irregular pair is chosen instead.

Another aspect of the implementation that was investigated in detail was attempting to model the intra-cluster similarities. The practical regularity lemma gives a method to model inter-cluster variations. However for clustering, modelling the intra-cluster variations are as important. One way of doing this is to sort the subsets in the refinement process by decreasing degree. By ordering subsets by degree it could be ensured that vertices with higher degrees remain in the same subset while the vertices with the lowest degree are put in the exceptional set. This seems intuitive as the vertices with the lowest degree would perhaps be left overs. An unexpected advantage of ordering vertices is that the randomness in the algorithm is substantially reduced. Using the strategy outlined above (and with a random irregular pair and no ordering of vertices) causes some variations in the results on each run with the same meta-parameters. By ordering vertices this randomness is substantially reduced and the results are more stable. Results obtained by ordering vertices were similar as those depicted in Figure 4.3. Like for the most irregular pair, ordering vertices did not necessarily lead to better accuracy in all datasets. We consider exploring this aspect of the methodology an important aspect to fine-tune and refine. For now we only report results when the vertices are not ordered.

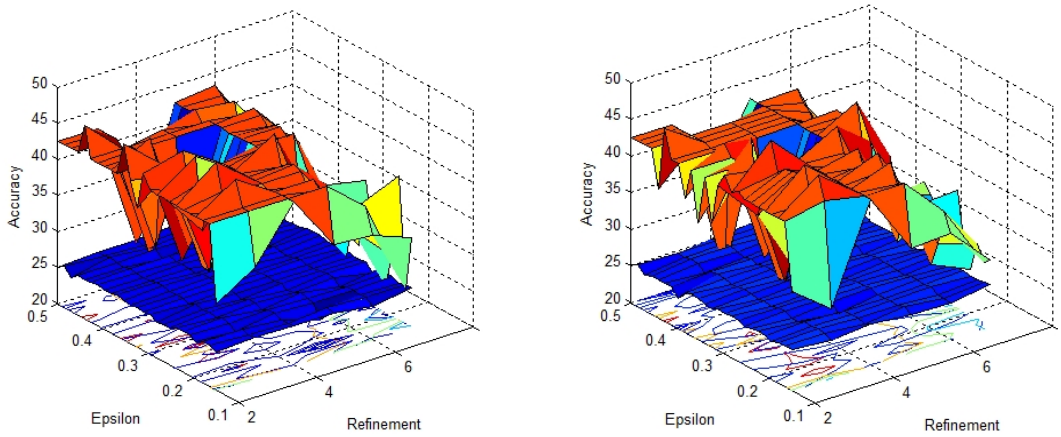

Figure 4.3: Accuracy Landscape on the Red Wine Dataset (with k = 6 on the left and k = 3 on the right) when the most irregular pair is considered in each refinement. The Plane cutting through in blue represents accuracy by running self-tuned spectral clustering using the fully connected similarity graph.

Finally, before reporting results we must make a comment on constructing the reduced graph. The reduced graph was defined in Definition 7. But note that there is some ambiguity in our case when it comes to constructing the reduced graph. We mention that the reduced graph G^R might be constructed such that the vertices in the reduced graph correspond to the classes in the partition and the edges are associated to the ε-regular pairs between classes with density above d. For constructing the reduced graph we thus tried two approaches: First, after the refinement process was halted we simply used all the classes obtained and found the degree between each class pair. For all pairs of classes this corresponds to the adjacency matrix which could then be input to a spectral clustering algorithm. Secondly, we can construct the reduced graph such that only those entries in the adjacency matrix are recorded when the pair of classes under consideration is ε-regular. Otherwise the matrix entry is set to zero. The second approach seems compelling, however in practice we don't report it to be useful. This is because in many cases the number of regular pairs is quite less (especially when ε is small) making the matrix too sparse, making it difficult to find the eigenvectors. We thus use the first approach. We contend that this approach works well because the classes that we consider (and thus the densities between them) are still obtained after the modified refinement procedure of the regularity lemma. Thus enough

information is still embedded in the reduced graph.

We now report clustering results on a number of benchmark datasets.

4.3.3 Clustering Results on Benchmark Datasets

In this section we report results on a number of datasets described earlier in Section 4.3.1. We do a five fold cross-validation on each of the datasets, where a validation set is used to learn the meta parameters for the data. The accuracy reported is the clustering quality on the rest of the data after using the learned parameters from the validation set. We use a grid-search to learn the meta-parameters. Initially a coarse grid is initialized with a set of 25 linearly spaced values for ε between 0.15 and 0.50 (we don't want ε to be outside this range). For l we simply pick values from 2 to 7 simply because that is the only practical range that we are looking at. This also justifies the use of grid-search in the following way: In the initial coarse grid search, because l can take only integer values, once a good value of l (with ε) has been identified the search becomes one dimensional (looking for the best ε given l) in the subsequent more finer grid searches.

We compare our results with a fixed σ spectral clustering with both a fully connected graph and a k-nearest neighbour graph. For the sake of comparison we also include results for k-means on the entire dataset. We report results both when the vertices are not ordered by their degrees before refinement as mentioned earlier. We also report results on the compression that was achieved on each dataset in Table 4.6. More precisely, we compare the sizes of the original affinity matrix of the entire dataset and the affinity matrix of the reduced graph.

4.4 Discussion and Future Directions

A thorough analysis of the different aspects of the experiments and their results was done in Section 4.3.2. But we summarize some of them nonetheless. In the results reported in the previous section we observed that the regularity clustering method, as indicated by the clustering accuracies is quite powerful. The results reported were when no ordering of vertices was done prior to refinement, a random irregular pair was selected in the refinement

Table 4.5: Clustering Results on UCI Datasets with No Ordering of Vertices

Dataset	**Regularity**	**Spectral (knn)**	**Spectral (Full)**	**k-means**
R-Wine	**47.0919**	23.9525	23.9524	23.8899
W-Wine	**44.7509**	23.1319	20.5798	23.8465
Arcene	**68**	61	62	59
Blood-T	**76.2032**	65.1070	66.2331	72.3262
Ionos	**74.0741**	70.0855	70.6553	71.2251
Cancer	93.5578	**97.2182**	97.2173	96.0469
Pima	**65.1042**	51.5625	60.8073	63.0156
Vertebral-1	67.7419	**74.5161**	71.9355	67.0968
Vertebral-2	**70**	49.3948	48.3871	65.4839
Steel	**42.5554**	29.0057	34.7244	29.7785
Musk	**84.5862**	53.9103	53.6072	53.9861
Haberman	**73.5294**	52.2876	51.9608	52.2876

Table 4.6: Compression Obtained on the UCI Datasets

Dataset	**No. of Features**	**Original Dimension**	**Reduced Dimension**
R-Wine	11	1599×1599	49×49
W-Wine	11	4898×4898	125×125
Arcene	10000	200×200	9×9
Blood-T	4	748×748	49×49
Ionos	34	351×351	25×25
Cancer	9	683×683	52×52
Pima	8	768×768	52×52
Vertebral-1	6	310×310	25×25
Vertebral-2	6	310×310	25×25
Steel	27	1941×1941	54×54
Musk	166	6598×6598	126×126
Haberman	3	306×306	16×16

process and the reduced graph was constructed as outlined in Section 4.3.2. Ordering of vertices led to more stable but not necessarily better results, the same was the case when vertices were ordered prior to refinement. This is one aspect that we think needs a deeper look at. The experimental results though promising did need tuning of parameters. Like in the case of spectral clustering where choosing the correct σ was quite time consuming till the self-tuned version [103] was introduced. It would be interesting if the aspect of tuning of Regularity Clustering is looked deeply at.

In fact we believe that this work opens up a lot of potential research problems. This isn't surprising because the fusion of the pervasive nature of clustering and deep theoretical results in extremal combinatorics concerning the regularity lemma naturally makes their intersection an area fecund for further investigation. We now identify some specific problems below that we think merit significant further research.

In our work we modify the algorithmic version of the regularity lemma due to Alon *et al.* [2] for constructing a reduced graph. As we described earlier, there is another constructive version of the regularity lemma due to Frieze and Kannan [48]. This version is based on the particularly intriguing connection between singular values and regularity and also has a different method to find the certificate if a given pair is irregular. It would thus be natural to investigate using this version and see how it compares to the results obtained by modifying the Alon *et al.* method. Preliminary work with the Frieze-Kannan version indicates that performance is similar to the Alon *et al.* version.

Yet another avenue for future work in the process of refinement other than employing the version due to Frieze and Kannan is exploring the constructive versions such as due to Fischer [44]. They give a new approach for finding regular partitions which is quite different form the previous approaches. All the previous ones work to find partitions of the tower type, while this paper gives a method to find a smaller regular partition if one exists in the graph. Employing this methodology for refinement instead of using an approximate version of the algorithmic regularity lemma could also be a fruitful direction of work.

The spectral clustering pipeline involves two stages: Construction of the pairwise affinity matrix (and hence the graph Laplacian) and eigendecomposition of the output of this stage for dimensionality reduction. It is on this reduced dimension that we run a traditional

clustering method such as k-means to obtain the final clustering. As we note earlier, both of these stages require significant computation and have inspired research to get around these bottlenecks. In our work we give a method to substantially ease the second bottleneck. The reduced graph gives a highly compressed representation of the original graph and eignedecomposition of this much smaller graph is notably easier and faster. However, this speed up is given the original affinity matrix, which makes the applicability of our method slightly more wider than the spectral clustering framework in its original form [71]. To make the entire method far more powerful with a very wide range of applicability we need to make changes to the first stage of the bottleneck. It must be noted that the practical regularity lemma is applicable when the graph is dense. However, there are sparse versions of the regularity lemma that work with, as the name indicates, sparse graphs. Utilizing the sparse regularity lemma for refinement in our method could be used to get around the first bottleneck in the framework above as well (this would be possible as it work allow us to work with k-nearest neighbor graphs). And thus together, the regularity clustering method could be made really powerful.

One of the most attractive notions of pairwise clustering methods is that they give a more "global" view of the data. Given enough number of data-points we could at least approximately get an idea about the geometry of the data, thus significantly improving it's performance over traditional centroid based methods which are more "local". As pointed out by Fowkles *et al.* [46],when seen through the lens of computer vision this makes such "global" clustering methods (for segmentation) closer to the original Gestaltian views on form and perception that for a human an image is much more than a mere collection of objects. However, while pairwise affinities capture a more global view of the data, it is not necessary that the relationship between data-points in most domains has to be dyadic and thus restricting it to being dyadic might lead to loss of information. Indeed, it might be the case that the relationship between data-points is triadic, tetradic or even higher. Thus, this natural extension has led to work on clustering methods for such problems, which can be naturally formulated as a hypergraph partitioning problem [1], [104], [16]. There are a number of important results that extend the regularity lemma to hypergraphs [86], [87],[54],[23]. It is thus natural that our methodology could be extended to hypergraphs and

then used for hypergraph clustering. This seems to be a particularly promising problem.

In final summary, our work gives a way to harness the regularity lemma for the task of clustering. We report results on a number of benchmark datasets which strongly indicate that the method is quite powerful. Based on this work we also suggest a number of possible avenues for future work towards improving and generalizing this methodology.

Part III

Clustering for Prediction

Chapter 5

Clustering for Prediction

Clustering is probably the most used exploratory data analysis technique across disciplines and is frequently employed to get an intuition about the structure of the data, for finding meaningful groups, also for feature extraction and summarizing. Given a space X, clustering could be thought of as a partitioning of this space into K parts i.e. $f : X \longmapsto \{1, \ldots, K\}$. This partitioning is done by optimizing some internal clustering criteria such as the intra-cluster distances etc. The value of is found usually by employing a second criterion that measures the robustness of the partitioning. While clustering is useful for data analysis and as a preprocessing step for a number of learning tasks, we are interested in the specific pre-processing task of using clustering to gain more information about the data to improve prediction accuracy. This leads to the questions: Can clustering of unlabeled data give any new information that can aid a classification task? It has been hinted in the literature that clustering of unlabeled data should help in a classification task as clustering can also be thought of as separating classes. It is not clear if clustering could help in a regression task, though there is some evidence [95]. Another question that could be asked is: Can a number of predictions obtained by varying clustering parameters give us access to new information that can be combined together to improve prediction accuracy even more? Can the idea of clustering as a predictor be formalized? Previous work (discussed in the next section) comprehensively answers at least the third question. This is an important question to ask since the answer justifies using clustering in a prediction task.

One of the motivations to this work is one of the co-authors successful participation in

the 2010 KDD Cup, which involved a prediction task on an educational dataset. Methods such as Bagged Decision Trees were used to get the second position in the student category. The dataset had instances for a number of students. Since students can be crudely binned into categories in terms of learning rate, forgetting rate etc., a natural question to ask is if clustering the students and trying to find such groups would aid in classification accuracy. This question was crudely tested in the 2010 UCSD Data Mining competition (an e-commerce task) in which the fourth position was secured using this clustering method alone. Motivated by the success of this technique, an internal graduate Machine Learning course competition was organized at Worcester Polytechnic Institute (WPI) that explored this notion further. This idea of using clustering coupled with simple predictors beat more complex methods such as Support Vector Machines and Random Forests on the KDD cup development set. This also led to a paper [95] that explored this idea in an educational dataset. This paper essentially develops this notion further. The rest of this chapter is organized as follows: Section 5.1 reviews some work on clustering, such as a theoretical justification of using clustering for a classification task. Sections 5.2,5.3 discuss the idea of using clustering in conjunction with a predictor in more detail, also providing some more work and intuition on how we can use clustering to improve accuracy on a prediction task. Section 5.4 reviews some similar work from the literature. Section 5.5 talks of the empirical study carried out and overview of the results obtained and section 5.6 has a discussion of observations and open questions.

5.1 PAC-MDL Bounds and Clustering as Classification

One of the most basic results in Learning Theory is the Occams Razor [11]i.e. if a set of m training examples can be described by a hypothesis using only $k << m$ bits, then we can be quite sure that the hypothesis generalizes well to unseen data. Another way of stating this is that compression implies learning for the description language of the hypothesis. If compression means learning then making predictions would mean decompression. The notion of *compression implies learning* for different description languages has lead to a number of important sample complexity bounds [101], [45] and has been generalized to any

description language by Blum and Langford [10]. This generalization, called the PAC-MDL bound gives a handle on understanding the generalization error and the tradeoff between good representations of the data and over-fitting it. Clustering too can be seen as a trade-off between the quality of the representing groups in the data and the complexity of the same.

The said PAC-MDL bound is defined for a transductive setting [101] and essentially states that it is quite unlikely that a transductive classifier that does well on the training set will do badly on the test set. This can be formalized as follows: Consider we have a training set S_{train} having m labeled examples and a test set S_{test} having n unlabeled examples which are drawn independently from a distribution D. If X is the instance and Y the target, then $S_{train} = \{X^m, Y^m\}$ and $S_{test} = \{X^n, Y^n\}$ with $Y = \{1, \ldots, l\}$. Given any compression procedure as discussed in the previous paragraph which could be represented as $A : (X \times Y)^m \times X^n \rightarrow \{0,1\}^*$ there would be a decompression procedure $B : X^{m+n} \times \{0,1\}^* \rightarrow Y^{m+n}$. For this compression-decompression pair the transmitted string σ would be the transductive classifier $\sigma : X^{m+n} \rightarrow Y^{m+n}$ that assigns labels to the examples. For a description language the bound on the test error ($\hat{\sigma}_{test}$) as a function of the error on the training set is given by the PAC-MDL bound [10], [5]:

Theorem 4 (PAC-MDL [10], [5]) *For any given distribution D and for the set of all description languages $L = \{\sigma\}$ with probability $1 - \delta$ over the train and test sets:*

$$S_{train}, S_{test} \sim \ D^{m+n} : \ \forall \sigma,$$

$$\hat{\sigma}_{test} \leq bmax(m, n, \hat{\sigma}_{train}, 2^{-|\sigma|}\delta)$$

While the PAC-MDL bound is used for a transductive setting Banerjee and Langford [5] show that clustering can be converted to a transductive classification problem. They also demonstrate that for a description language $L = \{\sigma\}$ to be a *valid* description language, it must be an instantaneous code and hence satisfy Krafts inequality [26]. For the case of clustering, with c clusters and l labels, the family of descriptions $L = \{\sigma\}$ has size l^c. This set can be encoded by $|\sigma| = clog(l)$ bits. Since L satisfies Krafts inequality, it is a valid description language. This essentially gives an information theoretic justification of using clustering as a transductive classifier and also gives a set of PAC-MDL bounds on the same.

The above review in simple terms states the following: Since clustering is a scheme for information compression. It will thus (when stated as a transductive problem for simplicity) most likely improve the prediction error. The PAC-MDL bounds that formalize this notion can be used without any loss of generality as an intuitive explanation of why clustering could be used in conjunction with a predictor as a pre-processing step. The next section discusses the notion of clustering for prediction further.

5.2 A Simple Bagging Strategy

Clustering is used to mine structure in the data. According to a pre-defined metric data-points in one group are by definition highly similar to each other than to data-points from other groups/clusters. One useful way of looking at this is thinking of clustering as [28]: Consider a dataset that is obtained by sampling a collection of distributions $\{D_1, D_2, \ldots, D_k\}$ with associated weights $\{w_1, w_2 \ldots w_k\}$ such that $\sum_i w_i = 1$ i.e. from each distribution D_i a point is picked with probability w_i. Now given the dataset, the idea behind clustering is to identify these distinct distributions that might have generated it and assign points in the dataset into different groups accordingly. This new representation is more concise. Following from the above and from the discussion in section II: Given a dataset, clustering it gives a compressed representation (albeit lossy). This can be thought of as giving the data to an operator as input (k-means for example) that gives an output of the same data but taking much fewer bits to represent it. This transformation tells us something interesting about the data and its structure which could be exploited to improve the predictive power. One potential way of doing so is by training a separate predictor on each cluster rather than train a single predictor on the entire dataset.

Consider a sample regression task (Fig. 5.1): Suppose we first cluster the dataset into k clusters using an algorithm such as k-means. A separate linear regression model is then trained on each of these clusters (any other model can be used in place of linear regression). Let us call each such model a "Cluster Model". All of the k Cluster Models together can be thought of as forming a more complex model that we call a "Prediction Model". We represent a prediction model as PM_k, with the subscript indicating the number of cluster

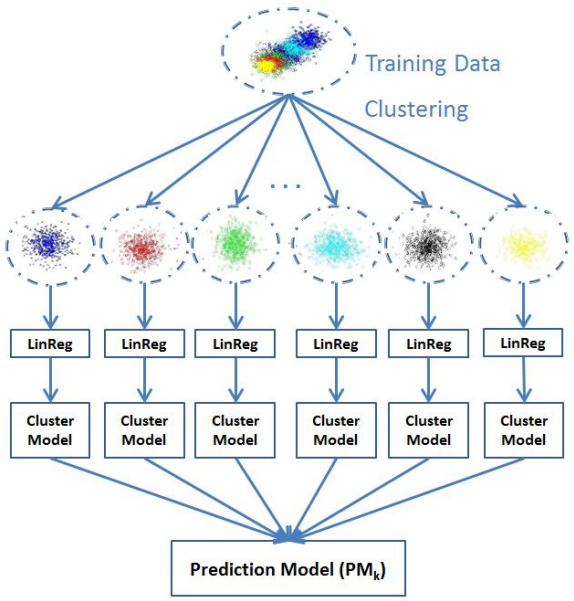

Figure 5.1: A "Prediction Model". A "Prediction Model" is composed of k cluster models (PM_k). It should be noted that any other method for regression could be used in place of Linear Regression

models in the given prediction model (which in turn will obviously equal the number of clusters). To summarize, to train a "prediction model", the following steps are followed:

1. Cluster the training data into k partitions.
2. For each partition train a separate classifier/predictor using the points inside that cluster as its training set.
3. Each such predictor represents a model of the cluster, and hence is called the cluster model.

Once a prediction model is obtained, making a prediction of a point from the test set would involve the following (Fig. 5.2):

Making predictions for a point from the test set would thus involve two steps:

1. Identify the cluster to which the test point belongs.

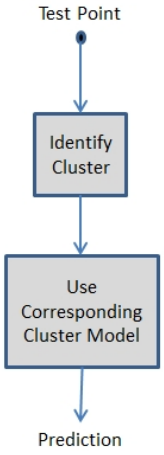

Figure 5.2: Mapping a test point to a cluster to make a prediction on it

2. Use the Cluster Model of the identified cluster to make the prediction for that data point.

It must be noted that PM_k would simply be our predictor fit on the entire data set (for the above example it would be fitting a linear regression model on the dataset, we can think of the entire dataset as one cluster).

5.2.1 k as a Tunable Parameter

The previous section describes a way by which clustering could be used to construct what we call a "Prediction Model". Building on the generic method, using the number of clusters k in k-means (or any other clustering that requires number of clusters to be input) as a free parameter, multiple prediction models can be obtained (Fig. 5.3) i.e. k can be varied from 1 to a value K and a Prediction Model for each instance can be obtained. For example if K = 3, there would be three prediction models: PM_1 (predictor trained on the entire dataset), PM_2 (predictors trained on two clusters), and PM_3 (predictors trained on three clusters). These K prediction models are then employed to make a set of K distinct predictions on the test set using the two step procedure for mapping and making predictions of test points sketched in the previous section. Before looking at how these K predictions can be of value, it must be noted that

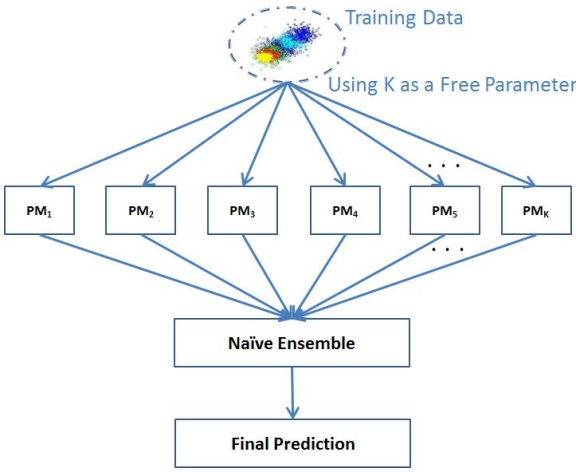

Figure 5.3: Generation of multiple prediction models by using k as a free parameter. Each of these prediction models will make a prediction on the test set. These predictions can then be combined together by a nave ensemble to get a final prediction.

1. Cluster models in different prediction models are different.

2. There might indeed exist a prediction model PMi for some arbitrary number of clusters that would have higher prediction accuracy than PM1. The reverse might also be true

The second factor i.e whether some arbitrary PM_i would do better than PM_1 would depend on two main factors: Clusterabilty of the dataset [91] and the choice of predictor.

Even if an arbitrary PM_i does not return higher accuracy than PM_1, a couple of questions of considerable interest would be: How good are the predictions made by each individual prediction model? How diverse are the predictions made by the various prediction models? If there is indeed some diversity in the error patterns in predictions made by the various prediction models, the next step would be to combine the predictions together to perhaps get a stronger prediction.

5.3 Combining Predictions

Before looking at combining predictions, it is useful to understand how the predictions made by the various prediction models might be diverse and why combining diverse predictions

might be helpful.

5.3.1 An Information Theoretic View of Clustering

As discussed in section 5.1, clustering seems useful for prediction as it is basically a scheme for data compression. By compression we learn something interesting about the structure and the regularities in the data that can be used to perhaps improve the prediction accuracy. A simple method to do so was outlined in section 5.2. Interestingly however, how much compression we can achieve will depend on what k (number of clusters) is chosen. A question that arises is: Is there at least some difference in the information content in these different cases? Lets consider this question in some detail: Consider k-means clustering; Now since the cluster centroids are found by optimizing a distortion function, the choice of this distortion function decides what information should be kept and what should not be. The distortion function for k-means is given by:

$$J(c, \mu) = \sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

μ_c is the cluster centroid to which a point x has been assigned. The data are described more concisely (and hence the compression) with all the points in a cluster approximated by their corresponding cluster centroids found using the distortion function. The rest of the irrelevant data is thrown away.

Previous work by Still & Bialek [91] has formalized and extended this notion of relevance. This formalization gives a tradeoff between the complexity of the model and the amount of relevant information. The tradeoff range gives an optimal number of clusters for a dataset of a finite size beyond which we begin to over-fit it. Other than this, the rate distortion theory applied to the problem of clustering also shows that the amount of relevant information coded at a certain clustering scale is different. Thus, in a sense there is no single best clustering of the data but a family of solutions that evolves with the tradeoff parameter [91]. This tradeoff in turn formalizes the notion of "clusterabilty" of the dataset and gives the valuable insight that at different values of this tradeoff we might get access to different information. While some of this information might be redundant, and some of it might be sampling noise, some information may also be unique to a grouping. We believe that it is

this source of novel information that lends at least some power to the use of clustering in a prediction task. In our case, it would lend some diversity to the predictions obtained by the various prediction models since each is trained at a different scale of clustering.

5.3.2 Ensemble Learning

When we have a set of diverse (and accurate) predictors, combining them together to obtain a single prediction leads to ensemble methods (ensemble methods can also be considered methods as ways of generating diverse and accurate individual predictors in the first place). Ensemble methods have seen a rapid growth in the past decade in the machine learning community (see [15], [35], [36]). An ensemble is a group of predictors each of which gives an estimate of a target variable. Ensemble learning is a way to combine these predictions with the goal that the generalization error of the combination is lesser than each of the individual predictors. The success of ensembling lies in the ability to exploit (or inject and exploit) diversity in the individual predictors. That is, if the individual predictors exhibit different patterns of generalization, then the strengths of each of the predictors can be combined to form a single stronger predictor. A lot of research in ensemble learning has gone into finding methods that encourage diversity in the predictors.

Dietterich [35] suggests three reasons why ensembles perform better than the individual predictors. The first reason is statistical. A learning algorithm can be considered as a way to search the space of hypotheses to identify the best hypothesis in it. The statistical problem is caused due to insufficient data. Due to this problem, the learning algorithm would give a set of different hypotheses with similar accuracy on the training data. By ensembling them, the risk of choosing the wrong hypothesis would be averaged out. The second reason is computational. Often, while looking for the best hypothesis, the algorithm might be stuck in local optima, thus giving us a bad hypothesis. By considering multiple such hypotheses, we can obtain a much better approximation to the true function. An example of the computational aspect is trying to train a neural network by restarting gradient descent a number of times to ensure that the result is better. The third reason is representational. Sometimes the true function might not be any hypothesis in the hypotheses space. By ensembling them, the representational space might be expanded to give a better approximation of the

true function. Given the discussion about ensemble methods, we now consider combining the predictions in the method in section 5.2.

5.3.3 Methodology for Combining Predictions

With each prediction model having access to different information about the data, combining them improves the representation and averages out the chance of finding an improper hypothesis. Hence we expect a combination to give an improvement in accuracy. As an example for improving representation, suppose a linear regression is to be used for training on the dataset. Such an arrangement will likely have a high bias on a real world dataset. Using linear regression on the clusters and not on the entire dataset gives a chance to expand the representational space and give a better fit to the data and increase variance.

As discussed in section 5.2, we obtain a set of K predictions by varying the value the free parameter k. These predictions can be combined by uniform averaging, weighted averaging or ensembling them together. The aim of our work is to show the utility of clustering in causing an improvement in accuracy, and hence though we can use ensemble methods to combine them together we show results by simple averaging only. Averaging the predictions in a regression task (equivalent to voting in a classification task) is probably the easiest way to combine them. First, the training set is clustered and by varying k, K prediction models are obtained. And then each of these prediction models are used to make a prediction on the test set. We thus obtain a set of K predictions on the test set. Averaging all these predictions might not be fruitful as some of them might be poor predictors and thus might prove to be detrimental to the prediction accuracy. Thus, a subset of the total number of predictions obtained must be averaged to improve accuracy. Like mentioned earlier, in place of uniform averaging, a weighted averaging or the use of an ensemble method could greatly improve the combined prediction.

5.4 Similarity with Existing Methods

Before looking at the empirical evaluation of the method so discussed, we compare this method with some papers that atleast talked of using clustering for prediction. We intro-

duced a simple yet effective bootstrap-aggregating meta-algorithm that uses clustering as means to bootstrap. This method can be thought of as a mixture of local experts similar to one discussed by Jacobs, Hinton *et al.* [61]. It is noteworthy however that unlike in other bagging methods which select a random subset of the data to bootstrap, this method has a specific expert for each "locality" i.e cluster; which can potentially lead to more interpretability. By varying the granularity of the clustering we are able to train a set of experts at different scales which leads to a set of diverse predictions amenable to ensembling together. For example, if a K of 10 is chosen then for each test point there are ten experts to "consult" for a prediction, one each at a different level of granularity (i.e for $k = 1$ there is an expert, at $k = 2$ there is another and so on till $k = 10$).

On their work on Statistical Predicate Invention, Kok & Domingos [65] use multiple clusterings to better capture the interactions between objects in relational learning. Deodhar & Ghosh [29] also mention the same, however they use it in co-clustering framework and both of these works do not combine the predictions at different scales.

5.5 Empirical Validation

In this section we report the mechanics of an empirical study performed on a number of benchmark datasets for the task of regression for three different predictors.

5.5.1 Algorithms

The algorithm used for clustering the various datasets for this set of experiments was the k-means algorithm. k-means finds a partition by optimizing a distortion function, and while it can be considered to converge in a certain sense (it can be stated to be Lyapunov Stable [74] and thus the objective function decreases monotonically), the distortion function for k-means is non-convex. It is thus sensitive to the choice of initial cluster centroids and returns sub-optimal solutions quite often. We randomly initialized k-means 200 times on each run and picked the best solution. For the prediction task (i.e. for training cluster models), Linear Regression, Step-Wise Linear Regression and Random Forests (for regression) were used. While this work can be extended to classification tasks as well, we do not discuss

them in this work.

5.5.2 Datasets

The datasets used for the empirical validation of this technique were taken from the University of California, Irvine Machine Learning repository [47]. Out of the 17 regression datasets available, those datasets were considered that did not have a large number of missing values or nominal attributes and thus we restricted ourselves to datasets having numerical attributes solely. Some of these datasets had more than one target variable. Such cases are reported as separate datasets.

The following datasets were considered (a) Breast Cancer Wisconsin Dataset (BREAST CANCER) has 569 data instances, each having 32 attributes. The prediction is for diagnosis (Benign or Malignant); (b) Cement Compressive Strength Dataset (COMPRESSIVE) has 1030 data points in total. Each data instance is described by 10 features [17]. The task is to predict the compressive strength (M Pa); (c) Concrete Slump I; (d) Concrete Slump II and (e) Concrete Slump III are essentially the same dataset (CONCRETE SLUMP) with the target attribute different in each case. This dataset has 103 data instances and 10 attributes, out of which 3 are target attributes (slump, flow and compressive strength); (f) The Forest Fires Dataset (FIRES) is one of the hardest regression datasets available[18]. It has 13 attributes and a total size of 513 observations. The task is to predict area burned in square kilometers; (g) Housing Dataset (HOUSING) has 506 instances of houses around the suburbs of Boston. There are 14 attributes; the task is to predict the median value of owner occupied houses in $ 1000s. The Parkinsons Telemonitoring Dataset (PARKINSON) [19] is a unique dataset in which about 5875 instances are provided, each with 26 attributes. This dataset has two target attributes which we denote as (h) Parkinson I and (i) Parkinson II; (j) Red Wine and (k) White Wine are two extensive datasets [20] that have 1599 and 4898 data points respectively, each with 12 features. Out of which one, the wine quality score (between 0 and 10) is the target attribute. These datasets give us a desired variety to test empirically our approach. Some of these datasets are straightforward tasks, while some are (such as FIRES) are amongst the hardest regression datasets available.

5.5.3 Methodology

For testing the efficacy of this method, the datasets were subject to a 5 fold cross validation. No feature selection was done on any of the datasets. This is beneficial in these experiments as that makes the prediction task harder. Some of these datasets have a large number of attributes and hence it is clear that not doing feature selection would make the prediction task harder. The only dataset in which a set of features were chosen was the forest fires dataset (f). As given in the description of the dataset in the UCI Machine Learning Repository, we used the last four attributes only. Features in all datasets were also normalized to values between 1 and -1 before applying this technique. This normalization was simply to ensure that none of the features dominated disproportionately in the clustering or regression tasks. While other normalization procedures were tested and some datasets returned better results with specific normalization techniques, we report the results only with one technique applied uniformly across datasets.

Two methodologies for combining predictions were employed in the experiments. Following is one of them:

1. Normalize the dataset such that the features are scaled to the interval [-1, 1]
2. Run k-means clustering on the dataset from 2 to k and assign the value of k for which the dataset hit an empty cluster (K_{empty}) to it.
3. Choose K = Kempty/2 for that dataset. This will signify how many prediction models are to be obtained. Clearly, Kempty/2 prediction models (discussed in section 5.2) are obtained.
4. For each prediction model obtained in step 3 obtain a prediction on the test set.
5. Uniformly average all predictions in step 4 to get a final prediction.

Clearly this method is simplistic in choosing a fixed value of k and not choosing a value empirically. To offset this problem we use a second methodology too. This is described below:

1. Normalize the features between [-1, 1] like in the previous case.

2. Recall that we have to run a 5 fold cross validation on the data. In each of the 5 runs, we have randomly chosen and mutually exclusive train and test sets, such that 4/5th of the data forms the train set and the remaining 1/5th forms the test set.

3. For each of the five folds, run a sub 5 fold cross validation on the training data of that fold (a cross validation within a cross validation i.e consider the 4/5th of the data mentioned in step 2 and divide it further into 5 folds).

    a In each such sub cross validation phase consider a high value of k (such as K_{empty}) and cluster the training data of this sub phase till that value.

    b Train prediction models till this value of k in the sub-phase training set or to a value of k where models can be trained.

    c Average the predictions obtained in this 5 fold sub-cross validation from prediction models 1 to the value in step 3 b above.

    d Find the k in step c averaging to which (from PM_1 to PM_k) gives the least prediction error.

    e Choose this k and return it to the main cross validation loop

4. The k returned in step 3 e. is the value for that fold to which the predictions are to be averaged to. i.e. train prediction models on the train set to this value of k and average the predictions of all of these prediction models.

5. Repeat the process for each fold.

6. Average the errors in the five folds to get a single prediction error (lets call it CVk error)

As discussed, the problem with the first method was that no matter what predictor was used, it always averaged the first $\frac{K_{empty}}{2}$ prediction models. This value did not depend on what predictor was used to make the final prediction. Clearly the choice of predictor would have an impact on how many prediction models are to be averaged (intuitively a weaker predictor would need more prediction models to improve performance while a stronger one would need fewer). The second method alleviates this problem to some degree. It however

suffers from the problem that training in the sub-cross validation phase, by virtue of having lesser points than training in the cross validation phase might return prediction models that are not completely representative of the prediction models returned in the main cross validation.

With these methodologies, experiments were run using three predictors:

1. Linear Regression (without feature selection)
2. Step-Wise Linear Regression
3. Random Forests (for regression)

There were multiple objectives to the experiments conducted using these two methodologies, some of which were:

1. In what kind of datasets is such a method of averaging predictions useful? Are there datasets when it does worse?
2. The choice of averaging $\frac{K_{empty}}{2}$ predictors is an approximation. However it would be interesting to see how the value of number of prediction models that returns the best error value changes depending on the nature of the dataset and the predictor.
3. How much does the utility of clustering depend on the predictor used? What if a strong predictor is used and what if a weak predictor is used?
4. How do these results compare with results when a cross validation within a cross validation is used to choose a value of k till which to average.
5. Does the nature of data normalization alter results?

5.5.4 Results

The three different predictors (Linear Regression, Stepwise Linear Regression and Random Forests) were chosen as representatives for different levels of predictor complexity. A linear regression model might be considered to have high bias with respect to most real world datasets and hence might be thought of as a nave choice in most prediction settings.

Stepwise Linear Regression on the other hand usually does a better job than its forced counterpart. Random Forests, however, represent the state of art in classification and regression. As discussed in the previous section, the experiments were done so as to evaluate how the information exploited by clustering the data aided in a prediction task given a dataset and type of predictor used. Another important question was to understand what kinds of datasets were suitable for such a technique. These observations are discussed in this section. The results with clustering are compared to the condition when no clustering was used (PM_1) using a paired t-test to check for statistical significance. The two methodologies described in the previous section were employed to combine predictions.

The results are organized in three tables (Tables 5.1,5.2 and 5.3) and two figures (Figures 5.4 and 5.5). The prediction results with clustering (employing both the methodologies discussed in the previous section) and without clustering (PM_1) for Linear Regression, Stepwise Linear Regression and Random Forests are tabulated in tables 5.1,5.2 and 5.3 respectively. Figures 5.4 and 5.5 show the error profiles for the different datasets for Stepwise Linear Regression and Random Forests. The error profile shown is essentially the mean absolute error in the prediction obtained by ensembles having 2 to k prediction models (this is represented in the x axis. i.e. a point 5 on the x axis would mean that the bar graph at that point shows the error returned by a model that averaged the first five prediction models). These figures underline the fact that the choice of k returned using the first methodology (of taking Kempty/2) gives quite a sub-optimal choice of k and thus the error value. While the CVk error (given by the second methodology for choosing k empirically) cannot be plotted in such graphs for obvious reasons, the number in the tables show a marked improvement over the first methodology.

In Table 5.1, we immediately notice a couple of broad trends: The CVk error is mostly better than the error obtained by averaging the first Kempty/2 prediction models (as indicated by kmeans I i.e. methodology I in the table). In all but one case it also improves the statistical significance for the improvement over PM_1. The only exception being the red wine dataset where the error returned by the second methodology CVk is a little worse than even $PM1$, however this difference is not statistically significant. Perhaps this improvement across board is not surprising. This is because of the nature of the Linear Regression model,

which is a very simple model that has a high bias w.r.t most real world datasets ($PM1$). So clustering even a little and not to a level that is optimal (clearly methodology I chooses a k that clearly could have been better) improves the prediction accuracy significantly as it boosts the variance. Improving this estimate of how many prediction models should be averaged (by using methodology II, CVk) further improves the prediction accuracy and statistical significance over PM1.

Another observation was that there are datasets which are more clusterable than others with respect to the size of the data matrix (rows by columns or number of data points by number of features). In such datasets, the improvement in prediction errors is not only huge, it is highly statistically significant. The only exception to this generalization is the Red Wine dataset. The red wine dataset is a moderate sized dataset as compared to the others, however clustering does not seem to help in prediction with it. In another dataset, Slump II, the prediction made by using methodology I is better than PM_1 but only marginally statistically significant, this improvement is made statistically significant by using the second methodology. This underlines the ability of CVk to find a better prediction. In conclusion, out of the 11 datasets, an improvement in prediction accuracy was seen in all of them (except the CVk error for Red Wine), this improvement was much more pronounced in the CVk error, both in terms of raw error and statistical significance (over PM_1). This observation points out that the choice of k to average in method I was perhaps suboptimal. This method of choosing k itself might not be optimal but certainly is more principled than the method employed in some experiments.

In the tables, K represents K_{empty}; PM-1 represents the baseline condition, kms represents the case when $K_{empty}/2$ Prediction Models are averaged and the errors are reported in MAE and p(kms) represents the statistical significance with respect to PM-1, CVk represents the error when the number of prediction models to be averaged are machine learning by an internal cross validation and also p(CVk) gives the statistical significance of CVk with PM-1.

Table **??**, which aggregates the results for Stepwise Linear Regression, shows trends similar to Linear Regression. The CVk errors are generally better as compared to the errors returned by the first methodology here as well. The only two exceptions in which

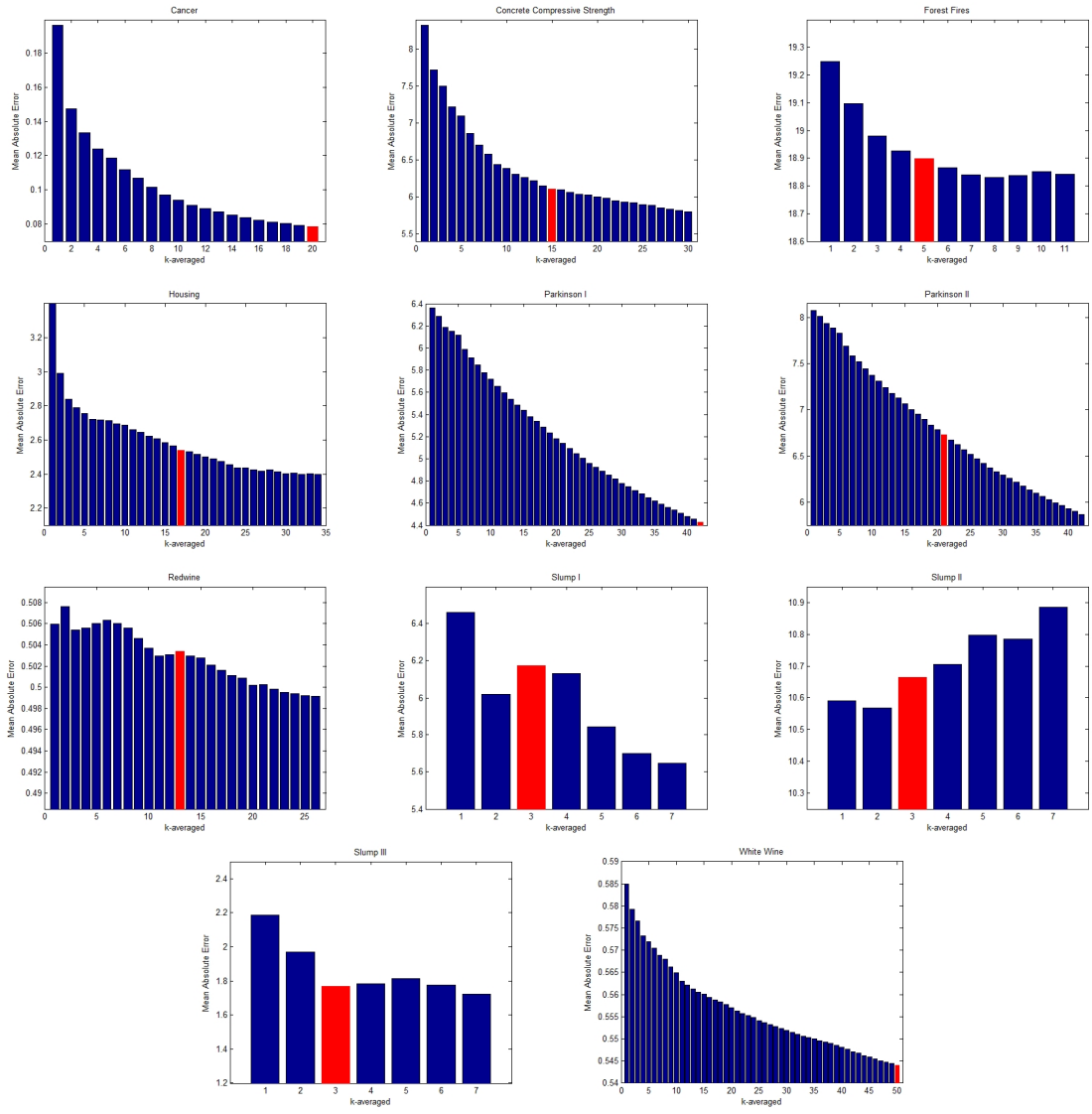

Figure 5.4: The error profiles for all 11 datasets for stepwise linear regression. The x-axis represents the number of prediction models averaged from 1. The bar marked in red indicates the one that has been chosen by the first heuristic as the final prediction. In many cases we notice that this is clearly a sub-optimal choice. The chosen value and the lowest value in the error profile for each dataset should be contrasted with the value of CVk mentioned in the table. Since the number of prediction models to average chosen is different in each fold by the second method, it has not been represented in the graph.

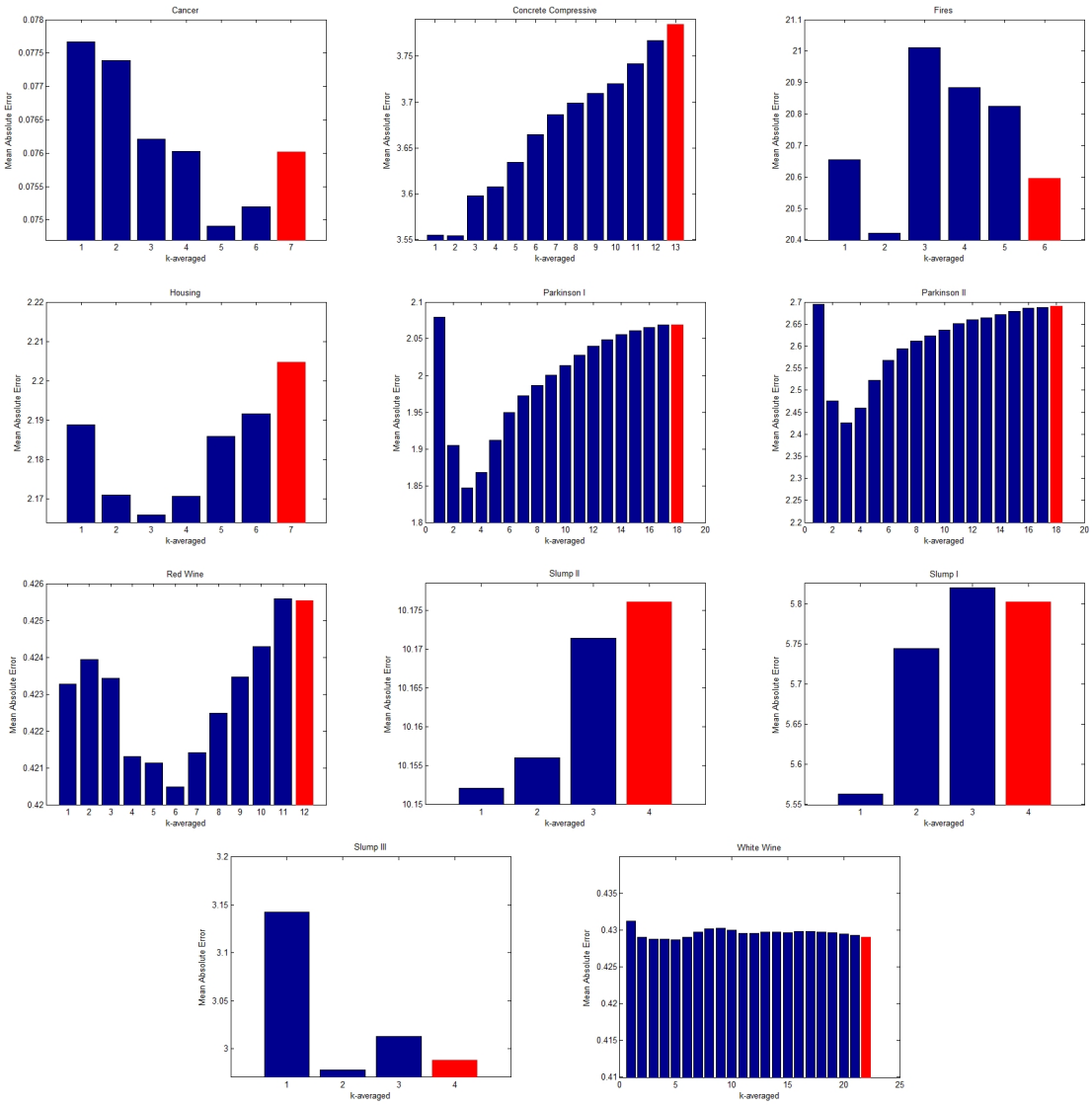

Figure 5.5: The error profiles for all 11 datasets for Random Forests (for regression). The x-axis represents the number of prediction models averaged from 1. The bar marked in red indicates the one that has been chosen by the first heuristic as the final prediction. In many cases we notice that this is clearly a sub-optimal choice. The chosen value and the lowest value in the error profile for each dataset should be contrasted with the value of CVk mentioned in the table. Since the number of prediction models to average chosen is different in each fold by the second method, it has not been represented in the graph.

Table 5.1: Predictions Using Linear Regression and Clustering

Dataset	K	PM-1	kms	p(kms)	CVk	p(CVk)
Parkinson I	42	6.3445	5.0809	$\ll 0.001$	4.3638	$\ll 0.001$
Parkinson II	42	8.0785	6.619	$\ll 0.001$	5.7727	$\ll 0.001$
Red Wine	26	0.5065	0.5048	0.686	0.5073	0.7888
White Wine	52	0.5858	0.5507	$\ll 0.001$	0.5394	$\ll 0.001$
Housing	35	3.4021	2.5904	$\ll 0.001$	2.5883	$\ll 0.001$
Breast Cancer	20	0.1944	0.1136	$\ll 0.001$	0.1139	$\ll 0.001$
Fires	11	19.5009	18.9246	0.0399	18.8739	0.0074
Concrete	30	8.273	5.9688	$\ll 0.001$	5.8316	$\ll 0.001$
Slump I	7	6.2958	5.8312	0.0959	5.7297	0.0155
Slump II	7	11.2	10.5712	0.1843	10.5203	0.1343
Slump III	7	2.0136	1.7655	0.0086	1.7475	0.0063

clustering (by both methods) does not seem to improve upon PM_1 are the SLUMP II and Red Wine datasets (just like for linear regression). As expected, results for stepwise linear regression with clustering give smaller errors as compared to simply linear regression with clustering. Like in the case of Linear Regression, the choice of the number of prediction models to average was suboptimal. This is indicated by the error profile (Figure 5.4) for all 11 datasets when stepwise linear regression was used. The bar in the graph marked in red indicates the error and k picked by using the first heuristic.

These can be contrasted with the CVk errors. The error profiles make a strong case for choosing the number of prediction models to average empirically. Similar error profiles were observed for Linear Regression. Since the two methods are simple in terms of representation power, more clustering seems to help the results (making $K_{empty}/2$ a bad choice), this is especially prominent in datasets that are more clusterable (such Parkinson I and II, White Wine etc), red wine being the only exception. This notion is also reinforced by the error profiles in smaller, noisier datasets such as the SLUMP datasets. By choosing k empirically, we frequently choose a better k for each fold and this is reflected in the results. The results

Table 5.2: Predictions using Stepwise Linear Regression and Clustering

Dataset	K	PM1	kms	p(kms)	CVk	p(CVk)
Parkinson I	42	6.3597	5.1411	$\ll 0.001$	4.429	$\ll 0.001$
Parkinson II	42	8.0798	6.7266	$\ll 0.001$	5.8678	$\ll 0.001$
Red Wine	26	0.5059	0.5034	0.4799	0.5005	0.1555
White Wine	52	0.585	0.5537	$\ll 0.001$	0.544	$\ll 0.001$
Housing	35	3.4252	2.5403	$\ll 0.001$	2.5503	$\ll 0.001$
Breast Cancer	20	0.1962	0.0941	$\ll 0.001$	0.0784	$\ll 0.001$
Fires	11	19.2495	18.8972	0.0215	18.8314	0.0368
Concrete	30	8.3243	6.1101	$\ll 0.001$	5.8025	$\ll 0.001$
Slump I	7	6.4607	6.1754	0.2709	5.7699	0.0255
Slump II	7	10.5918	10.6652	0.8909	10.5639	0.8376
Slump III	7	2.1864	1.7687	$\ll 0.001$	1.788	$\ll 0.001$

Table 5.3: Predictions using Random Forests and Clustering

Dataset	K	PM1	kms	p(kms)	CVk	p(CVk)
Parkinson I	42	2.079	2.0687	0.565	1.8468	$\ll 0.001$
Parkinson II	42	2.6942	2.69	0.8363	2.4264	$\ll 0.001$
Red Wine	26	0.4233	0.4255	0.4239	0.4211	0.326
White Wine	52	0.4312	0.429	0.2313	0.4297	0.3186
Housing	35	2.1888	2.2046	0.7394	2.1764	0.6789
Breast Cancer	20	0.0777	0.076	0.4359	0.076	0.354
Fires	11	20.6546	20.5958	0.8879	20.1743	0.049
Concrete	30	3.555	3.7846	$\ll 0.001$	3.5202	0.0489
Slump I	7	5.5629	5.802	0.1086	5.7336	0.3031
Slump II	7	10.1521	10.1762	0.9537	10.1521	–
Slump III	7	3.1424	2.9877	0.039	3.0051	0.0373

for random forests are the most interesting. This is because it is a strong predictor by itself and hence it is not clear how much help clustering would lend to improve prediction accuracy. It being a strong predictor also in turn means that the earlier heuristic (first methodology) of choosing how many prediction models to average would not work. Choosing this number empirically seems to be a better bet (CVk). This is reflected in the error profiles for all the datasets (Figure 5.5), which are very different from the error profiles of simpler predictors such as Linear Regression and Stepwise Linear Regression. We also notice that the red bar is usually much worse in terms of results. Also, the "correct" choice of how many prediction models to average seems to change from dataset to dataset and there does not seem to be a clear trend unlike for linear regression and stepwise linear regression. Table 5.3 has results that confirm the above speculations. Except in a couple of datasets, the first methodology for combining predictions does not help in improving the prediction accuracy at all. In fact, it goes worse in more than half of the datasets and significantly worse in one dataset. The results for CVk as expected are much better; with the prediction errors improving across datasets and importantly, significantly improving over PM_1 (Random Forest on the entire dataset with no clustering) in 6 datasets. This is an important result. Even in the dataset where the first method returned a significantly worse prediction, the CVk error is better, though not statistically significant. As a remark on implementation, it should be noted that Random Forests could not be trained to a high enough value of k as they need a certain number of points to train properly. And hence much lesser values of k are shown in the bar graphs beyond which training Random Forests was untenable.

One of the advantages of choosing k empirically is illustrated very clearly in the case of SLUMP II. In this dataset, clustering does not seem to give any advantage in prediction at all. The cross validation within cross validation affirms this and returns the best value of k to be 1. This means that we end up with a final prediction which is the same as for PM_1. This example shows that choosing k empirically ensures that we do not force clustering on a dataset where its performance after clustering will actually go worse.

5.6 Future Work

The results obtained in using clustering in conjunction with Linear Regression are not very surprising. The Linear Regression Model is a model with a high bias and is thus not expected to do too well on most real world datasets. Using Linear Regression in conjunction with clustering makes it a much more powerful method as it gives it access to more variance in the data, thus improving the bias-variance trade-off of the complete system. The improvement in prediction accuracy is very significant when it is combined with clustering after doing some feature selection (stepwise regression). In some cases stepwise with clustering returned accuracies comparable to those returned by Random Forests without clustering. Therefore, clustering seems to be giving a cheap method of accessing a lot of information about the data.

It must also be noted that clustering a dataset at a single value of k, with any predictor (only one PM alone making a prediction without any ensembling), rarely improved prediction accuracy in a statistically significant manner compared to the predictor trained without clustering. But, if done at different scales with a prediction obtained at each scale and then combined by means of a nave ensemble, the improvement is very significant as discussed in the previous section. It was also observed that in datasets that were not very clusterable, this technique did not improve upon much.

The experiments done using random forests were more interesting. On smaller datasets the results obtained by using a random forest on the entire dataset and those obtained using the combination of predictions obtained at different scales of clustering did not have a statistically significant difference. This is understandable, as for small datasets clustering at a high value of k might not be able to reveal the true structure for lack of enough data points and might just end up considering sampling noise as structure [91]. This would not contribute much information to aid in the prediction task (might instead reduce the quality). The second and the more important reason would be that for small datasets, techniques such as random forests can exploit enough information such that the generalization error on the test set approaches a limit. Since Random Forest is itself an ensemble method, by means of random sampling of instances and attributes, it already gains a lot of information about the data. Because of this reason, information provided by clustering might not be necessarily

novel. An implicit justification for this is given by the results returned by datasets that are large in size and are much more clusterable. Clustering in such cases is thus more likely to give a novel source of variance that can improve prediction significantly.

An important aspect about the method was choosing which predictions to average. One of the methodologies followed was a nave averaging of the first half of the predictions. This was a suboptimal choice, as there could have been better combinations of the set of predictions that could have been averaged. The choice of using the first half of the predictors was based on the following intuition: Finding the optimal clustering for a dataset might also be considered to be a bias-variance problem. If the number of clusters is too few as compared to the "true" number of clusters, then, most likely, the clustering has a high bias. Inversely, if the number of clusters is too high, we would be over fitting on the data. We selected the first half as a crude tradeoff between this tension. Ideally, the optimal choice of the predictors would be a function of both the clusterabilty of the dataset and the base predictor used. For example, if Linear Regression is used, averaging more predictions could be beneficial. The point of the method discussed in this work was to indicate that clustering gives access to a novel source of information in the data, and thus the aspect of combinations was not optimized. However, a method to pick k empirically was still employed and experimented with. It showed superior results to the earlier nave heuristic. There were some problems with this methodology too. One being that k-means clustering is not a particularly stable clustering. The method utilized to choose a k was based on a cross validation within each fold. Since this stage chose a sub fold that was of a smaller size that the original fold it was not necessarily representative of it. And thus many times it was observed that the error profiles for the sub-cross validation phase were quite different from the error profiles for the main cross validation phase (Figure 5.4 and Figure 5.5 have the error profiles of the main cross validation phase). While this definitely hurt the best choice of k, this experiment establishes how the prediction could be improved. This discussion poses an open model selection problem that could be solved by methods such as those used by the authors in the KDD cup[80] or using averaging as discussed by Caruana[18]. Perhaps the best method for model selection in this case would be the PAC-MDL bound[5].

Another open question is if injecting randomness at various stages can improve the

methods prediction performance. This randomness can be injected in many stages, such as: Currently, we assign each test point to a cluster centroid based on the Euclidean distance and make a prediction for that point. Instead, the point could be assigned in a fuzzy manner, with probabilities of it lying in all clusters. Predictions on each cluster can then be obtained for that point and then weighted averaging can be done to obtain the final prediction. The weights in this case would be the probability that the point belongs to a particular cluster.

Also, for each cluster model we use all features and training examples in the cluster. A random selection with replacement can be made to generate more diversity in the predictors. Preliminary work shows that such an ensemble gives promising performance. Yet another source of variance can be the k-means clustering algorithm itself. The k-means algorithm can give unstable results. In the experiments, we ran k-means 200 times and picked the best clustering. However, each of the converged runs can be used to generate more predictions that can then be combined together. Yet another area that can be worked on to improve the performance of the system can be by using supervised clustering. In our task, we use clustering to boost a prediction performance. However, the clustering is done in a completely unsupervised manner without any regard to the target. The clustering might be completely different if the target is accounted for. A process where the target is taken into consideration while clustering and then models are trained on these clusters would potentially be more beneficial.

Chapter 6

Improving Student Test Score Prediction

This chapter represents work that was published at the Artificial Intelligence in Education 2011 and Educational Data Mining 2011 conferences (see [95], [96]). The strategy used is the same as described in the previous Chapter (i.e. Chapter 5). Infact work outlined in the previous chapter originated from research explicated here which involved mining educational data. The purpose was to be able to predict student test scores better than by existing ideas by employing clustering at different scales.

In Section 6.1 we review in detail the motivation for this work and the relevant literature.

6.1 Motivation and Literature Survey

Feng, Heffernan & Koedinger [41] reported the counter-intuitive result that data from an intelligent tutoring system could better predict state test scores if it considered the extra measures collected while providing the students with feedback and help. These measures included metrics such as number of hints that students needed to solve a problem correctly and the time it took them to solve. That paper[41] was judged as best article of the year at *User Modeling and User-Adapted Interaction* and was cited in the National Educational Technology plan. It mentions a weakness of the paper concerning the fact that time was never held constant. Feng et al. go one step ahead and controlled for time in following

work[42]. In that paper, students did half the number of problems in a dynamic test setting (where help was administered by the tutor) as opposed to the static condition (where students received no help) and reported better predictions on the state test by the dynamic condition, but the difference was not statistically reliable. This present work starts from Feng *et al.* [42] and investigates if the dynamic assessment data can be better utilized to increase prediction accuracy over the static condition. We use a newly introduced method that clusters students, creates a mixture of experts and then ensembles the predictions made by each cluster model (see Chapter 5 for details) to achieve a reliable improvement.

Below we review some literature and the background behind the work mentioned in the paragraph above:

The Bayesian knowledge tracing model[25] and its variants (see [4], [80]) have become the mainstay in the Intelligent Tutoring System (ITS) community to track student knowledge. This knowledge estimate is used for calibrating the amount of training students require for skill mastery. One of the most important aspects of such modeling is to ensure that performance on a tutoring system is transferred to actual post tests. If this is not the case, then that implies over-training within the tutoring system. In fact, it is reasonable to say that one of the most important measures of success of a tutoring system is its ability to predict student performance on a post-test. Since such a transfer is dependent on the quality of assessment, a tension exists between focusing on quality of assessment and quality of student assistance.

Traditionally, performance on a post-test is predicted by using practice tests. Practice tests based on past questions from specific state tests can give a crude estimate of how well the student might perform in the actual state test. Improving this estimate would be highly beneficial for educators and students. For improving such assessment, dynamic assessment [55], [17] has long been advocated as an effective method. Dynamic assessment is an interactive approach to student assessment that is based on how much help a student requires during a practice test. Campione *et al.*[49] compared the traditional testing paradigm, in which the students are not given any help, with a dynamic testing paradigm in which students are given graduated hints for questions that they answer incorrectly. They tried to measure learning gains for both the paradigms from pre-test to post-test and sug-

gested that such dynamic testing could be done effectively with computers. Such assessment makes intuitive sense as standard practice tests simply measure the percent of questions that a student gets correct. This might not give a good estimate of a students knowledge limitations. If a student gets a question wrong, it might not necessarily imply absence of knowledge pertaining to the question. It is likely that the student has some knowledge related to the question but not enough to get it correct. It is thus desirable to have a fine grained measure of the knowledge limitations of the student during assessment. Such a measure might be obtained by monitoring the amount of help the student needs to get to a correct response from an incorrect response. ITS provide the tools for doing dynamic assessment more effectively as they adapt while interacting with individual students and make it easier to provide interventions and measure their effect. Fuchs *et al.* [50] studied dynamic assessment focusing on unique information, such as how responsive a user is to intervention. Feng *et al.*[41], [42] used extensive information collected by the ASSISTments tutor[84] to show that the dynamic assessment gives a relatively better prediction as compared to static assessment. This work effectively showed that dynamic assessment led to better predictions on the post test. This was done by fitting a linear regression model on the dynamic assessment features and making predictions on the MCAS test scores.

They concluded that while dynamic assessment gave good assessment of students, the MCAS predictions made using those features lead to only a marginally statistically significant improvement as compared to the static condition. In this paper we explored the dynamic assessment data to see if we could make significantly better predictions on the MCAS test score. A significant result would further validate the use of ITS as a replacement to static assessments.

6.2 Data and Metrics

The dataset that we considered was the same as used by Feng [41], [42]. It comes from the 2004-05 school year, the first full year when ASSISTments was used in two schools in Massachusetts. ASSISTments is an e-learning and e-assessing research platform [84] developed at Worcester Polytechnic Institute (WPI). Complete data for the 2004-05 year

was obtained for 628 students. The data contained the dynamic interaction measures of the students and the final grades obtained in the state test (MCAS) taken in 2005. The dynamic measures were aggregated as students used the tutor.

The following metrics were developed for dynamic testing by Feng [41], [42] and were used in these experiments. They try to incorporate a variety of features that summarize a students performance in the system. The features were as follows:

1. The students percent correct on the main problems.
2. Number of problems done.
3. Percent correct on the help questions.
4. Average time spent per item.
5. Average number of attempts per item
6. Average numbers of hints per item

Out of these, only the first was as a static metric and was used to predict the MCAS score in the static condition. The other five and a dynamic version of students percent correct on the main problems were used to make predictions in the dynamic condition. The predictions were made on the MCAS scores. The MCAS or the Massachusetts Comprehensive Assessment System is a state administered test. It produces tests for English, Mathematics, Science and Social Studies for grades 3 to 10. The data set we explore is from an 8th grade mathematics test.

6.3 Methodology

The data was split into randomly selected disjoint 70% train and 30% test sets. Feng *et al.* [42] fit a stepwise linear regression model using the dynamic assessment features on the training set to make a prediction on the MCAS scores on the test set. They reported an improvement in prediction accuracy with a marginal statistical significance relative to the predictions made only using data from the static condition. Fitting in a single linear

regression model for the entire student data might be a bad idea for two reasons. First, the relationship between the independent variables (dynamic assessment features) and the dependent variables (MCAS test scores) might not be a linear one. If so, training a linear model would have high bias for the data and no matter how much data is used to train the model, there would always be a high prediction error. The second conceivable source of error is related to the first. A student population would have students with varying knowledge levels, thus requiring different amounts of assistance. Thus it might be a bad idea to fit the entire population in a single model. Students often fall into groups having similar knowledge levels, assistance requirements, etc. It is thus worth attempting to fit different models for different groups of students. It, however, must be noted that while such groups could be identified using clustering, the groups obtained may not be easily interpretable.

The methodology used on using clustering for prediction on this data has been outlined in Chapter 5. Below we summarize the results that were obtained when k-means clustering was used.

6.4 Results using k-means Clustering

The data was first clustered with K taken from 2 to 7. Clustering beyond 7 clusters was problematic as it returned empty clusters. Hence the experiments were restricted to a maximum of K=7 for this dataset. The prediction on the MCAS was made first by using PM1. Then, K was varied from 2 to 7 and a set of six more predictions on the MCAS were obtained (all dynamic features were used). The Mean Absolute Difference (MAD) and the Root Mean Square Errors (RMSE) of the MCAS in the test set were found. This section summarizes these results. It also compares the results with the static condition.

Almost all Prediction Models (Table 6.1) showed a statistically significant improvement in prediction as compared to the static condition demonstrating greater assessment power using the dynamic condition. However, though there is an improvement in the error as compared to the Prediction Model 1, the improvement is not statistically significant, as was previously found to be the case [41].

Table 6.1: Prediction errors by different Prediction Models on the ASSISTments Data with k-means Clustering.

Model	**MAE**	**p-value (with PM_1)**	**p-value (with static)**
Static	10.4900	0.0180	-
PM_1	9.6170	-	0.0180
PM_2	9.3530	0.1255	0.0036
PM_3	9.3522	0.2005	0.0074
PM_4	9.3005	0.1975	0.0062
PM_5	9.3667	0.3375	0.0067
PM_6	9.3518	0.2347	0.0052
PM_7	9.4818	0.6138	0.0134

Averaging Predictions: As reported above the prediction models do not show a statistically significant improvement in prediction accuracy of the MCAS score relative to the PM1. As discussed in Chapter 5, combining them might lead to improved predictions. Indeed, it was observed that averaging across prediction models clearly improves predictions as compared to the prediction models taken alone (Table 6.2). The improvement is not just in the error but also in terms of statistical significance and thus improves the results reported in Table 6.1. These results validate the idea that clustering helps in predictions. These results show how the dynamic assessment prediction accuracy can be further improved.

6.5 Results Using Spectral Clustering

As in the previous section, we report results for the various prediction models and the various averaged models albeit in a graph (Figure 6.1). The spectral clustering ensemble results are not only significant over the static condition ($K = 1$ in Figure 6.1) but also are significant for the k-means generated ensemble beyond $K = 3$ with $p < 0.03$ in each case on a paired t-test. The results for the prediction models are indicated by PMs and the results for the ensembles are represented by the "averaged" results in the graph.

Below we also visualize the clusterings obtained by both k-means and Spectral Clus-

Table 6.2: Prediction errors by different Prediction Models Averaged on the ASSISTments Data with k-means Clustering. The subscripts refer to the models whose predictions were used in averaging.

Model	**MAE**	**p-value (with PM_1)**	**p-value (with static)**
Static	10.4900	0.0180	-
PM_1	9.6170	-	0.0180
PM_{1to4}	9.2375	0.0192	0.0013
PM_{1to5}	9.2286	0.0251	0.0012
PM_{1to6}	9.2268	0.0260	0.0012
PM_{1to7}	9.2398	0.0365	0.0013
PM_{2to4}	9.2604	0.0526	0.0022
PM_{2to5}	9.2406	0.0540	0.0018
PM_{2to6}	9.2348	0.0475	0.0016
PM_{2to7}	9.2507	0.0630	0.0017

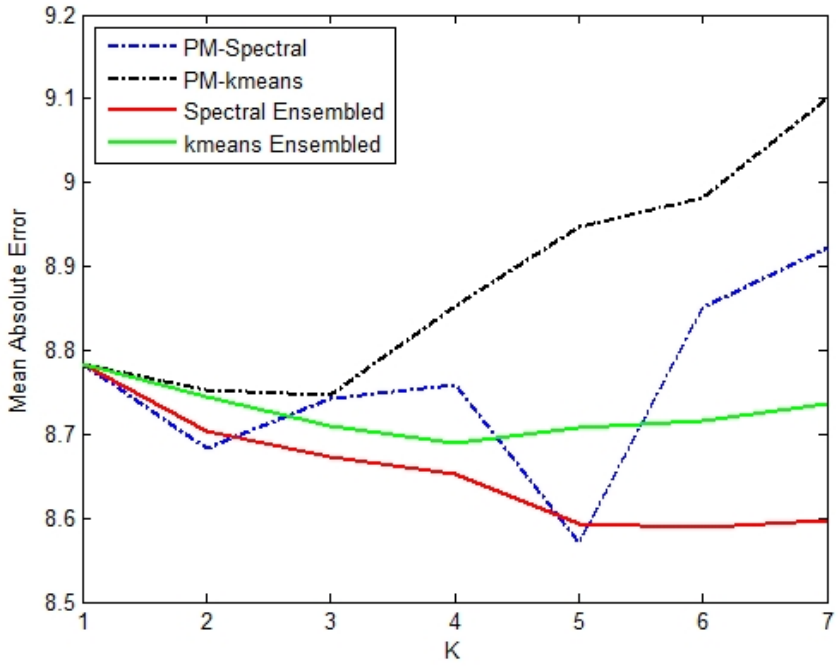

Figure 6.1: Results on the ASSISTments Data using Spectral Clustering

tering on the ASSISTments data. It must be noted that since the data was originally six dimensional, a Multidimensional scaling to three dimensions was done. The visualization was done after this rescaling. We note that for many points, especially those on the fringe, k-means and Spectral Clustering differ in classifying them.

6.6 Contributions and Future Work

The work reported in this chapter makes one clear contribution. This is the first research we know of that clearly demonstrates that not only can an Intelligent Tutoring System allow students to learn while being assessed but also indicates a significant gain in assessment accuracy. This is important, as many classrooms take away time from instruction to administer tests. If we can provide such a technology it would save instruction time and give better assessment and would thus be highly beneficial to students and instructors.

The results for the task of predicting the post test scores have been very encouraging; however there are some areas that need further work and could improve prediction accuracy further. One such area of possible improvement is allowing for fuzzy clustering. To make a prediction, the cluster closest to a test a point was identified and then the expert for that cluster was used to make a prediction on it. In many real world examples, membership of a data point to a particular cluster is a tricky question to answer. A more realistic view is to allow for fuzzy clustering. That is, given a test point, we determine its probability of occurring in each of the clusters. Then, we can obtain predictions by the cluster model /expert for each of the clusters and obtain one prediction for one test point that is a weighted average of the predictions returned by each cluster model (earlier a prediction was made on the test point by only one cluster model), with the weights being the probability that the point lies in that cluster. While fuzzy counterparts to the k-means algorithm such as fuzzy c-means are well known, the idea of doing fuzzy spectral clustering is something to be explored. Clearly, spectral clustering uses k-means at a lower dimensional representation of the Laplacian and fuzzy c-means can be used at this level. However, the effectiveness of doing the same is not known.

Another possible area of improvement is using methods to merge clusters that are

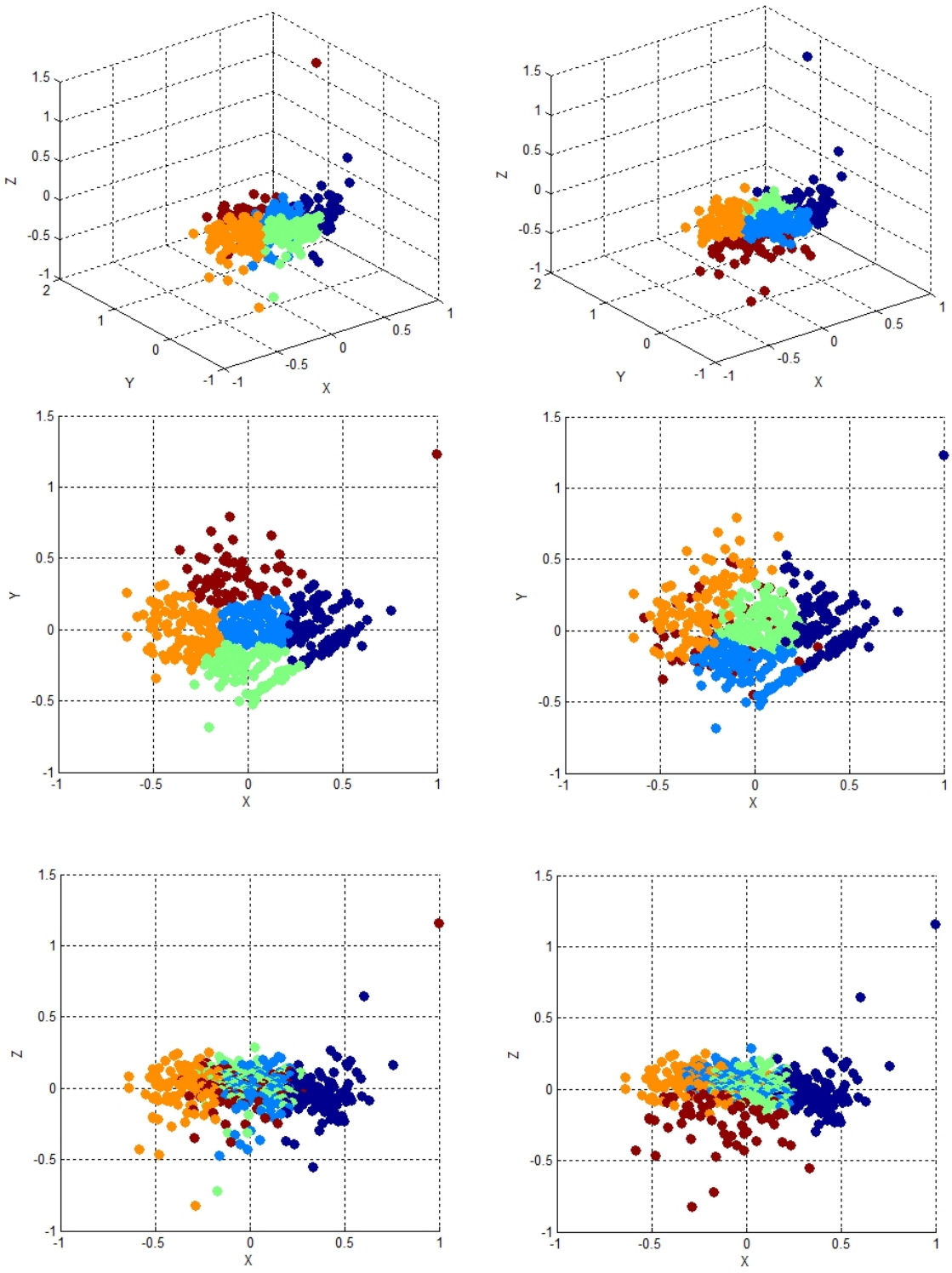

Figure 6.2: Visualization of Clustering on the ASSISTments Data using k-means (L) and Spectral Clustering (R). Top row is a 3-D Visualization while the lower rows are planar views of the same.

sparsely populated[21]. By this method we could improve both the quality of clustering and we hypothesize the prediction accuracies too.

In this work we combine predictions by averaging them. Clearly this is a sub-optimal choice. Ideally, we would want to pick those predictions (made by prediction models) which are good in prediction and have less correlation with each other (are diverse). Since the method used to make the post-test predictions was an ensemble method, it can be used to combine the predictions themselves. Preliminary work utilizing this idea of using clustering to boot-strap the predictions returned by various prediction models has shown promise.

Chapter 7

Improving Knowledge Tracing

Work described in this chapter was published in the 11th International Conference on Intelligent Tutoring Systems [99]. Here we apply the Bagging idea from Chapter 5 for the task of improving the predictive power of Knowledge Tracing, a dominant model in the Student Modeling community. This chapter is organized as follows: Section 7.1 describes some of the background on research on improving Knowledge Tracing and places the idea of clustering students to improve Knowledge Tracing in context, Section

7.1 Motivation and Background on Knowledge Tracing

Corbett and Andersons Bayesian knowledge tracing model [25] and its numerous variants [4], [80] in many ways have become a mainstay in the Intelligent Tutoring Systems community to track student knowledge within learning software. This knowledge estimate is used for calibrating the amount of training students require for skill mastery. With exception of a few techniques such as Performance Factor Analysis[81] for doing this task, most of the methodologies proposed are direct extensions of the classical knowledge tracing (KT) model. Most such extensions differ in what parameters are considered to estimate student knowledge. For example, Baker *et al.*[80] use a separate parameter for the initial knowledge and learning like in the original KT model, but dont estimate Guess and Slip for each skill. Instead they used machine learned models of guess and slip to incorporate contextual information. Similarly, other methods attempt to import different features into the model

to better estimate student knowledge.

Apart from the fact that the most recently proposed models are based on knowledge tracing, it is noteworthy that in most cases such changes have enjoyed mixed success and that the classical KT has remained competitive. This appears to point us to two things: First, the knowledge tracing model for its simplicity is a very strong model for modeling student knowledge. Second, adding new nodes or features is an inherently noisy task unless a very good factor is identified and modeled. In this paper we take a different view of the problem. Instead of working with a KT variant such as in [4] which are known to work better than the classical model, and instead of making changes in what factors are modeled, we look at the input space of the base KT model. A more distributed representation of the input space could potentially be quite useful to boost the accuracy of this based model. Such a representation should also demonstrate where we could have access to novel variance in the input space. One way to learn a richer and more distributed representation of the input space is by clustering. In this work we initially cluster students on some tutor usage features. Based on the groups of students obtained on these features, KT is trained on each such group for each skill. Using a technique developed and described in Chapter 5 we then exploit the information handed over by varying the granularity of the clustering to boost the prediction accuracy.

A recent work that involved clustering of the knowledge tracing space was that by Ritter *et al.*[85]. Their work focused on clustering the parameter space of knowledge tracing (it must be noted however that they cluster skills, we cluster students in our work) and essentially showed that the information compression offered by clustering was enough to significantly reduce the parameter space without compromising the performance of the system. Ritter et al. also mention this as their motivation. It thus cannot be considered an extension to knowledge tracing per se, but it raises important questions about the nature of the parameter space. In Chapter 6 we used clustering to make better out-of-tutor predictions and didnt deal with knowledge tracing at all. We clustered students based on features of tutor usage and then used those features to fit a model to predict performance on a test that students are given at the end of the school year. In our case, we cluster students based on some tutor usage features and then use these distinct clusters to train knowledge tracing

on these clusters. It is an interesting question to see how clusters of students in the feature space correspond to clusters of students in the knowledge tracing space based on the KT parameters. While this is only an auxiliary question that comes up as a result of this work, it points towards some interesting future work wherein the tutor usage features could be improved based on how they influence the clusters in the knowledge tracing space.

7.1.1 Bayesian Knowledge Tracing

The standard Bayesian Knowledge Tracing model [25] models student understanding as a collection of knowledge components, also called skills. The models estimate of whether a student knows a skill is controlled by a set of four parameters which are learned from data for each skill. Based on the intuitive notion that students would learn a skill over time, these set of parameters are fit per skill based on students responses to steps of that skill. The latent in this model represents knowledge of that skill and the two transition probabilities for the latent are prior knowledge and learning rate. Prior knowledge is the estimate that students had the requisite knowledge of the skill prior to their working on the tutor. The probability that students will transition from the unlearned to the learned state while they answer steps is given by the learning rate. The two emission probabilities are given by the parameters guess and slip. The guess parameter represents the probability that the student would answer the question correctly without knowing the requisite skill. The slip parameter on the other hand is the probability that a student answers a question wrong even when she knows the skill. Intuitively skills might be differentiated by the guess and slips associated with them. For example a skill with a high guess rate might be considered to be easier. Going one step ahead it could be said that students might be differentiated into very rough groups based on how they perform on certain skills. It is also likely that students have some characteristics of all such groups in some degree, what differentiates them are some dominant characteristics. Exploiting this idea with KT is also the main logic behind this work which could be done using clustering.

The classical KT model is represented schematically in Figure 7.1. The three observable nodes representing questions in the tutor are placed below the nodes representing the latent knowledge. The figure shows the topology when a student answers three questions, though

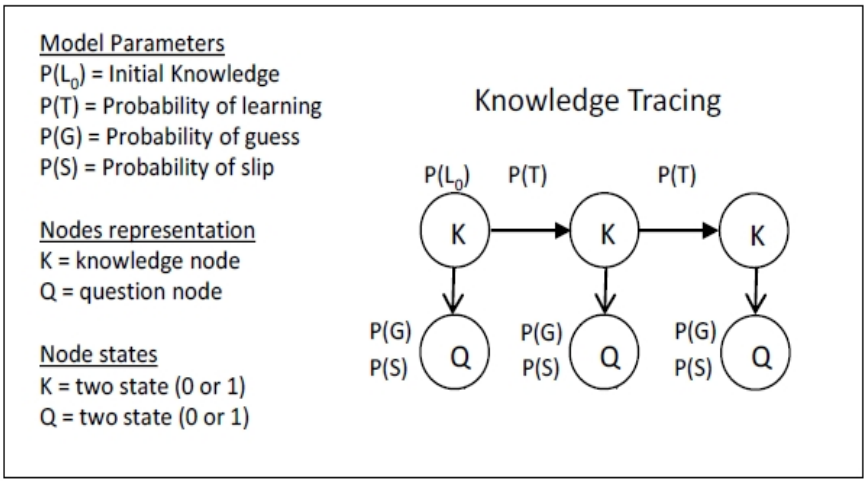

Figure 7.1: The Bayesian Knowledge Tracing Model

this can extend to an arbitrary length based on how many questions a student answers. $P(G)$ and $P(S)$ represent the Guess and Slip parameters respectively. The two parameters that have a bearing on the knowledge node are $P(T)$ and $P(L_0)$, which are the probability of learning and the probability that a student already knows the skill or simply the prior.

7.2 Clustered Knowledge Tracing

Using the methodology outlined in Chapter 5 we use clustering for bagging predictors while using Knowledge Tracing. Using the features from tutor usage we initially employ clustering to find K student groups. Corresponding to each group identified we train KT models separately, thus getting K different models. All of these models together will make one set of predictions on the test data (all of the cluster models together for a given K are called a "prediction model"). The number of clusters K is then varied and the above process is repeated iteratively from K = K -1 to K = 1 (K = 1 corresponds to Knowledge Tracing trained on the entire dataset as is usually done). By this process we get a set of K different predictions. These predictions are then averaged to get a single final prediction.

7.3 Empirical Validation

In this section we present results of experiments to evaluate the performance of "Clustered Knowledge Tracing" as described above and compare it with the baseline. Both k-means and spectral clustering are used. Specifically we used the classical k-means with random initialization and for spectral clustering we used self-tuned spectral clustering[103] with an all connected graph of the data-points.

7.3.1 Datasets

The datasets comes from the 2010 KDD Cup competition on educational data mining. We used the Algebra 2005-2006 and the Bridge to Algebra 2006-2007 datasets. These represent two different Algebra tutoring systems which are part of the Cognitive Tutor family of tutors[63]. The number of students in the Algebra set was 575 with 813,661 total logged responses over 387 skills. There were 1,146 students in the Bridge to Algebra set with 3,656,871 total logged responses over 470 skills. These datasets included skill information for each response and no response was tagged with more than one skill. The Cognitive Tutor divides its online curriculum into units. Skills which appear in different units, even if they have the same name, are considered different skills. Within units there are many problems which students try to solve. Each problem consists of many sub questions called steps. Steps are the level at which the responses in this dataset were logged. Our training and test set is the same as defined by the competition organizers. To create the test set, the organizers randomly picked a problem in each students completed unit and placed into the test set all the students answers to steps in that problem. All the responses to steps prior to that problem within the unit were placed into the training set and everything after that problem within the unit was thrown out. This was departure from the typical evaluation of student models where the data is cross-validated at the student level. We stick to the competitions train and test set format so that comparisons can be made between the error levels we find and the error levels of other published work with this dataset. The various tutor features that were used to cluster the students were: number of skills completed, total number of data-points, user prior, user learn rate, user guess, user slip, number of EM iterations, Log

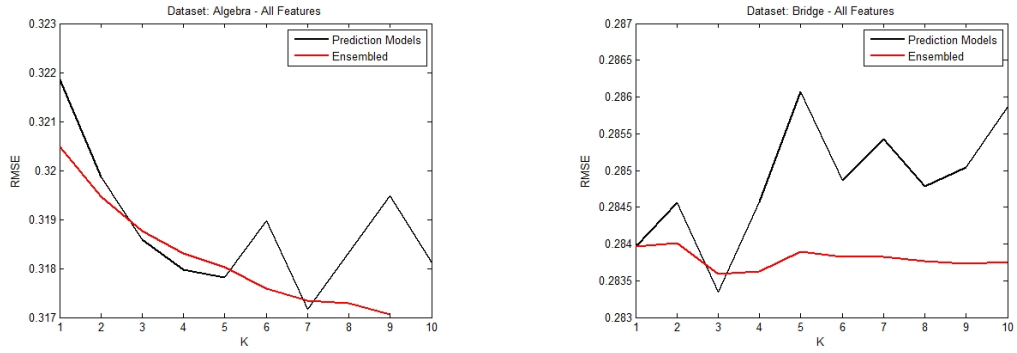

Figure 7.2: Results on the Algebra (L) and the Bridge to Algebra (R) datasets with spectral clustering when all the features are considered. The red line shows the ensembled results after averaging from PM_1 to PM_K while the black one shows the results for each Prediction Model (PM_K).

likelihood improvement, percent correct, average response time. In experiments, students were clustered using all these features and also only using the user tutor features (user prior, user learn rate, user guess, user slip). These user specific KT parameters were generated like in[80] by training a separate KT model per student based on all of that students data in the training set (across all skills).

7.3.2 Results

For both datasets we report results using all the features described above and also by only using the user features. The results while using all features are with both kmeans and spectral clustering, and while using the user features are only by k-means. We report the results for both the individuals prediction models (i.e. the model obtained by training KT on each cluster for a given K i.e. PM_K) and the ensembled results (results obtained by averaging from PM_1 to PM_K). For results we report the RMSE defined per user. The justification to use the RMSE per user is that it equally weighs the benefit to each student without biasing it to students who have contributed more data points.

Initially we tried spectral clustering for the purpose of bootstrapping. This was motivated by the fact that spectral clustering is generally better than k-means clustering. Figure 7.2 shows the results for bagging using spectral clustering considering all the features on

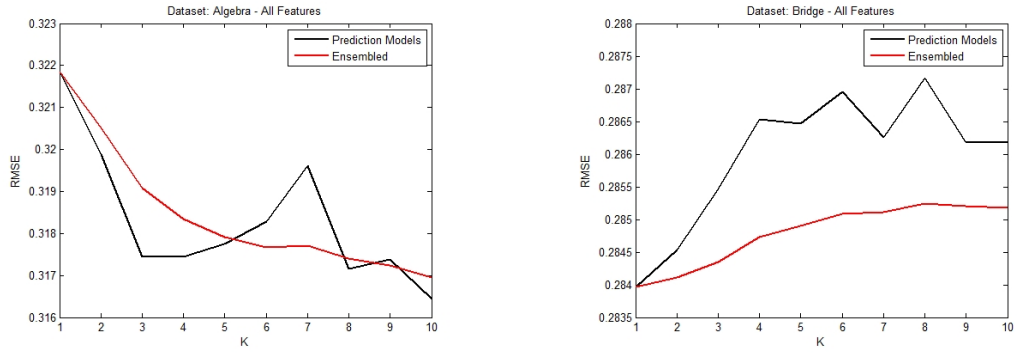

Figure 7.3: Algebra (L) and the Bridge to Algebra (R) with k-means clust. considering all features.

both the datasets. We see the declining trend in error when the results are ensembled and also notice that the individual prediction models dont do too well showing that clustering alone does not help but blending the predictions does. Fig 7.3 indicates that a similar result is repeated in the same scenario with k-means (all features) in the algebra dataset. Such a result is not observed in the bridge dataset however. In fact in the bridge dataset both the various PM_k and the ensembled results do worse than the baseline (which is PM_1 i.e. KT trained on the entire dataset). But in further experiments we see that we can do better even on the bridge dataset if we consider only the user features. For the algebra dataset the base-line (i.e PM_1) RMSE is 0.32185. The best result in the Algebra dataset for spectral (Figure 7.2) is obtained on averaging the first ten prediction models (0.31706). The best result for k-means (Figure 7.3) on this dataset is 0.31696, also after averaging the first ten prediction models. The result is surprising as k-means seems to do better than spectral clustering in this case. The trend however is reversed in the Bridge to algebra data-set, however we still note that the ensemble using spectral clustering does better than the baseline for all the Ks considered in this dataset.

Given that k-means appeared to do well in one dataset and also given its speed, the above procedure was repeated in both the datasets with k-means using only the user specific features. We also cluster to a much higher K and see that the error trend line only decreases as K is increased as is shown in Figure 7.4. Here again, for the Algebra dataset, PM_1 has an RMSE of 0.32185. The best prediction accuracy on averaging is attained at $K = 20$ where

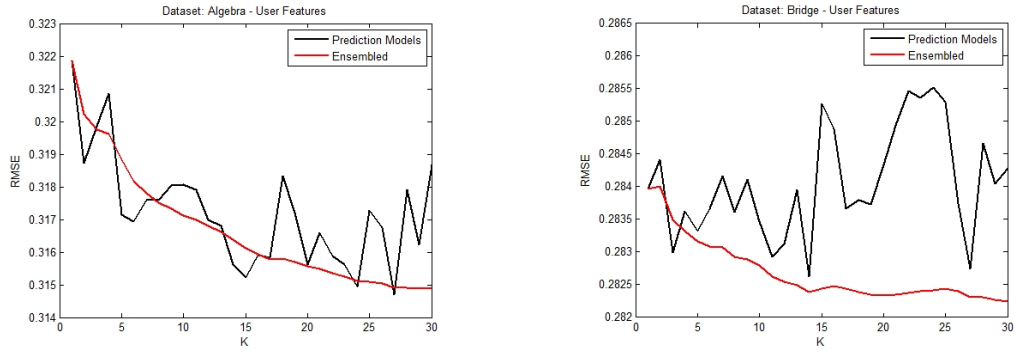

Figure 7.4: Algebra (L) and the Bridge to Algebra (R) with k-means clust. considering user features.

the RMSE is 0.3149. This accuracy is even better as was reported earlier considering both the clustering methods indicating that the user features are much richer for clustering the students. When only the user features are considered a similar error profile is also observed in the bridge to algebra dataset too (PM_1 RMSE = 0.28397 and RMSE of the average from PM_1 to PM_30 is 0.28225). Except for the case when kmeans was run on the bridge to algebra set considering all the features, all the improvements are statistically significant over the baseline ($p < 0.05$). In another experiment in which all the above models are combined, the best accuracy that we obtain for the algebra dataset is 0.31506 and 0.2827 for the bridge to algebra dataset. Like we noted earlier, we report the RMSE per user. However even if we considered the RMSE on the leaderboard we get a statistically significant improvement over the baseline with PM_1 being 0.32408 and the best prediction being 0.32318.

7.4 Discussion

Given the various extensions to the base Knowledge Tracing model that mostly have focused on adding new features to the base knowledge tracing, in this work we took a slightly different view. Instead of trying to model new parameters we try to learn a more distributed representation of the KT input space. We achieve this by using clustering for bootstrapping. In extensive validation we show that our strategy indeed works very well. We report an improvement in prediction accuracy in most cases. We also report that the user features

are much richer for clustering than the features of interaction of a student with a tutor. In fact we believe that this leads to an interesting research problem. Often the interaction of students of an intelligent tutor is measured and these are recorded as features. These features should be such that if students were clustered on this feature space, the clustering should correspond to one on the KT parameter space. If it is not the case then it indicates that the task of feature generation in the tutor is noisy and could be improved in a more principled manner. An improvement in methodology here would be greatly useful in getting features that would be most helpful in making better out of tutor predictions and other tasks too.

An interesting problem would be to consider a case study in which the various clusters as obtained in our methodology are analyzed and an attempt is made to interpret them. While exact meanings are hard to assign to clusters obtained by statistical methods, on the basis of the KT parameters approximate meanings could be assigned and those could be quite useful, especially in making some data driven inferences and pedagogy.

This exploration concerning the KT input space, especially concerning learning a more distributed representation could be quite useful even when used in conjunction with KT variants such as [80] that are known to be stronger predictors than the base KT. This is also a problem that we believe is worth exploring.

Chapter 8

Co-Clustering for Prediction

Learning a more distributed representation of the input feature space is a powerful method to boost the performance of a given predictor. Often this is accomplished by partitioning the data into homogeneous groups by clustering so that separate models could be trained on each cluster. Intuitively each such predictor is a better representative of the members of the given cluster than a predictor trained on the entire data-set. Previous work as outlined in Chapters 5, 6 and 7 has used this basic premise to construct a simple yet strong bagging strategy and demonstrated it in a number of settings. However, such models have one significant drawback: Instances (such as students) are clustered while features (tutor usage features/items) are left alone. One-way clustering by using some objective function measures the degree of homogeneity between data instances. Often it is noticed that features also influence final prediction in homogeneous groups. This indicates a duality in the relationship between clusters of instances and clusters of features. Co-Clustering simultaneously measures the degree of homogeneity in both data instances and features, thus also achieving clustering and dimensionality reduction simultaneously. Instances (Students) and features (question items) could be modelled as a bipartite graph and a simultaneous clustering could be posed as a bipartite graph partitioning problem. In this chapter we integrate the bagging strategy of Chapter 5 with Co-Clustering. While this should be applicable in general, we consider a specific case when the data is of students and tutor usage in particular. We report that such a strategy is very useful and intuitive, even improving upon performance achieved by previous work based on clustering alone. This chapter is

organized as follows: Section 8.1 discusses the general motivation on why co-clustering might be useful. Section 8.2 describes Co-Clustering in general, Section 8.3 describes how predictions could be bagged using Co-Clustering, Section 8.4 describes the experimental results and Section 8.5 talks about the future directions of work.

8.1 Why Co-Cluster?

A significantly large student population would usually have a wide variation in learning rates and knowledge levels. While there are numerous reasons for this diversity, three major reasons are related to: the type of instruction or help they respond best to, the way they are oriented towards learning and their levels of intellectual development [40],[9]. Needless to say, such differences would be reflected in the way students interact with educational software, making educational data quite difficult to mine well. Specifically there are many educational data mining problems where the end goal is to predict the performance of a student on a given in-tutor or out-of-tutor task. In-tutor tasks include predicting the probability that a student will answer an item correctly after attempting a sequence of similar questions whereas out-of-tutor tasks include being to predict student performance in post-tests based on the data from their tutor usage.

The idea that students are quite different makes it apparent that perhaps it is not such a good idea to fit a global prediction model over the entire dataset for making predictions. In spite of the differences between students, educators commonly observe that students actually lie in very rough groups and have similar pedagogical needs. Taking a cue from this intuition, the task of prediction can be improved by clustering students into somewhat homogeneous groups and then training a separate predictor for each group. Such a predictor would obviously be a much better representative of students in that cluster as compared to a predictor which is fit on the entire dataset. For example, it makes sense to have a different model for students roughly classified as fast learners and a different model for slow learners than the same for both. This rather simple strategy of grouping students together and then modeling them separately can lead to improved performance in prediction and perhaps even better interpret-ability.

While the above approach is compelling, there are two major issues with it. Firstly, while it is useful to model students as belonging to different groups, it is also known that such groupings are quite fuzzy and approximate. Students might actually possess different characteristics in varying degrees and what really sets them apart are certain dominant characteristics. For example students classified as fast learners might actually be slow learners in certain skills. A fast learner might also belong to the group of students that are good at recalling information etc. Thus, such complex characteristics can not be possibly modelled by simply clustering students to a certain limit and then training models for each cluster. This "spread" of features in a student across groups also needs to be captured to make a distributed predictive model such as the above more meaningful. Such an issue can be resolved by varying the granularity of the clustering and training separate models each time so that such features can be accounted for. A simple yet quite effective strategy to do so was proposed by the authors and was seen to work quite well both in educational contexts (in-tutor predictions Chapter 7, out-of-tutor predictions: Chapter 6) and more generally in Chapter 5.

The second problem with the above approach is that clustering is implicitly suggested to be one-way i.e only clustering students. But this need not necessarily be the case and only clustering students would consider only half of the story. As an example, consider a matrix in which the rows represent students and the columns represent their responses to certain items. Clearly, clustering students would depend upon their item distributions, implicitly suggesting that for certain students certain items are more important than others. Similarly if items were to be clustered, they would depend on which groups of students get them correct (or incorrect) most frequently. This indicates a duality between these two clusterings, which on simultaneous co-clustering could be very useful in answering many research questions. Co clustering of such a student versus item matrix would pair clusters of student proficiency to clusters of item performance which could be seen as a sort of a subject treatment interaction. This idea could be extended to the more general case of students and features rather than just items. In this chapter we describe this idea of co-clustering students and their tutor interaction features and interleave it with the bagging strategy which was used with clustering. This combined approach is then used to predict

the post-test scores of students.

8.2 Co-Clustering

Clustering algorithms are one-way, i.e. one dimension of the data (say the rows of the data matrix) is clustered based on the similarities measured on the second dimension (say the columns). As pointed out in the previous section it might be desirable, quite frequently, to cluster along both the dimensions simultaneously, exploiting the apparent duality between them. Such simultaneous clustering can often offer interesting insights about the nature of interaction between the clusters at both the dimensions [59]. This utility is fast making co-clustering a fundamental tool for data analysis as is indicated by its widespread use in text and document mining [31], [32]; bioinformatics and gene expression analysis [58], [30]; collaborative filtering [51] and many others practical applications.

While there are now a number of approaches to co-clustering such as based on spectral graph theory [32] and information theory [33], [6], each with its advantages, we consider the approach proposed by Dhillon [32] which formulates the problem of co-clustering as a bipartite graph partitioning problem. We now briefly describe this approach starting with the relevant notation and definitions.

8.2.1 Notation and Definitions

A graph is represented as $G = (\mathcal{V}, E)$ where $\mathcal{V}$ represents the set of vertices and E represents the set of all edge weights E_{ij}, where E_{ij} is the edge weight between vertices $\{i, j\}$.

Definition 8 *The $n \times n$* **Weighted Adjacency Matrix** *of an undirected graph is defined as the matrix $(m_{ij})_{i,j=1,\ldots,n}$. If $m_{ij} = 0$ it implies that vertices v_i and v_j are not connected by an edge. If $m_{i,j} \neq 0$ it implies that the vertices $\{i, j\}$ are connected and $m_{i,j}$ is the corresponding edge weight. Since the graph is undirected, $m_{ij} = m_{ji}$ necessarily.*

Definition 9 *Given the weighted adjacency matrix of a graph and a partition of the vertex set $\mathcal{V}$ into two disjoint subsets $\mathcal{V}_1$ and $\mathcal{V}_2$, the* **cut** *between these two subsets is defined as:*

$$cut(\mathcal{V}_1, \mathcal{V}_2) = \sum_{i \in \mathcal{V}_1, j \in \mathcal{V}_2} M_{ij}$$

An undirected **bipartite graph** is a triple represented by $G = (\mathcal{S}, \mathcal{F}, E)$ where $\mathcal{S}$ and $\mathcal{F}$ are two sets of vertices and E is the set of edges. Since it is a bipartite graph one end of the edges in set E have an endpoint in $\mathcal{S}$ and another in $\mathcal{F}$. In our case the set $\mathcal{S}$ is the set of students while the set $\mathcal{F}$ is the set of features. The set of features could readily be seen as a set of item-responses as well. If $\mathcal{F}$ is the set of items, then an edge between s_i and f_j exists if that item was answered correctly by a student and not otherwise. More generally, if $\mathcal{F}$ is just a set of features, then the edge $\{s_i, f_i\}$ simply represents the value of that feature scaled between 0 and 1 for that student. Given this definition of a Bipartite Graph, now we define the adjacency matrix of the same.

Consider a $m \times n$ dimensional data matrix with students on the rows and the items or features on the columns. Let's suppose this matrix is given by A. Clearly, the adjacency of the bipartite graph is given as:

$$M = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$$

The zeroes on the top-left and the bottom-right sub-matrices signify the absence of connections amongst the elements of $\mathcal{S}$ and $\mathcal{F}$ respectively (since connections in a bipartite graph can only run between $\mathcal{S}$ and $\mathcal{F}$). The matrix M is represented such that taking A at the top right corner and A^T at the bottom left implies that the first m rows of M represent the set of students and the next n rows represent the set of features or items.

Suppose the Bipartite Graphs (whose adjacency matrix is defined above) is partitioned into k clusters $\mathcal{V}_1, \ldots, V_k$. Given this partitioning, a corresponding set of student clusters $\mathcal{S}_1 \ldots \mathcal{S}_k$ and corresponding feature clusters $\mathcal{F}_1 \ldots \mathcal{F}_k$ would also be obtained. It could be intuitively seen that the best possible such set of clustering for all such pairs would be when the sum of all edges which cross between clusters is the minimum possible. As defined by [32] this corresponds to:

$$cut(\mathcal{S}_1 \cup \mathcal{F}_1, \ldots, \mathcal{S}_k \cup \mathcal{F}_k) = min_{\mathcal{V}_1,\ldots,\mathcal{V}_k} cut(\mathcal{V}_1, \ldots, V_k)$$

Where $\mathcal{V}_1, \ldots, \mathcal{V}_k$ represents a k-partitioning of the graph.

The above definition leads us to the Bipartite Graph Partitioning problem:

Definition 10 ***The bipartite graph partitioning problem:*** *Given a graph as defined earlier and subsets of $\mathcal{V}$ which are almost of equal size, say $\mathcal{V}_1^*$ and $\mathcal{V}_2^*$. The required partition is*

$$cut(\mathcal{V}_1^*, \mathcal{V}_2^*) = min_{\mathcal{V}_1, \mathcal{V}_2} cut(\mathcal{V}_1, \mathcal{V}_2)$$

The bipartite graph partitioning problem as defined above is NP-Complete. However, a good relaxation to this problem is given by spectral graph bi-partitioning. This relaxation is achieved via the graph Laplacian. The laplacian L of a graph is a symmetric positive semi-definite matrix such that its un-normalized form is given by $L = D - M$ where D is the degree matrix and M is the adjacency matrix as defined earlier. Note that D is only a diagonal matrix while M is a symmetric matrix with all zeros in the diagonal. Thus, the Laplacian encodes both D and M in it and has many useful properties such as being positive semi-definite, which make it very useful for tasks such as clustering [75]. One property of the Graph Laplacian that make it particularly suitable for clustering are related to the properties of its spectrum. The spectra of the Graph Laplacian unfolds the data manifold to give an lower dimensional embedding which can give "better" clustering results.

Returning to the Bipartite Graph Partitioning Problem, as demonstrated by Dhillon [32] and Mohar [75], the second eigenvector of the generalized eigenvalue problem $Lz = \lambda Dz$ gives a real relaxation to the problem of finding the minimum normalized cut $\mathcal{Q}(\mathcal{V}_1, \mathcal{V}_2)$. The normalized cut is basically a cut that favours finding balanced partitions i.e. if the cut of two different partitions is the same, then the normalized cut is smaller for that partition which is more balanced. Thus it favours partitions that are balanced and have a small cut value. Clearly, the normalized cut is more suitable for tasks such as clustering [88]. Note that this relates to the ideas above relating to the optimal bi-partitionings in the following way: We want balanced clusterings with minimum cut for solving the bipartite graph partitioning problem, which would also be the optimal clustering for us. Thus looking at the Laplacian of the bipartite graph might provide such a clustering.

8.2.2 Spectral Co-Clustering

Given the definitions and notions in the previous section, in this section we state an algorithm [32] for finding the optimal co-clusters $\{\mathcal{S}_1 \cup \mathcal{F}_1\}, \ldots, \{\mathcal{S}_k \cup \mathcal{F}_k\}$ as mentioned above.

For that we define the graph laplacian of a bipartite graph as such an optimal clustering can be found using a laplacian. Using the definition of $L = D - M$ as defined above and also the definitions of D and M. The laplacian may be written as:

$$L = \begin{bmatrix} D_1 & -A \\ -A^T & D_2 \end{bmatrix}$$

and

$$D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}$$

where D_1 and D_2 correspond to the degree matrices of A and A^T respectively.

If the generalized eigenvalue problem $Lz = \lambda Dz$ is written for the above laplacian for a bipartite graph and then rearranged, it has been demonstrated [32] that the resulting equations define the equations for a singular value decomposition of the normalized matrix

$$A_n = D_1^{-1/2} A D_2^{-1/2}$$

Thus instead of finding the second smallest eigenvector corresponding to the second eigenvalue, one could find the left and the right singular values in its place. Finding the right singular value gives a bi-partitioning of students while the left singular value gives a bi-partitioning of the features. These can then be used to find the optimal bi-partition as defined above.

Algorithm

1. *Given the co-occurrence or data matrix scaled to between* 0 *and* 1 *A, form the normalized matrix.*

$$A_n = D_1^{-1/2} A D_2^{-1/2}$$

2. *Compute the second left and right singular vectors for* A_n*, concatenate them together to form a vector* z*.*

3. *Run k-means on this vector to obtain a simultaneous clustering of both the students and the features.*

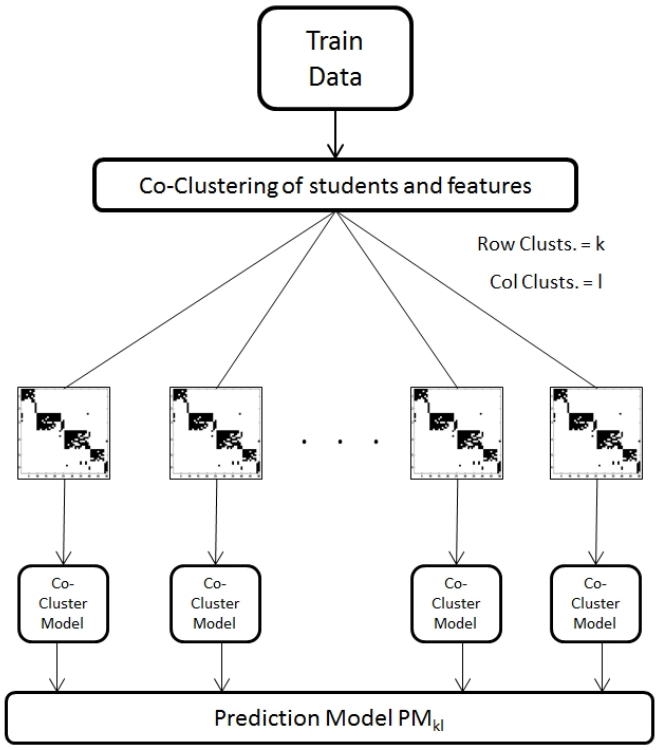

Figure 8.1: Finding a Prediction Model, PM_{kl} with k row clusters and l column clusters

This algorithm can be extended to a multipartition case if instead of finding the second singular values, the first $log_2(k)$ singular vectors are found. The rest of the process remains the same.

Note that this algorithm gives a simultaneous clustering of the rows and the columns and is restricted in the sense that the number of row and columns clusters have to be the same. We modify this by running k-means two times. If the number of row clusters is k and then the number of column vectors is l, then we run k-means on the vector z twice, once to find k clusters and then to find l clusters. The first m elements of the length $m+n$ cluster assignment vector run will then correspond to the row clusters and the last n elements of the cluster assignment vector in the second run will correspond to the column cluster indices.

8.3 Co-Clustering for Bootstrapping

Note that in the bagging idea using clustering described in the previous chapters is only one-way. That is, bootstrapping is done by only changing the data instances available for each cluster model (by changing the number of cluster models itself) but the number of features used in each case is the same. A cluster basically is a bunch of rows in the data matrix with all columns. A co-cluster on the other hand would be a "block" in the data matrix with a sub-set of rows and a sub-set of columns assigned to each "co-cluster". Thus a co-clustering could be thought of as a simultaneous clustering and dimensionality reduction of the data. Note that a clustering is only a special case of co-clustering when the columns are not clustered at all (or have only one column cluster).

Clearly, the above bagging methodology can be suitably modified using co-clustering. For a given number of row clusters k and column clusters l we could have k co-clusters wherein each cluster has only some features assigned to it (note that the definition is symmetric i.e we could think of this as l co-clusters). For each co-cluster we train a separate linear regression model only using the data instances and features assigned to it. We thus obtain k Co-Cluster Models. Like in the above case for clustering, the combination of the k co-cluster models would be considered to be a Prediction Model which makes a single prediction on the test set. We can then vary k from 1 to some value K and l from 1 to some value L. By doing so, we would get a total of $K \times L$ prediction models. We then average a subset of the predictions made by these models to obtain a much stronger prediction.

There are some interesting aspects to such a methodology using co-clustering. For $k = 4$ and $l = 4$, the grid in Figure 8.2 illustrates all the Prediction Models (PM_{kl}) that could be obainted by co-clustering. The Prediction Model $PM_{1,1}$ represented by $(1, 1)$ is simply the case when there is one data cluster and only one feature cluster i.e the original data matrix itself. The prediction model for this case would simply be training a linear regression on the entire dataset, considering all the features. The first column of this grid represents the case when the number of feature clusters is just one, while the number of row clusters are changed. Note that this is simply the methodology described above in Section 8.3 using clustering. The first row of this grid is also equally interesting. In this case the number of row clusters is always one i.e the entire dataset is considered in all co-clusters, while the

column clusters are successively changed. It should be noted that this is a sort of a step-wise regression, where a linear regression is trained on the entire dataset but the number of features that are used to train it are changed (usually reduced as l increases). All the other cases are a cross between these two extreme cases. We see that it seems plausible that a bagging strategy using co-clustering if averaged properly could definitely have more predictive power as it generates diversity by considering a different subset of data instances and features each time, consequently also generating a much larger set of predictions.

8.3.1 Blending Predictions

As mentioned before, the method for combining the predictions returned by the various prediction models is a naive averaging strategy. When the prediction models were generated by clustering (PM_k), we either averaged the first $K/2$ predictions (where K was the maximum number of clusters) (see Chapter 5) or we learned the best number of prediction models that could be averaged by an internal cross-validation (see Chapter 5). The averaging idea is not immediately straightforward when co-clustering is used to generate the prediction models. This is because the prediction models are obtained by changing two parameters. It is also observed that prediction models with a high k or l return poor accuracies, thus it wouldn't be useful to average predictions from all the PM_{k1} models first and then PM_{k2} models and so on (i.e. traversing the grid row-wise or column-wise). Since high values of k and l are counter-productive, we take the order of the prediction models such that the sizes of k and l increase uniformly. This ordering is illustrated by the curve in Figure 8.2. The first half of this reordered set of predictions are then averaged.

8.4 Experimental Validation

In this section we report experimental results for using co-clustering for bagging and compare results with the benchmark (PM_{11}) and clustering alone.

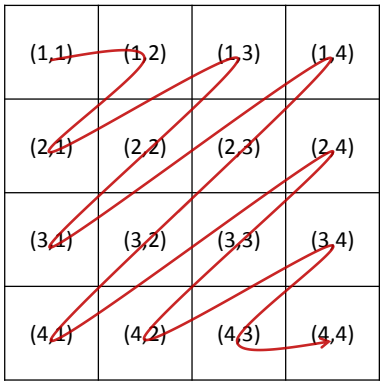

Figure 8.2: Ordering the Co-Cluster Prediction Models, PM_{kl}

8.4.1 Dataset Description and Context

We primarily experiment with two datasets in this study. One of these datasets is the same as that considered in Chapter 6. As mentioned in Chapter 6, this data was collected to study if dynamic assessment, which has long been advocated as an effective method for assessment, was actually better than the traditional static assessment [**?**], [**?**]. Dynamic assessment is an interactive approach to student assessment which is primarily based on how much help a student requires during a practice test. Traditional static testing only takes into account the percentage of questions that the student gets correct. Feng *et al.* [42] showed that features that only recorded how much assistance a student got while interacting with a tutor alone were better predictors of student performance in post-tests held later in the year as compared to how many questions students got correct. This was confirmed in subsequent studies (see Chapter 6). Thus if Co-Clustering is able to improve predictions, then this study could further lend weight to the idea that dynamic testing is indeed better than static testing and that we could further improve upon PM_{11}. It must be noted that PM_{11} would correspond to results reported in [42] which were better than static assessment. PM_{11} basically corresponds to the condition when all the dynamic features are considered and all of the training set is used to train a predictor.

Since one of the datasets is new as compared to the one used earlier in Chapter 6, we describe the features and the prediction task again. The datasets come from the 2004-05 and 2005-06 school years, the first two full years when ASSISTments.org was used in schools

in Massachusetts. ASSISTments is an e-learning tutoring system developed at Worcester Polytechnic Institute which assesses students as it assists. These datasets contain features that measure the interaction of students with the tutor and their actual final grades, which they obtained at the end of the year in the Massachusetts state test (MCAS). There a total number of six features in these datasets **1) DA Original Count** is the number of questions that the students answered with assistance in the dynamic condition. **2) DA Original Percent Correct** is the percent of questions of feature 1 that students get correct . **3) DA Scaffold Percent Correct** is the percentage on tutorial help questions that students get correct. **4) DA Average Time** is the average time that a student spends on a question **5) DA Average Attempt** is the average number of attempts students made per question. **6) DA Average Hints** is the average number of hints that students used. The task is to use these interaction features to predict the MCAS scores that students might get at the end of the school year. The static condition feature is percentage of questions answered correct in static testing. This feature is never used for making predictions for the dynamic condition. The data in the 2004-05 set (ASSISTments 2004-05) is for 628 students, while the 2005-06 data (ASSISTments 2005-06) is for 761 students.

For experimentation we do a five fold cross-validation on the dataset and report results for the base condition (PM_{11}) and the various blended results which were obtained by averaging as discussed in Section 8.3.1. For the sake of comparison we also include results with k-means clustering too. In both cases we consider the ensembled results, with the top K predictions averaged as described in Chapter 5. Following results in Chapter 6 we report results in terms of the mean absolute difference (MAD).

Finally, for pre-processing: As mentioned in Section 8.2, to obtain a bipartite partitioning A must contain values that are either binary or scaled between 0 and 1. Thus, in each fold each feature column is scaled to between 0 and 1 so that A_n could be considered a co-occurrence matrix. This marks a slight difference from earlier papers in which the feature scaling was done so as to map all the data-points to between -1 and 1 by using the mapminmax command of MATLAB. This slight difference might result in a small variation in the results.

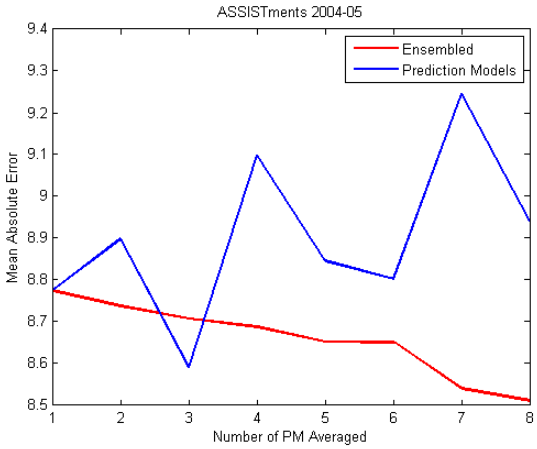

Figure 8.3: Performance on the 2004-05 Set

8.4.2 Experimental Results

We first report results on the ASSISTments 2004-05 dataset. The five fold cross-validated results using co-clustering are reported in Figure 8.3. The number of row clusters (k) and the number of column clusters (l) were restricted to 4 each. This resulted in 16 prediction models. The x-axis in the graph represents the first eight prediction models on doing co-clustering, while the y-axis simply gives the mean absolute error. We observe that the accuracy of co-clustering alone is quite bad (as seen by the blue line) as compared to the baseline (PM_{kl}, which is basically the result for x = 1 in this graph. Note that the baseline is the dynamic condition of Feng [42]). These predictions are those given by the first elements of the ordered set of co-cluster prediction models as defined in Section 8.3.1. However, averaging these prediction models successively gives better and better predictions (as can be seen by the red line).

Similar results were reported in the ASSISTments 2005-06 dataset as shown in Figure 8.4. In this dataset the prediction models are far worse than the ensembled results as compared to the previous dataset. Again, we obtain 16 prediction models after co-clustering and successively average the first eight (the first with second, the first with second and third and so on) after they have been arranged in the way suggested in Section 8.3.1. Again the ensembled results do much better over the baseline (we report exact figures and significance in Tables 1 and 2).

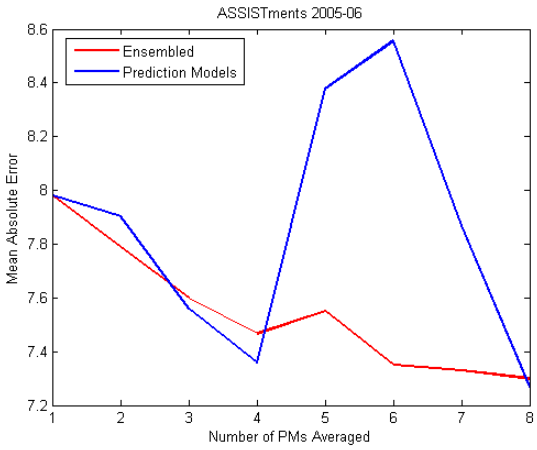

Figure 8.4: Performance on the 2005-06 Set

In Table 8.1 we compare the mean absolute errors when predictions of the first five prediction models are bagged. We report results when the Prediction Models are obtained both by using co-clustering and using k-means clustering on the ASSISTments 2004-05 dataset. The figures in bold indicate statistical significance over the baseline prediction on a paired t-test. Results in Table 8.2 compare the predictions obtained by using co-clustering and k-means for bagging on the ASSISTments 2005-06 dataset.

The results are significantly better over the baseline and also indicate that the dynamic assessment condition returns a much better prediction of student test scores as compared to the static condition. It has already been noted that the static test condition results are significantly worse as compared to even the baseline by [42] and Chapter 6, and thus we don't report results for the static condition.

8.5 Discussion and Future Work

The datasets that were used for the validation of this bagging technique, which is based on co-clustering were not very large and did not have a large number of columns. Thus, these results were initially surprising. One would imagine that in a dataset which has a small number of features, perhaps a feature selection might not be too helpful. However, our experiments show us otherwise. The results that we obtain, while modest improvements

Table 8.1: Comparison of predictions based on k-means and Co-Clustering for the ASSISTments 2004-05 Dataset. Figures in bold indicate significance over the baseline on paired t-test. Numbers are Mean Absolute Errors. Also note that Pred. Model 1 corresponds to the baseline

Pred. Models	Co-Clust	k-means
1	8.7741	8.7741
2	8.7379	8.7518
3	**8.7087**	**8.6725**
4	**8.6879**	8.7153
5	**8.6574**	**8.7100**

show that this technique though simple can give access to a novel source of variance in the data. It can potentially also have some nice properties in terms of returning simpler and more interpretable groups. For example, it was earlier pointed out that one row of the prediction models were actually nearly like a linear regression model in which the features are successively eliminated. At the same time it was observed that one column of the prediction models were actually just the various prediction models that we obtained on clustering alone as reported in some previous work. It would be interesting to see how the Co-Clusters (which are basically blocks in the data matrix) on a student-item dataset would pair clusters of student proficiency to clusters of item performance which could be seen as a sort of a subject treatment interaction.

In the literature, it has been said that the real strength of co-clustering is with binary valued data, co-occurrence tables and basically in scenarios which involve collaborative filtering. Hence, datasets which are basically a student by item matrix would be an ideal candidate for trying out this technique. In the KDD Cup 2010 Töscher and Jahrer modelled student response data as a collaborative filtering task and used matrix factorization techniques for the same. Given the connections of co-clustering with matrix factorization, it is worth investigating how useful it could be in such a setting.

In Chapter 6, we clustered students based on tutor interaction features and then trained separate Knowledge Tracing models for students based on the cluster they were in. This was

Table 8.2: Comparison of predictions based on k-means and Co-Clustering for the ASSISTments 2005-06 Dataset. Figures in bold indicate significance over the baseline on paired t-test. Numbers are Mean Absolute Errors. Also note that Pred. Model 1 corresponds to the baseline

Pred. Models	Co-Clust	k-means
1	7.9822	7.9822
2	**7.7716**	**7.8185**
3	**7.5990**	**7.8034**
4	**7.4680**	**7.7815**
5	**7.5503**	**7.6487**

done so because it was not possible to cluster the item sequences directly and an indirect approach had to be taken. This co-clustering technique seems to give an alternative by which such matrices might be clustered more readily without the need to cluster the tutor interaction features.

In summary, in this work we propose a bagging technique that uses co-clustering and demonstrate that it's performance is better than that obtained by bagging using clustering. We also suggest that it is most suitable for datasets which are like co-occurrence tables and believe that it would be a good direction for future work since such student-item datasets are usually of this form.

Part IV

Concluding Remarks

Chapter 9

Conclusions

While each chapter aptly summarizes the contributions made. Here we briefly review all of the contributions again. First we looked at the Spectral Clustering pipeline and while working towards a solution for easing the associated computational bottleneck, we introduced a new clustering algorithm that eases the second stage (finding the Eigendecomposition, the first being finding the similarity graph) of the pipeline. This algorithm harnesses the power of the Szemerédi Regularity Lemma, a fundamental result from Extremal Graph Theory. This section of the thesis also presents one of the first direct attempts to use the Regularity Lemma for more practical applications. We do so by modifying the constructive versions of the Regularity Lemma so that they could be made more useful for graphs of much smaller size than what the Regularity Lemma works for as stated in the Original form. We also propose a two-stage strategy to use to the Regularity Lemma for clustering and compare the results of such a strategy with standard clustering methods such as Spectral Clustering and k-means clustering. The results are quite encouraging.

In a second contribution we propose a simple yet effective bagging strategy that uses clustering to generate an ensemble. This *Mixture of Experts* like strategy uses clustering at various scales to generate a set of diverse predictions that are then naively averaged to get a final prediction. The efficacy of this meta-algorithm was tested on a number of benchmark datasets and also extended to using Co-Clustering. An auxiliary contribution of using this strategy is that it was used to extend the power of a number of techniques from the educational domain, such as Knowledge Tracing.

9.1 Future Work

Each chapter while summarizing contributions also summarized some possible avenues for further work. In this section we briefly review some major directions of future work and also describe them in more detail (for every other, we direct the reader to each chapter).

1. *Extending the Regularity Clustering Framework to Hypergraphs:* As described earlier, one of the most attractive notions of pairwise clustering methods is that they give a more "global" view of the data. Given enough number of data-points we could at least approximately get an idea about the geometry of the data, thus significantly improving it's performance over traditional centroid based methods which are more "local". As pointed out by Fowkles *et al.* [46],when seen through the lens of computer vision this makes such "global" clustering methods (for segmentation) closer to the original Gestaltian views on form and perception that for a human an image is much more than a mere collection of objects. However, while pairwise affinities capture a more global view of the data, it is not necessary that the relationship between data-points in most domains has to be dyadic and thus restricting it to being dyadic might lead to loss of information. Indeed, it might be the case that the relationship between data-points is triadic, tetradic or even higher. Thus, this natural extension has led to work on clustering methods for such problems, which can be naturally formulated as a hypergraph partitioning problem [1], [104], [16]. There are a number of important results that extend the regularity lemma to hypergraphs [86], [87],[54],[23]. It is thus natural that our methodology could be extended to hypergraphs and then used for hypergraph clustering. This seems to be a particularly promising problem.

   While the above algorithms give a way to construct a regular partition for hypergraphs, some of them are too technical. If we consider the Frieze-Kannan algorithm for finding Regular Partitions based on singular values [48], then it gives a possible idea on how to develop a hypergraph regular partition algorithm that is easier to implement and equally powerful. Thus, our first interest is to investigate whether this simple algorithm can be extended to hypergraphs or not. The singular values for Higher Order SVD or Tensors are not well defined for many higher orders and ranks. With the

exception of the first rank for all orders and the second order (a matrix). Incidentally this is all that is needed to relate regularity to a singular value in the Frieze-Kannan version. A sketch of a plausible approach could be described as:

The goal would be to extend Frieze-Kannan from a simple two-dimensional regular partition algorithm to hypergraphs as stated. Such an extension would solve the following problem: Given a large dense r-uniform hypergraph H, a small positive number ε, we want to have an algorithm to construct a weak ε-regular partition for this hypergraph. The term weak here implies that it works only for r-uniform hypergraphs. For doing so, we would need a lemma of the following kind.

Statement 1: Let W be an r-dimensional matrix on $V_1 \times V_2 \cdots \times V_r, |V_r| = p_i$, $W(i, j \ldots, k) \leq 1$, γ is a small positive real number. (a) If there exists $A_i \subseteq V_i$, such that $|A_i| \geq \gamma p_i$, and $|W(A_1, A_2, \ldots, A_r)| \geq \gamma |A_1||A_2| \ldots |A_r|$ then $\sigma_{max}(W) \geq \gamma^{r+1}\sqrt{p_1 p_2 \ldots p_r}$. (b) If $\sigma_{max}(W) \geq \gamma^{r+1}\sqrt{p_1 p_2 \ldots p_r}$, then there exists $A_i \subseteq V_i$, such that $|A_i| \geq \gamma' p_i$ and $W|(A_1, A_2, \ldots A_r)| \geq \gamma' |A_1||A_2| \ldots |A_r|$ where $\gamma' = c\gamma^{r+1}$ and c is a constant. Furthermore $A_1, A_2, \ldots, A_r$ can be constructed in polynomial time.

It would be interesting to look into the possibility of having such a theorem that could pave way to a simple hypergraph partition algorithm. If possible, it could make it quite easy to construct a hypergraph spectral clustering algorithm similar to one that we introduced in this thesis by using this new algorithm on singular values.

2. *Applying the Regularity Clustering Framework for Image Segmentation:* Another fruitful line of work would be to employ the Regularity Clustering framework for image segmetation. One possible way is to use the same two phase strategy as described earlier, where in each pixel in an image could become a node in the graph and the edges could represent the similarity between two pixels. This similarity measure could be based on brightness proximity and might be given as

$$w_{ij} = exp\left(\frac{-(I(i)-I(j))^2}{2\sigma^2}\right)$$

where i, j are pixels and $I(i)$ the corresponding pixel intensity of pixel i.

Yet another way of using the Regularity framework for segmentation could be using it to make an initial guess for a framework such as the Normalized Cuts due to Maji

et al. [72]. Biased Normalized Cuts gives a way to find normalized cuts for an image given there is a prior, this prior could be supplied using Regularity Clustering coupled with some contour detection engine.

3. **Using Hypergraphs for Bagging:** In this thesis we introduced a simple method to use clustering to generate an ensemble. The procedure was to vary the granularity of the clustering, obtain a set of prediction and average them to some level of granularity, say k. It might be noted that this strategy is quite simple and might mean loss of valuable information in some higher level of granularity that was not used for averaging. To fix this - we propose the following methodology: We could find a clustering of the data at a range of k's as before, however, now we could represent each clustering appropriately as an edge in an hypergraph. This hypergraph could then clustered to some value and then predictions made on this basis could be averaged. The intuition is that such a hypergraph would have a more global notion of the clusterings and perhaps it would not throw away information as we do in our present strategy.

Part V

Bibliography

Bibliography

[1] S. Agarwal, L. Zelnik-Manor J. Lim, P. Perona, D. Kriegman, and S. Belongie. Beyond pairwise clustering. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2005.

[2] N. Alon, R. A. Duke, H. Lefmann, V. Rödl, R. Yuster, The Algorithmic Aspects of the Regularity Lemma. *Journal of Algorithms*, 16, (1994), pp. 80-109.

[3] F.R. Bach, M.I. Jordan, Learning spectral clustering, Tech. Rep. UCB/CSD-03-1249, Computer Science Division, University of California, Berkeley, CA, USA, Jun. 2003.

[4] R. S. J. d. Baker, A. T. Corbett, V. Aleven, More Accurate Student Modeling Through Contextual Estimation of Guess and Slip Probabilities in Bayesian Knowledge Tracing. In *The Proceedings of the 14th International Conference on Artificial Intelligence in Education*. Brightion, UK, pp. 531-538, 2008.

[5] A. Banerjee, and J. Langford, An Objective Evaluation Criterion for Clustering, 2004, KDD pp 515-520.

[6] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Merugu and D. S. Modha, A Generalized Maximum Entropy Approach to Bregman Co-clustering and Matrix Approximation, In *Journal of Machine Learning Research*, 8, pp. 1919-1198, 2007.

[7] A. - L. Barabási and R. Albert, Emergence of Scaling in Random Networks, 1999, Science 286, 509.

[8] M. A. Belabbas and P. J. Wolfe, Spectral methods in Machine learning: New strategies for very large datasets, *Proceedings of the National Academy of Sciences of the USA*, 106, (2009), pp. 369-374.

[9] J. D. Bransford, A. L. Brown and R. Cocking eds., How People Learn: Brain, Mind, Experience, and School, Washington, D.C.: National Academy Press, 2000.

[10] A. Blum, and J. Langord, PAC-MDL bounds, In *Proceedings of the 16th Annual Conference on Computational Learning Theory*, COLT 2003.

[11] A. Blumer, A. Ehrenfeucth, D. Haussler, and M. K. Warmuth, Occams Razor, In *Information Processing Letters*, 1987, 24:377-380.

[12] M. Belkin and P. Niyogi. Semisupervised learning on Riemannian manifolds. Machine Learning, 56, 209239, 2004.

[13] Y. Bengio, Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1127, Now Publishers, 2009.

[14] Breiman, L., Random Forests, Machine Learning 45(1), pp. 53, 2001.

[15] G. Brown, J. L. Wyatt, P. Tino, Managing Diversity in Regression Ensembles, In *Journal of Machine Learning Research*, 2005, Vol 6, pp. 1621-1650.

[16] S. Bulo and M. Pelillo, A game-theoretic approach to hypergraph clustering, In *Advances in Neural Information Processing Systems*, 2009.

[17] J. C. Campione and A. L. Brown, Dynamic Assessment: One Approach and some Initial Data. Technical Report. No. 361. Cambridge, MA. Illinois University, Urbana, Center for the Study of Reading. ED 269735, 1985.

[18] R. Caruana and A. Niculescu-Mizil, Ensemble Selection from Libraries of Models, In *The Proceedings of the 21st International Conference on Machine Learning (ICML04)*, 2004.

[19] D. Chakrabarti, C. Faloutsos, Graph mining: Laws, generators, and algorithms, ACM Computing Surveys 38 (1), Article No. 2, 2006.

[20] K. Chaudhari, S. Dasgupta, A. Vattani, Learning Mixture of Gaussians using the k-means Algorithm, In CoRR vol.abs/0912.0086, http://arxiv.org/abs/0912.0086, 2009

[21] D. Cheng, R. Kannan, S. Vempala, G. Wang, A Divide and Merge Methodology for Clustering. In *The Journal of the ACM*, Vol V, 2006

[22] F. R. K. Chung, Spectral Graph Theory, *Regional Conference Series in Mathematics*, 1997.

[23] F. Chung, Regularity lemmas for hypergraphs and quasi-randomness, In *Random Struct. Alg.* , 2 (1991), pp. 241252.

[24] F. R. K. Chung and Z. Wenbo, Pagerank and Random Walks on Graphs., *In Proceedings of Fete of Combinatorics and Computer Science Conference in honor of Laci Lovász*, Keszthely, Hungary, 2008.

[25] A. T. Corbett and J. R. Anderson, Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. In *User Modeling and User Adapted Interaction*, 4, pp. 253-278, 1995.

[26] T. Cover and J. Thomas, Elements of Information Theory. New York: Wiley, 1991.

[27] V. Csiszár, L. Rejtö and G. Tusnády, Statistical Inference on Random Structures, In *Horizon of Combinatorics*, eds. E. Györi et al., pp. 3767, 2008.

[28] S. Dasgupta, Learning Mixtures of Gaussians, In *The 40th Annual IEEE Symposium on Foundations of Computer Science* ,1999, pp 349-358.

[29] M. Deodhar, and J. Ghosh, A Framework for Simultaneous Co-clustering and Learning from Complex Data, In *KDD 2007*, pp. 250-259, 2007.

[30] P. D'Haeseleer, S. Liang and R. Somoyogi, Genetic Network Inference: From co-expression clustering to reverse engineering, In *Bioinformatics*, 16, 2000, pp. 707-726.

[31] I. S. Dhillon, S. Mallela and R. Kumar, A divisive Information-Theoretic Feature Clustering Algorithm for Text Classication. In *Journal of Machine Learning Research*, 3(4): pp. 1265-1287, 2003.

[32] I. S. Dhillon, Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining*, pp. 269-274, 2001.

[33] I. S. Dhillon, S. Mallela, and D. Modha, Information-theoretic co-clustering. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 89-98, 2003.

[34] C. Ding, X. H, H. Zha, M. Gu, M and H. Simon, A min-max cut algorithm for graph partitioning and data clustering. *In Proceedings of the first IEEE International Conference on Data Mining (ICDM)* pp. 107-114. Washington, DC, USA: IEEE Computer Society, 2001.

[35] T. G. Dietterich, Ensemble Methods in Machine Learning, In *First International workshop on Multiple Classifier Systems*. Kittler J., and Roli., F. (Eds.), Lecture Notes in Computer Science, New York, Springer Varlag, 2000, pp. 1-15.

[36] T. G. Dietterirch, An Experimental Comparison of three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization, In *Machine Learning*, Kluwer Academic Publishers, 2000, Vol. 40, pp. 139-157.

[37] W. E. Donath and A. J. Hofman, Lower Bounds for the Partitioning of Graphs, In *IBM J. Res. Develop.*, 17, pp.420-425, 1973.

[38] F. J. Dyson, Living Through Four Revolutions, Lecture at the Perimeter Institute. Online at `http://www.perimeterinstitute.ca/index.php?option=com_content&task=view&id=551&Itemid=568&lecture_id=10397`

[39] P. Erdös, P. Turán, On some sequences of integers, *J. London Math. Soc*, 11, (1936), pp. 261-264.

[40] R. M. Felder and R. Brent, Understanding Student Differences, In *Journal of Engineering Education*, Vol. 94, No. 1, 2005, pp. 57-72.

[41] M. Feng, N. T. Heffernan and K. R. Koedinger, Addressing the Assessment Challenge in an Online System that Tutors as it Assesses. In *User Modeling and User-Adapted Interaction: The Journal of Personalization Research.* 19(3). 2009.

[42] M. Feng and N. T. Heffernan, Can We Get Better Assessment From A Tutoring System Compared to Traditional Paper Testing? Can We Have Our Cake (better assessment) and Eat it too (student learning during the test). In *The Proceedings of the 3rd International Conference on Educational Data Mining*, pp. 41-50, 2010.

[43] M. Fielder, Algebraic Connectivity of Graphs, In *Czechoslovak Math. J.* , 23, pp. 298-305.

[44] E. Fischer, A. Matsliah, A. Shapira, Approximate hypergraph partitioning and applications, In *Proceedings of the 48th annual IEEE Symposium on Foundations of Computer Science (FOCS)* 2007, pp. 579-589.

[45] S. Floyd, and M. K. Warmuth, Sample Compression, Learnability and the Vapnik-Chervonenkis Dimension, In *Machine Learning*, 1995, pp. 1-36.

[46] C. Fowlkes, S. Belongie, F. Chung and J. Malik, Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, (2004), pp. 214-225.

[47] A. Frank and A. Asuncion, UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science, 2010.

[48] A. M. Frieze, R. Kannan, A simple algorithm for constructing Szemerédi's regularity partition. *Electron. J. Comb*, 6, (1999).

[49] L. S. Fuchs, D. L. Compton, D. Fuchs, K. N. Hollenbeck, C. F. Craddock and C. L.Hamlett, Dynamic Assessment of Algebraic Learning in Predicting Third Graders of Mathematical Problem Solving. In *Journal of Educational Psychology*, 100(4), pp. 829-850, 2008.

[50] D. Fuchs, L. S. Fuchs, D. L. Compton, B. Bouton, E. Caffrey, and L. Hill, Dynamic Assessment as Responsiveness to Intervention. Teaching Exceptional Children. 39(5), pp. 58-63, 2007.

[51] T. George and S. Merugu, A Scalable Collaborative Filtering Framework based on Co-Clustering. In *Proceedings of the IEEE Conference on Data Mining*, pp. 625-628, 2005.

[52] W. T. Gowers, Lower bounds of tower type for Szemerédi's uniformly lemma. Geom. Funct. Anal 7, (1997), pp. 322-337.

[53] W. T. Gowers, The Work of Endre Szemerédi. Exposition on Endre Szemerédi's work for the Abel Prize 2012, Online at `http://www.abelprize.no/c54147/binfil/download.php?tid=54060`

[54] W. T. Gowers, Hypergraph regularity and the multidimensional Szemerédi theorem, In *Annals of Mathematics*, (2) 166 (2007), no. 3, pp. 897-946.

[55] E. L. Grigerenko and R. J. Steinberg, Dynamic Testing. In *Psychological Bulletin*, 124, pp. 75-111, 1998.

[56] A. Gyárfás, M. Ruszinkó, G.N. Sárközy, E. Szemerédi, An improved bound for the monochromatic cycle partition number, *Journal of Combinatorial Theory, Ser. B* 96, (2006), pp. 855-873.

[57] A. Gyárfás, M. Ruszinkó, G. Sárközy, E. Szemerédi, Three-color Ramsey numbers for paths, *Combinatorica*, 27(1), (2007), 35-69.

[58] D. Hanisch, A. Zein, R. Zimmer, T. Lengauer, Co-clustering of Biological Networks and Gene Expression Data, In *Bioinformatics* 18 (Suppl.), 2002, pp. 145-154.

[59] J. A. Hartigan, Direct Clustering of a Data Matrix, In *Journal of the American Statistical Association*, 67(337): pp. 123-129, 1972.

[60] E. Hartuv, R. Shamir, A clustering algorithm based on graph connectivity, Information Processing Letters 76 (4-6), pp. 175-181, 2000. 175181

[61] R. A. Jacbos, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, Adaptive Mixture of Experts, In *Neural Computation*, vol. 3, pp. 79-87, 1991

[62] J. Kleinberg. Authoritative Sources in a Hyperlinked Environment. In *Journal of the ACM* 46 (5): 604632, 1999.

[63] K. R. Koedinger, A. T. Corbett, Cognitive Tutors: Technology bringing learning Science to the Classroom. In K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (pp. 61-78). New York: Cambridge University Press, 2006.

[64] Y. Kohayakawa, V. Rödl, L. Thoma, An optimal algorithm for checking regularity. *SIAM J. Comput*, 32(5), (2003), pp. 1210-1235.

[65] S. Kok, and P. Domingos, Statistical Predicate Invention. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, pp. 433-440, ACM Press, 2007.

[66] D. Koller and N. Friedman, Probabilistic Graphical Models: Principles and Techniques. MIT Press, 2009.

[67] J. Komlós, G. N. Sárközy and E. Szemerédi, Blow-up Lemma, *Combinatorica* 17(1), (1997), pp. 109-123.

[68] J. Komlós, G. N. Sárközy and E. Szemerédi, An algorithmic version of the Blow-up Lemma, Random Structures and Algorithms 12, (1998), pp. 297-312.

[69] J. Komlós, A. Shokoufandeh, M. Simonovits, and E. Szemerédi, The Regularity Lemma and Its Applications in Graph Theory. *Theoretical Aspects of Computer Science*, LNCS 2292, (2002), pp. 84-112.

[70] H. W. Kuhn, The Hungarian method for the Assignment Problem, *Naval Research Logistics*, 52(1), 2005. Originally appeared in Naval Research Logistics Quarterly, 2, 1955, pp. 83-97.

[71] U. Luxburg, A Tutorial on Spectral Clustering, In *Statistics and Computing*, Kluwer Academic Publishers, Hingham, MA, USA. Vol 17, Issue 4, 2007.

[72] S. Maji, N. K. Vishnoi, and J. Malik. Biased normalized cuts. In *The Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2057-2064, 2011.

[73] M. Meila and J. Shi, A random walks view of spectral segmentation. In *The 8th International Workshop on Artificial Intelligence and Statistics* (AISTATS), 2001

[74] D. J. C. MacKay, Information Theory, Inference and Learning Algorithms, Cambridge University Press, 2003.

[75] B. Mohar, The Laplacian Spectrum of Graphs. In *Graph theory, Combinatorics, and Applications*. Vol. 2 (Kalamazoo, MI, 1988), New York: Wiley, 1991, pp. 871-898.

[76] T. Nepusz, L. Négyessy, G. Tusnády, F. Bazsó, Reconstructing Cortical Networks: Case of Directed Graphs with high level of Reciprocity. Handbook of Large-Scale Random Networks, eds. B. Bollobás, R. Kozma, and D. Miklós (Springer, Berlin), pp. 325-368, 2008.

[77] T. Nepusz, F. Bazsó, Likelihood-based Clustering of Directed Graphs, In *Proceedings of the Third International Symposium on Computational Intelligence and Intelligent Informatics of the IEEE*, pp. 189-194, 2007.

[78] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: T.G. Dietterich, S. Becker, Z. Ghahramani (Eds.), *Proceedings of the Fourteenth Conference on Advances in Neural Information Processing Systems*, vol. 2, The MIT Press, Cambridge, MA, USA, 2002.

[79] M. Pavan and M. Pelillo, Dominant Sets and Pairwise Clustering. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29, pp. 167-172, 2007.

[80] Z. A. Pardos, N. T. Heffernan. In Press: Using HMMs and bagged decision trees to leverage rich features of user and skill from an intelligent tutoring system dataset. In *The Journal of Machine Learning Research C & WP*, 2011.

[81] P. I. Pavlik, H. Cen, K. R. Koedinger, Performance Factor Analysis A New Alternative to Knowledge Tracing. In *The Proceedings of the 14th International Conference on Artificial Intelligence in Education*, Brighton, UK, pp. 531-538, 2009.

[82] V. Pehkonen and H. Reittu, Szemerdi-type clustering of peer-to-peer streaming system, In *The Proceedings of the 2011 International Workshop on Modeling, Analysis and Control of Complex Networks*, 2011.

[83] P. Raghavan, Probabilistic Construction of Deterministic Algorithms: Approximating packing integer programs, In *Journal of Computer and System Sciences*, 37 pp.130-143, 1988.

[84] L. Razzaq, M. Feng, G. Nuzzo-Jones, N. T. Heffernan, K. R. Koedinger, B. Junker, S. Ritter, A. Knight, C. Aniszczyk, S. Choksey, T. Livak, E. Mercado, T. E. Turner, R. Upalekar, J. A. Walonoski, M. A. Macasek, and K. P. Rasmussen, The Assistment Project: Blending Assessment and Assisting. In C. K. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds). *Proceedings of the 12th International Conference on Artificial Intelligence in Education*, Amsterdam. ISO Press, pp 555-562, 2005.

[85] S. Ritter, T. Harris, T. Nixon, D. Dickison, R. Murray, B. Towle, Reducing the knowledge tracing space. In *Proceedings of the International Conference on Educational Data Mining*, Cordoba, Spain, pp. 151-160, 2009.

[86] V. Rödl and M. Schacht, Regular partitions of hypergraphs: regularity lemmas, In *Combinatorics, Probability and Computing*, 16(6): pp. 833-885.

[87] V. Rödl, B. Nagle, J. Skokan, M. Schacht and Y. Kohayakawa, The hypergraph regularity method and its applications, In *Proceedings of the National Academy of Sciences* USA, 102 (2005), pp. 8109-8113.

[88] J. Shi, and J. Malik, Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (8), pp. 888-905, 2000.

[89] J. Spencer, Ten Lectures on the Probabilistic Method, SIAM, Philadelphia, 1987.

[90] A. Sperotto and M. Pelilo. Szemerédis Regularity Lemma and its Applications to pairwise clustering and segmentation. *In Energy minimization methods in Computer Science and Pattern Recognition*, volume 4679 of Lecture Notes in Computer Science. Springer, 2007.

[91] S. Still and W. Bialek, How Many Clusters? An Information Theoretic Perspective, *Neural Computation*, (2004), pp. 2483-2506.

[92] E. Szemerédi, Regular Partitions of Graphs, Colloques Internationaux C.N.R.S. Nº 260 - *Problèmes Combinatoires et Théorie des Graphes*, Orsay (1976), pp. 399-401.

[93] Gábor N. Sárközy, Fei Song, Endre Szemerédi, Shubhendu Trivedi, A Practical Regularity Partitioning Algorithm and its Applications in Clustering, In *Computing Research Repository , abs/1209.6540*, 2012.

[94] E. Szemerédi, On sets of integers containing no k elements in arithmetic progression, In *Acta Arithmetica* 27: pp. 199245, 1975.

[95] S. Trivedi, Z. A. Pardos, Neil T. Heffernan, Clustering Students to generate an ensemble to Improve Standard Test Score Predictions, G. Biswas et al. (Eds.): AIED 2011, LNAI 6738, In *The proceedings of the 15th International Conference on Artificial Intelligence in Education* 2011, Auckland, New Zealand, pp. 377384, 2011.

[96] S. Trivedi, Z. A. Pardos, Gábor N. Sárközy, Neil T. Heffernan, Spectral Clustering in Educational Data Mining. *In Proceedings of the 4th International Conference on Educational Data Mining* 2011, Eindhoven Netherlands, pp. 129-138, 2011.

[97] S. Trivedi, Z. A. Pardos, Neil T. Heffernan, The Utility of Clustering in Prediction Tasks. (Under Review) In *IEEE Transactions on Systems, Man and Cybernetics: Part B*, Submitted: 2011.

[98] S. Trivedi, Z. A. Pardos, Gábor N. Sárközy, Neil T. Heffernan,Co-Clustering by Spectral Bipartite Graph Partitioning for Out-of-Tutor Prediction, In *The Proceedings of the 5th International Conference on Educational Data Mining* 2012, Crete Greece, 2012.

[99] Z. A. Pardos, S. Trivedi, N. T. Heffernan and G. N. Sárközy, Clustered Knowledge Tracing, In *The Proceedings of the 11th International Conference on Intelligent Tutoring Systems* 2012, Chania, Greece.

[100] L. Valiant, A theory of the learnable, In *Communications of the ACM*, vol. 27, pp.11341142, Nov. 1984.

[101] V. Vapnik, The Nature of Statistical Learning Theory, 1995, New York, Springer-Verlag.

[102] M. Wu and B. Schölkopf, A Local Learning Approach for Clustering. In *Proceedings of the Neural Information Processing Systems*, pp. 1529-1536, 2007.

[103] L. Zelnik-Manor, P. Perona, Self-tuning Spectral Clustering. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems* 17, MIT Press, Cambridge, MA, pp. 1601-1608, 2005.

[104] D. Zhou, J. Huang, and B. Schölkopf. Learning with Hypergraphs: Clustering, Classification, and Embedding. In *Advances in Neural Information Processing Systems* , 19, pp. 16011608, 2007

[105] X. Zhu, Semi-Supervised Learning Literature Survey, *University of Wisconsin, Madison Technical Report*, PDF available online `http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf`, 2005.