

Metric Learning for Clustering in Streaming Large-Scale Data

Parag Jain

A Thesis Submitted to
Indian Institute of Technology Hyderabad
In Partial Fulfillment of the Requirements for
The Degree of Master of Technology



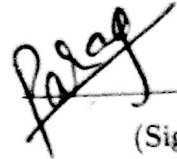
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Department of Computer Science & Engineering

June 2015

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.



(Signature)

(Parag Jain)

CS13M1008

(Roll No.)

Approval Sheet

This Thesis entitled Metric Learning for Clustering in Streaming Large-Scale Data by Parag Jain is approved for the degree of Master of Technology from IIT Hyderabad

C. Krishna Mohan

(_____) Examiner
Dr. C. Krishna Mohan
IITH

A.H. 25.5.15

(_____) Examiner
S. Mohana Chandra Prasad
IITH

V. B. Vineeth

(Dr. Vineeth N Balasubramanian) Adviser
Dept. of Computer Science & Engineering
IITH

Naveen

(_____) Chairman
Dr. Naveen Sivadasan
IITH

Acknowledgements

First and foremost, I would like to thank my advisor, Dr. Vineeth N Balasubramanian, for all his guidance and patience while introducing me to the research world. I am not sure how much I could make it but I have always strived to live up his standards. I also thank Indian National Centre for Ocean Information Services(INCOIS), specially Dr. Udaya Bhaskar, for bringing up this interesting problem.

I would particularly like to thank my friends and senior Debaditya Roy and Adepu Ravi Sankar for all their technical and non-technical discussions which motivated me towards this awesome field of machine learning in first place.

Finally, I would like to thank Dept. of Computer Science and Engineering, IIT Hyderabad for providing all the resources to pursue this work

Dedication

To my family & friends

Abstract

Given enormous amount of data produced each day it would be immensely useful if we could use it to learn hidden patterns in the data without the need for explicit labels. Clustering is one of the most popular approaches to label-less or unsupervised learning where the goal is to group together data points (for example, images, objects, web articles etc) into meaningful sub-classes called clusters. Although clustering is a well studied problem in machine learning but being unguided in nature, it may result in uninteresting patterns or trends. In general clustering is considered to be an ill-posed problem and any type of user input will help in guiding clustering towards a useful solution. For specific problems supervised learning is a conventional alternative, but in the real world it is costly to manually label the data and a supervised approach is no longer an option.

Most clustering algorithms fundamentally depend on the measure of similarity or dissimilarity of data points. Traditional distance measures like Euclidean distance, Mahalanobis distance etc. can be employed for measuring similarity but the choice of a particular measure depends on the problem and geometry of data itself. This raises a couple of issues. How can one learn a distance metric from the data according to the geometric properties of data? How can a few data points be selected intelligently from the entire dataset and expert knowledge be used to provide labels or give hints about them? If these questions can be answered sufficiently, it can lead to a significant improvement in results over a fully unsupervised approach.

In this work we present some new ideas to address the issues raised above. We propose a method to extend Diffusion Maps in an incremental framework using incremental Singular Value Decomposition (SVD) which allows us to approximate diffusion distance in a computationally efficient way. Our work also extends the Information Theoretic Metric Learning (ITML) by leveraging the idea of low dimensional embedding using manifold learning techniques. Apart from these ideas, we also propose to extend 'Pseudo-Metric Online Learning Algorithm' for Active learning for clustering by intelligently selecting the few points for which expert can provide hints. We have tested these proposed solutions on different standard UCI machine learning datasets.

Contents

| | |
|--|-------------|
| Declaration | ii |
| Approval Sheet | iii |
| Acknowledgements | iv |
| Abstract | vi |
| Nomenclature | viii |
| 1 Introduction | 1 |
| 1.1 Introduction to Metric Learning | 1 |
| 1.2 Application of Metric Learning in Clustering | 1 |
| 1.3 Applicability to Ocean Data Analysis | 1 |
| 1.4 Main objective | 2 |
| 1.5 Summary of contributions | 2 |
| 1.5.1 Contribution 1: Unsupervised Metric Learning using low dimensional embedding | 2 |
| 1.5.2 Contribution 2: Incremental Diffusion Maps | 3 |
| 1.5.3 Contribution 3: Online Active Metric Learning for Clustering | 3 |
| 2 Related work | 4 |
| 2.1 Metric Learning methods | 4 |
| 2.1.1 Supervised Metric Learning | 5 |
| 2.1.2 Unsupervised Metric Learning | 7 |
| 2.1.3 Active Metric Learning | 9 |
| 2.1.4 Review of Ocean Data analysis using Machine Learning | 9 |
| 3 Proposed Work | 11 |
| 3.1 Unsupervised Metric Learning using low dimensional embedding | 11 |
| 3.1.1 Laplacian eigenmaps | 11 |
| 3.1.2 Information Theoretic Metric Learning | 12 |
| 3.1.3 Proposed algorithm | 13 |
| 3.1.4 Results | 14 |
| 3.1.5 Summary | 15 |
| 3.2 Incremental Diffusion Maps | 15 |
| 3.2.1 Diffusion maps | 15 |
| 3.2.2 Updating SVD | 16 |
| 3.2.3 Proposed algorithm | 16 |

| | | |
|----------|---|-----------|
| 3.2.4 | Results | 17 |
| 3.2.5 | Summary | 17 |
| 3.3 | Online Active Metric Learning for Clustering | 18 |
| 3.3.1 | Pseudo-Metric Online Learning Algorithm(POLA) | 18 |
| 3.3.2 | Proposed Method | 19 |
| 3.3.3 | Results | 22 |
| 3.3.4 | Summary | 23 |
| 4 | Conclusion and future work | 24 |
| 4.0.5 | Directions for Future Work | 24 |
| | References | 24 |

Chapter 1

Introduction

1.1 Introduction to Metric Learning

Most machine learning algorithms, such as Support Vector Machines (SVM), kernel regression, Gaussian Processes, k-means or k-nearest neighbors (kNN) fundamentally depend on the representation of input data for which a reliable measure of (dis)similarity is known. This fundamental requirement of machine learning algorithms raises a question on how objects are compared. If an algorithm can determine how to measure (dis)similarity between objects then the subsequent tasks become relatively simpler.

One of the commonly used similarity measure is Euclidean distance, which is applied under the assumption that feature space is a Euclidean subspace. Other popular metrics like Mahalanobis distance, Manhattan distance can also be used but a simplistic assumption on the metric may not work well due to sophisticated hidden structure of feature space. However, manually deriving a good metric for a specific dataset is especially arduous. This has led to Metric learning, which can be viewed as a way to automatically learn a metric by understanding the hidden geometry of data.

1.2 Application of Metric Learning in Clustering

Metric learning is useful whenever any algorithm depends on the notion of distance measure between data instances. Clustering, being unsupervised in nature depends on the distance measure at a very fundamental level and using the correct metric can lead to a significant improvement in results [1]. There are many applications which involve metric learning in clustering [2] like detecting general trends in web by clustering text, clustering news in Google news etc.

1.3 Applicability to Ocean Data Analysis

All the analytic methods on oceanic data fundamentally requires us to compute the distance between two measurements. These data are in the form of measurements of temperature, salinity, dissolved oxygen etc which are taken at different places by floats in ocean around the world. Clustering is one of the widely used methods by scientists and experts to visualize and detect trends in ocean behaviour. Although one can manually analyze the clusters but for that correctly clustering the

enormous amount of data produced and choosing correct distance metric for it becomes crucial[3]. Further, most of the clustering methods used by them involve euclidean distance. This may not be a reasonable assumption as models are constantly updated to reflect the current behaviour. This makes metric learning more important.

To apply metric learning for oceanic data two factors have to be considered. First, the dimensionality of generated data is high, which makes it difficult to derive a correct distance metric manually. More importantly, ocean data is ever evolving which means updating the metric is a challenge that needs to be solved continuously. This makes the problem of finding trends in oceanic data an ideal candidate for applying automated metric learning for clustering.

1.4 Main objective

The goal of this work is to propose an unsupervised (or semi-supervised) metric learning method which can solve the problems discussed in the previous sections. Although an unsupervised metric learning method is an ideal solution it sets an ambitious aim of learning insights from nothing but raw data which is difficult to achieve without compromising on the accuracy of results. A more practical way is to get a few labels or hints from human experts which generally lead to a huge improvement in results. This is more cost effective than the completely supervised setting without a major sacrifice in performance. There are various challenges related to metric learning and clustering which need to be tackled and it is difficult to address all of them in a single solution.

In this work we have contributed different methods each of which can be used to address some challenges.

1.5 Summary of contributions

This section describes a summary of contributions of this work. Detailed descriptions of each approach is present in chapter 3.

1.5.1 Contribution 1: Unsupervised Metric Learning using low dimensional embedding

It can be observed that natural high dimensional data usually resides in an intrinsic low dimensional space. Manifold learning techniques like Laplacian eigenmaps[4], LLE[5], Diffusion maps[6] etc can be used to recover the intrinsic low dimensional geometry of the data. Understanding the geometry of data brings us close to correctly measure distance between data points. Low dimensional embedded space of Laplacian eigenmaps and Diffusion maps are euclidean which means we can use euclidean distance measure in embedded space to get (dis)similarity information.

Low dimensional embedding computed by Laplacian eigenmaps follows euclidean geometry, which means we can use euclidean distance in embedding space to measure distance between points. We leverage this property of Laplacian eigenmaps to compute similarity information between the points and use them as an input constraint to Information Theoretic Metric Learning(ITML)[7]. We combine Laplacian eigenmaps and ITML to get an unsupervised metric learning method. Once we

learn a metric by our method it can be used to measure distance for new data points without the need of any further projection or learning as required by other incremental methods[8].

To proof the proposed concept we have tested our proposed method on various standard UCI datasets and we observe that it performs better than using euclidean distance in original space.

1.5.2 Contribution 2: Incremental Diffusion Maps

Diffusion Maps[6] is a non-linear manifold learning technique, given data in high dimensional space it can learn a low dimensional embedding of the data such that local geometry of data is preserved. Embedded space is euclidean which means we can use euclidean distance as a (dis)similarity measure between data points. Since we have pairwise distance between points we can use this to get (dis)similarity information.

To the best of our knowledge there is no incremental method proposed which can be used to calculate diffusion distance for out of sample(new points which does not belong to training data) points. This solves the issue of efficiently calculating the distance for newer points which is one of the challenges in metric learning. To solve this we have leveraged the idea of incremental singular value decomposition to approximate diffusion distance for out-of-sample data and proposed an incremental version of diffusion maps. We have got considerable results in toy dataset but overall we conclude that due to high approximation error this method does not work for real datasets.

1.5.3 Contribution 3: Online Active Metric Learning for Clustering

Clustering is considered as an unsupervised learning problem and many clever algorithms have been proposed to solve it. In many cases where it is possible to get a few labels from the expert it is more practical to view clustering as semi-supervised learning problem, but in real world when unlabeled data is very large, to maximize the gain over the limited availability of expert feedback training examples should be actively selected as maximally informative ones.

Although we can select more informative points using active learning methods which distance measure to use still remains a question. To solve this we propose a two stage approach for 'Online Active metric learning for clustering'. Our method updates the learned metric using Pseudo-Metric Online Learning Algorithm(POLA)[9] based on actively selecting pairwise data constraints by iteratively improving the metric with user feedback.

Chapter 2

Related work

2.1 Metric Learning methods

Most popular machine learning algorithms like k-nearest neighbour, k-means, SVM uses a metric to identify the distance(or similarity) between data instances. It is clear that performances of these algorithm heavily depends on the metric being used. In absence of prior knowledge about data we can only use general purpose metrics like Euclidean distance, Cosine similarity or Manhattan distance etc, but these metric often fail to capture the correct behaviour of data which directly affects the performance of the learning algorithm. Solution to this problem is to tune the metric according to the data and the problem, manually deriving the metric for high dimensional data which is often difficult to even visualize is not only tedious but is extremely difficult. Which leads to put effort on *metric learning* which satisfies the data geometry.

Goal of metric learning algorithm is to learn a metric which assigns small distance to similar points and relatively large distance to dissimilar points.

Definition 1 *A metric on a set X is a function (called the distance function or simply distance).*

$$d : X \times X \rightarrow R,$$

where R is a set of real numbers, and for all x,y,z in X following condition are satisfied:

1. $d(x, y) \geq 0$ (non-negativity)
2. $d(x, y) = 0$ if and only if $x = y$ (coincidence axiom)
3. $d(x, y) = d(y, x)$ (symmetry)
4. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality).

If a function does not satisfy the second property but satisfies other three then it is called a **pseudometric**. But since most of the metric learning methods learns a pseudometric instead of a metric for rest of the discussion we will refer pseudometric as metric. Most of the metric learning methods in literature learns the metric of form,

$$d_M(x, x') = \sqrt{(x - x')^T M (x - x')} \tag{2.1}$$

which is Mahalanobis distance, where, $M = (A^{1/2})^T (A^{1/2})$ is a positive semi-definite matrix.

2.1.1 Supervised Metric Learning

Given a set of k dimensional data points $X \in \mathcal{R}^{N \times k}$, supervised metric learning methods learn a metric by using some similarity/dissimilarity information provided as constraints. There are different formulations proposed for supervised metric learning accommodating different kinds of constraints. In a general supervised setting most popular form of constraints used in literature [7] are:

1. Similarity/dissimilarity constraints

$$d_A(x_i, x_j) \leq u \quad (i, j) \in S$$

$$d_A(x_i, x_j) \geq l \quad (i, j) \in D$$

where, $(i, j) \in S$ for objects that are similar, $(i, j) \in D$ for objects that are dissimilar.

2. Relative constraints

$R = (x_i, x_j, x_k) : x_i$ should be more similar to x_j than to x_k ..

$$d_A(x_i, x_j) < d_A(x_i, x_k) - m$$

Where m is margin, generally m is chosen to be 1.

Next section summarizes some of the widely used methods.

2.1.1.1 Large Margin Nearest Neighbor

Large Margin Nearest Neighbour(LMNN) [10] learns a metric of form 2.1 parameterized by matrix A for kNN classification setting. Intuition behind this method is to learn a metric so that the k-nearest-neighbours belongs to the same class while instances with difference class labels should be separated by a margin.

Let $X_{n \times d}$ is a set of data points in d dimensional space, and class labels $y_i : i = 1 \dots n$ we define

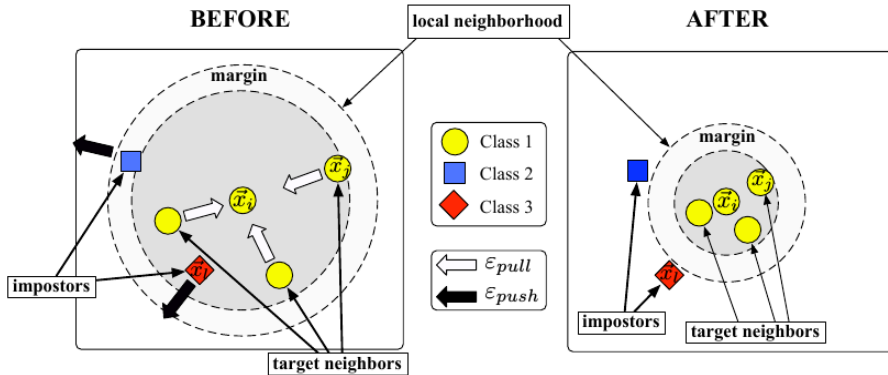


Figure 2.1: Schematic illustration of LMNN approach [10]

target neighbours for each point $x_i \in X$ as those points which are in k-nearest-neighbour of x_i and share the same label y_i and points which do not have same label as of x_i we call them *impostors*. Formulation consist of two terms which compete with each other, first term is to penalizes the large

distance between each point x_i and its target neighbors while second term penalizes small distance between x_i and impostors. Cost function is defined as:

$$\epsilon(L) = \sum_{ij} \eta_{ij} \|L(x_i - x_j)\|^2 + c \sum_{ij} \eta_{ij} (1 - Y_{il} [1 + \|L(x_i - x_j)\|^2 + \|L(x_i - x_l)\|^2]_+) \quad (2.2)$$

Where Y_{ij} and η_{ij} are binary matrices such that Y_{ij} is 1 when labels y_i and y_j match and η_{ij} is 1 when x_j is in the target neighbours of x_i , in second term $[z]_+ = \max(0, z)$ is a standard hinge loss function and c is some positive constant. Using cost function defined in 2.2 a convex optimization problem can be formulated as:

$$\begin{aligned} \min \sum_{ij} \eta_{ij} (x_i - x_j)^T M (x_i - x_j) + c \sum_{ij} \eta_{ij} (1 - Y_{il} \xi_{ijl}) \quad (2.3) \\ \text{subject to } (x_i - x_l)^T M (x_i - x_l) - (x_i - x_j)^T M (x_i - x_j) \geq 1 - \xi_{ijl} \\ \xi_{ijl} \geq 0 \\ M \succeq 0 \end{aligned}$$

where matrix $M = L^T L$ and η_{ijl} are slack variables.

2.1.1.2 Information Theoretic Metric Learning

Given similarity and dissimilarity constraints Information Theoretic Metric Learning (ITML) [11] learns a metric of form 2.1. Problem is formulated as a convex optimization using LogDet divergence:

$$\begin{aligned} \min_A D_{ld}(A, A_0) \\ \text{subject to } d_A(x_i, x_j) \leq u \quad \forall (x_i, x_j) \in S \\ d_A(x_i, x_j) \geq l \quad \forall (x_i, x_j) \in D \\ A \succeq 0 \end{aligned} \quad (2.4)$$

Details are described in section 3.1.2.

2.1.1.3 Mirror Descent for Metric Learning

Mirror Descent for Metric Learning, by Kunapuli and Shavlik [12], is online metric learning approach which learns a pseudo-metric of form,

$$d_M(x, z)^2 = (x - z)^T M (x - z)$$

given a pair of labeled points, $(x_t, z_t, y_t)^T$, where y_t denotes similarity/dissimilarity.

Taking μ as a margin, constraints can be written as,

$$\begin{aligned} y(\mu - d_M(x, z)^2) \geq 1 \\ l(M, \mu) = \max \{0, 1 - y(\mu - d_M(x, z)^2)\} \end{aligned}$$

Where $l(M, \mu)$ is hinge loss. To learn pseudo-metric incrementally from triplets, updates can be computed as,

$$M_{t+1} = \underset{M \succ 0}{\operatorname{argmin}} B_\psi(M, M_t) + \eta \langle \Delta_M l_t(M_t, \mu_t), M - M_t \rangle + \eta \rho \|M\|$$

$$\mu_{t+1} = \underset{\mu \geq 1}{\operatorname{argmin}} B_\psi(\mu, \mu_t) + \eta \Delta_\mu l_t(M_t, \mu_t)'(\mu - \mu_t).$$

Where $B_\psi(M, M_t)$ is bregman divergence, with $\psi(x)$ was taken as either squared-Frobenius distance and von Neumann divergence.

2.1.2 Unsupervised Metric Learning

Unsupervised metric learning is generally seen as a byproduct of manifold learning or dimensionality reduction algorithms, although metric learning has a direct connection between linear manifold learning techniques as it finally learns a projective mapping but for non linear techniques, which are more useful, connection is not exact and can only be seen with some approximations. Because of these limitations of manifold techniques unsupervised metric learning has its own importance. Unsupervised metric learning aims to learn a metric without any supervision, most of the method proposed in this area either solve this problem in a domain specific way like clustering Gupta Abhishek A. [13] or by understanding the geometric properties of data.

2.1.2.1 Diffusion Maps

Diffusion maps Coifman and Lafon [6] is a non-linear dimensionality reduction technique. Consider a graph $G = (\Omega, W)$ where $\Omega = \{x_i\}_{i=1}^N$ are data samples and W is a similarity matrix with $W(i, j) \in [0, 1]$. W is obtained by applying Gaussian kernel on distances,

$$W(i, j) = \exp\left\{\frac{-d^2(i, j)}{\sigma^2}\right\} \quad (2.5)$$

Using W we can obtain a transition matrix by row wise normalizing the similarity matrix:

$$P(i, j) = \frac{W(i, j)}{d_i} \text{ where, } d_i = \sum_{j=1}^N W_{ij} \quad (2.6)$$

Diffusion map introduce diffusion distance based on transition probabilities P of data, given as:

$$d_t^2 = \|P_t(i, :) - P_t(j, :)\|_{1/\phi}^2 \quad (2.7)$$

where, $P_t = P^t$.

2.1.2.2 Unsupervised metric learning using self-smoothing operator

Unsupervised metric learning using self-smoothing operator Jiang, Wang, and Tu [14] proposed a diffusion based approach to improve input similarity between data points. It uses similar framework as diffusion maps but instead of using the notion of diffusion distance it uses a Self Smoothing

Operator(SSO) which preserves the structure of weight matrix W described in equation 2.5. Main steps of SSO algorithm are summarized below:

1. Compute smoothing kernel: $P = D^{-1}W$, where D is a diagonal matrix such that $D(i, i) = \sum_{k=1}^n W(i, k)$
2. Perform smoothing for t steps: $W_t = WP^t$
3. Self-normalization: $W^* = \Gamma^{-1}W_t$ where Γ is a diagonal matrix such that $\Gamma(i, i) = W_t(i, i)$
4. Project W^* to psd cone $\hat{W}^* = psd(W^*)$

2.1.2.3 Unsupervised Distance Metric Learning using Predictability

Unsupervised distance metric learning using predictability Gupta Abhishek A. [13] learns a transformation of data which give well separated clusters by minimizing the *blur ratio*. This work proposes a two step algorithm to achieve this task which alternates between predicting cluster membership by using linear regression model and again cluster these predictions. Given input data matrix $X_{N \times p}$ with N number of points in p dimensional space goal is to find learn a mahalanobis distance metric $d(x, y) = \sqrt{(x - y)A(x - y)^T}$ which minimizes the blur ration defined as:

$$\min_{A, c} BR(A, c) \equiv \frac{SSC}{SST}$$

where SSC and SST are within cluster variance and total variance respectively.

2.1.2.4 Laplacian Eigenmaps

Laplacian eigenmaps learns a low dimensional embedding of the data such that the local geometry is preserved optimally using spectral decomposition of graph laplacian. Data is represented in the form of a graph which can be considered as an approximation of low dimensional manifold. Algorithm comprises of three steps:

1. Construct weighted graph: This steps computes a weighted graph representation W of input data by weighting the neighbourhood graph.
2. Construct graph laplacian: Calculate unnormalized graph laplacian as $L = D - W$
3. Calculate low dimensional embedding: Low dimensional embedding is calculates by doing eigen-decomposition of graph laplacian.

2.1.2.5 Why don't these work for us?

Unsupervised methods described in previous section has some limitations, manifold learning techniques like Diffusion maps or Laplacian eigenmaps aims to learn a low dimensional embedding which is then used to calculate distance between pair of points, but it does not provide us with a actual metric which can be used to measure distance between points which were not in the sample which essentially means we have to recompute the distances all again. Self smoothing operator approach 2.1.2.2 which is presented as an unsupervised metric learning approach has same limitation it cannot be used to compute a general metric. Cluster predictability and cluster membership approach

described in 2.1.2.3 learn a metric by learning a transformation by minimizing the blur ratio, but this transformation may not be optimal if new data is added to input set.

2.1.3 Active Metric Learning

Active learning is a form of semi-supervised learning, difference is that in an active learning setup algorithm itself chooses what data it wants to learn. Aim is to select data instances which is most effective in training the model this saves significant cost to the end user end by asking less queries.

2.1.3.1 Active Metric Learning for Object Recognition

Active metric learning for object recognition by Ebert, Fritz, and Schiele [15] propose to combine metric learning with active sample selection strategy for classification. This work explores to exploitation(entropy based and margin based) and two exploration(kernel farthest first and graph density) based strategy for active sample selection. To learn a metric Information theoretic metric learning is used, which is combined with active sample selection is two different modes,

1. Batch active metric learning: In this mode metric is learned only once, it starts with querying the desired number of labeled data points according to the chosen sample selection strategy and learns a metric based on this labeled data.
2. Interleaved active metric learning: This approach alternates between active sample selection and metric learning.

2.1.3.2 Metric+Active Learning and Its Applications for IT Service Classification

Metric+Active learning Wang et al. [16] learns a metric for ticket classification which are used by IT service providers. This work proposed two methods to solve this problem:

1. Discriminative Neighborhood Metric Learning (DNML): DNML aims to minimize the local discriminability of data which is same as maximize the local scatterness and to minimize the local compactness simultaneously.

$$J = \frac{\sum_{j:x_j \in \mathcal{N}_i^o} (x_i - x_j)^T C (x_i - x_j)}{\sum_{k:x_k \in \mathcal{N}_i^e} (x_i - x_k)^T C (x_i - x_k)}$$

Where \mathcal{N}_i^o is nearest points from x_i with same labels as of x_i , \mathcal{N}_i^e are nearest points from x_i which have different labels than of x_i .

2. Active Learning with Median Selection(ALMS): ALMS improves Transductive Experimental Design (TED) by using available labelled information.

2.1.4 Review of Ocean Data analysis using Machine Learning

Machine learning techniques has been applied in many ocean data analysis tasks. One of the problem is ocean biome classification. Biomes are region on earth with similar climate,ocean biome classification can be formulated as a clustering problem which is well studied in machine learning literature. Below are reviewed some of the published work in this area.

2.1.4.1 Mapping Uncharted Waters

This work by Lewis et al. [3], provides quantitative classification of ocean biomes by directly applying leading methods on high dimensional data analysis. Analysis was done on World Ocean Atlas 2005(WOA05) data considering in total 14 parameters including temperature, salinity, phosphate etc measured at 9105 locations.

Different methods like k-means, principal component analysis (PCA), multidimensional scaling, ISOMAP, Maximum variance unfolding (MVU) were compared. This work concludes that MVU works best for those regions seeking to measure fine spatial or temporal gradients.

2.1.4.2 ST-DBSCAN

Spatio-temporal DBSCAN(ST-DBSCAN), proposed by Birant and Kut [17], is an extension of density based DBSCAN algorithm which uses two parameters instead of one in DBSCAN, Eps1 and Eps2, to determine whether set of points can be considered in a same cluster.

Eps1 measures the geographical closeness of two points(latitude and longitude), while Eps2 measures the similarity between parameters. Neighbor of object p in dataset D is defined by the points in radius,

$$\max \{ \text{dist}(p, q) | q \in D \wedge \text{dist1}(p, q) \leq \text{Eps1} \wedge \text{dist2}(p, q) \leq \text{Eps2} \}$$

Chapter 3

Proposed Work

In this chapter we describe the details of major contributions of this work. Chapter is divided into subsections each of them gives details of method proposed, results and summary.

3.1 Unsupervised Metric Learning using low dimensional embedding

Goal of approach presented in this section is to come up with an unsupervised metric learning technique. Unsupervised metric learning has been generally studied as a byproduct of dimensionality reduction or manifold learning techniques. Manifold learning techniques like Diffusion maps, Laplacian eigenmaps discussed in previous chapter 2 has a special property that embedded space is euclidean. Although laplacian eigenmaps can provide us with some (dis)similarity information it does not provide with a metric which can further be used on out-of-sample data. On other hand supervised metric learning technique like ITML which can learn a metric needs labelled data for learning.

In this approach we combine Laplacian eigenmaps and Information Theoretic Metric Learning(ITML) to form an unsupervised metric learning method. We first project data into a low dimensional manifold using Laplacian eigenmaps, in embedded space we use euclidean distance to get an idea of similarity between points. If euclidean distance between points in embedded space is below a threshold t_1 value we consider them as similar points and if it is greater than a certain threshold t_2 we consider them as dissimilar points. Using this we collect a batch of similar and dissimilar points which are then used as a constraints for ITML algorithm and learn a metric. To prove this concept we have tested our approach on various UCI machine learning datasets.

3.1.1 Laplacian eigenmaps

Laplacian eigenmaps learns a low dimensional representation of the data such that the local geometry is optimally preserved, this low-dimensional manifold approximate the geometric structure of data. Steps below describes the methods in detail.

Consider set of data points $X \in \mathcal{R}^N$, goal of laplacian eigenmaps is to find an embedding in m dimensional space where $m < N$ preserving the local properties of data.

1. Construct a graph $G(V, E)$ where E is set of edges and V is a set of vertices. Each node in the

graph G corresponds to a point in X , we connect any two vertices v_i and v_j by an edge if they are close, closeness can be defined in 2 ways:

- (a) $\|x_i - x_j\|^2 < \epsilon$, $\|\cdot\|$ is euclidean norm in \mathcal{R}^N or,
- (b) x_i is in k nearest neighbour of x_j

here ϵ & k are user defined parameters.

2. We construct a weight matrix $W(i, j)$ which assigns weights between each edge in the graph G , weights can be assigned in two ways:

- (a) Simple minded approach is to assign $W(i, j) = 1$ if vertices v_i and v_j are connected otherwise 0.
- (b) Heat kernel based, we assign weight $W(i, j)$ such that:

$$W(i, j) = \begin{cases} \exp\left(\frac{\|x_i - x_j\|^2}{t}\right) & \text{if } v_i \text{ and } v_j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}$$

3. Construct laplacian matrix $L = D - W$ of the graph G , where D is a diagonal matrix with $D_{ii} = \sum_j W(i, j)$. Final low dimensional embedding can be computed by solving generalized eigen decomposition

$$Lv = \lambda Dv$$

Let $0 = \lambda_0 \leq \lambda_1 \dots \leq \lambda_m$ be the first smallest $m + 1$ eigenvalues, choose corresponding eigenvectors $v_1, v_2 \dots v_m$ ignoring eigenvector corresponding to $\lambda_0 = 0$. Embedding coordinates can be calculated as mapping:

$$x_i \in \mathcal{R}^N \mapsto y_i \in \mathcal{R}^m$$

where $y_i^T = [v_1(i), v_2(i), \dots v_m(i)]$

$y_i, i = 1, 2, \dots n$ is the coordinates in m dimensional embedded space.

3.1.2 Information Theoretic Metric Learning

Information Theoretic Metric Learning (ITML) [11] learns a Mahalanobis distance metric that satisfies some given similarity and dissimilarity constraints on input data. Goal of ITML algorithm is to learn a metric of form $d_A = (x_i - x_j)^T A (x_i - x_j)$ according to which similar data points are close relative to dissimilar points.

ITML starts with an initial matrix d_{A_0} where A_0 can be set to identity matrix (I) or inverse of covariance of the data and eventually learns a metric d_A which is close to starting metric d_{A_0} and satisfies the defined constraints. To measure distance between metrics it exploits the bijection between Gaussian distribution with fixed mean μ and Mahalanobis distance,

$$\mathcal{N}(x|\mu, A) = \frac{1}{Z} \exp\left(-\frac{1}{2} d_A(x, \mu)\right)$$

Using the above connection, the problem is formulated as:

$$\begin{aligned}
& \min_A \int \mathcal{N}(x, \mu, A_0) \log \left(\frac{\mathcal{N}(x, \mu, A_0)}{\mathcal{N}(x, \mu, A)} \right) dx \\
& \text{subject to } d_A(x_i, x_j) \leq u \quad \forall (x_i, x_j) \in S, \\
& \quad \quad \quad d_A(x_i, x_j) \geq l \quad \forall (x_i, x_j) \in D, \\
& \quad \quad \quad A \succeq 0
\end{aligned} \tag{3.1}$$

Above formulation can be simplified by utilizing the connection between KL-divergence and LogDet divergence which is given as,

$$\begin{aligned}
& \int \mathcal{N}(x, \mu, A_0) \log \left(\frac{\mathcal{N}(x, \mu, A_0)}{\mathcal{N}(x, \mu, A)} \right) dx = \frac{1}{2} D_{ld}(A, A_0) \\
& \text{where, } D_{ld}(A, A_0) = \text{tr}(AA_0^{-1}) - \log \det(AA_0^{-1}) - d
\end{aligned} \tag{3.2}$$

Using 3.2 and 3.1 problem can be reformulated as:

$$\begin{aligned}
& \min_A D_{ld}(A, A_0) \\
& \text{subject to } d_A(x_i, x_j) \leq u \quad \forall (x_i, x_j) \in S \\
& \quad \quad \quad d_A(x_i, x_j) \geq l \quad \forall (x_i, x_j) \in D \\
& \quad \quad \quad A \succeq 0
\end{aligned} \tag{3.3}$$

Above formulation can be solved efficiently using bregman projection method as described in Davis et al. [11].

3.1.3 Proposed algorithm

We propose a method which combines Laplacian eigenmaps and ITML to form an unsupervised metric learning method. Laplacian eigenmaps as described in 3.1.1 can be used to recover underlying low dimensional manifold of data where we can use euclidean distance to get (dis)similarity information using which we can learn a metric using supervised metric learning settings like ITML. In box 3.1.3 we describe the details of proposed algorithm.

Manifold + supervised metric learning

Input:

$X \in N \times k$, is input data in k dimensional space

t_s : threshold for similarity

t_d : threshold for dissimilarity

ϵ, m : parameters for laplacian eigenmaps algorithm, $m < k$

Output: Learned metric $A_{k \times k}$ **Steps:**

1. Construct low dimensional embedding:
 $\mathcal{E} = \text{laplacianEigenmaps}(X, \epsilon, m)$
2. Construct similarity and dissimilarity pairs:
for each pair $(x_i, x_j) \in \mathcal{E}$:
 $p = \|x_i - x_j\|^2$
if $p \leq t_s$ then $S \leftarrow S \cup (x_i, x_j)$
if $p \geq t_d$ then $D \leftarrow D \cup (x_i, x_j)$
3. Apply ITML 3.1.2 procedure to learn metric:
 $A = \text{itml}(X, S, D)$

We reduce the time complexity of above algorithm by limiting number of similar and dissimilar points.

Calculating threshold: Manually setting thresholds for similarity and dissimilarity can be difficult in real case, but we can set these thresholds with a simple procedure of calculating the distance extremes for data in embedded space \mathcal{E} . A safe way for setting threshold is to compute histogram of distances and set t_s & t_d to be the 5th and 95th percentiles respectively.

3.1.4 Results

We have evaluated our method on different UCI datasets[18], to best of our knowledge there is no other method that does exactly what we tried we compare our algorithm with euclidean distance, we construct similar and dissimilar pairs using euclidean distance and then learn a metric using ITML. We split each dataset randomly into two parts 80% for training and 20% for testing, to evaluate the learned metric with k-NN classification using learned metric as distance measure. All results presented are the average of 5 runs.

| Dataset | Proposed method | Euclidean + ITML |
|--------------------|-----------------|------------------|
| Letter recognition | 95.25 | 93.75 |
| Iris | 83.3 | 76.6 |
| Scale | 84.7 | 82.6 |
| Yeast | 60.7 | 59.4 |
| Wine | 75.9 | 74.1 |

The method we proposed in this section is general and the metric learned can be used for clustering dataset.

3.1.5 Summary

From results it is clear that performing low dimensional embedding to obtain similar and dissimilar pairs performs better than directly apply a general measure than euclidean distance. One important thing to notice is that comparison is not been made directly between euclidean distance and proposed method rather in both the cases we learned a metric using ITML . We can notice that results are still close which implies that there is still some scope for improvement.

3.2 Incremental Diffusion Maps

3.2.1 Diffusion maps

Diffusion maps[6] are non-linear dimensionality reduction technique. It achieves dimensionality reduction by exploiting relation between Markov chains and heat diffusion. Diffusion map embeds data into low dimensional space such that euclidean distance between points in embedded space is approximated as diffusion distance in the original feature space.

Consider a graph $G = (\Omega, W)$ where $\Omega = \{x_i\}_{i=1}^N$ are data samples and W is a similarity matrix with $W(i, j) \in [0, 1]$. W is obtained by applying Gaussian kernel on distances,

$$W(i, j) = \exp\left\{\frac{-d^2(i, j)}{\sigma^2}\right\} \quad (3.4)$$

where $d(i, j)$ is the distance between x_i and x_j and σ is kernel size. Using W we can obtain a transition matrix by row wise normalizing the similarity matrix:

$$P(i, j) = \frac{W(i, j)}{d_i} \quad \text{where, } d_i = \sum_{j=1}^N W_{ij} \quad (3.5)$$

Transition matrix reflects the local geometry of the data, where $p(x, y)$ is the probability of transition from x to y in one step. If we look forward in time than P^t gives the probability of transition from x to y in t time steps. Intuitively what that means is running the diffusion in time will reveal the geometric structure of data at different scales.

Diffusion process We define a new kernel L using normalized laplacian:

$$L^{(\alpha)} = D^{-\alpha} W D^{-\alpha} \quad (3.6)$$

where, D is diagonal matrix such that $D_{ii} = \sum_j W_{i,j}$ and $\alpha \in \mathbb{R}$. Apply weighted graph Laplacian normalization to this kernel:

$$M = (D^{(\alpha)})^{-1} L^{(\alpha)} \quad (3.7)$$

where $D^{(\alpha)}$ is a diagonal matrix such that $D_{ii}^{(\alpha)} = \sum_j L_{i,j}^{(\alpha)}$

Diffusion distance can be defined in terms of eigenvectors of matrix M^t

$$D_t(i, j) = \|\Psi_t(x_i) - \Psi_t(x_j)\|^2 \quad (3.8)$$

ψ_l is right eigenvector and λ_i are eigenvalues of M^t and $\Psi_t(x) = (\lambda_1^t \psi_1(x), \lambda_2^t \psi_2(x), \dots, \lambda_k^t \psi_k(x))$.

3.2.2 Updating SVD

Given matrix $A_{m \times n}$ and $\hat{A}_{m \times n} = U\Sigma V'$ where $\hat{A}_{m \times n}$ is rank-k approximation of $A_{m \times n}$, Zha and Simon [19] describes the procedure to get the approximate rank-k approximation of $[\hat{A}_{m \times n}, B_{m \times r}]$ and $[\hat{A}_{m \times n}; B_{r \times n}]$. Method proposed by Zha and Simon [19] is summarized below,

Updating Columns

1. Let the QR decomposition of $(I - UU')B$ be $(I - UU')B = QR$ where R is upper triangular.
2. Get SVD decomposition of $\begin{bmatrix} \Sigma & U'B \\ 0 & R \end{bmatrix} = \hat{U}\hat{\Sigma}\hat{V}'$
3. Then best rank-k approximation of $[\hat{A}_{m \times n}, B_{m \times r}]$ is given as $([U, Q]\hat{U})\hat{\Sigma}\begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix}\hat{V}'$

Updating Rows

1. Let the QR decomposition of $(I - VV')B'$ be $(I - VV')B' = QL'$ where L' is lower triangular.
2. Get SVD decomposition of $\begin{bmatrix} \Sigma & 0 \\ BV & L \end{bmatrix} = \hat{U}\hat{\Sigma}\hat{V}'$
3. Then best rank-k approximation of $[\hat{A}_{m \times n}; B_{m \times r}]$ is given as $\begin{bmatrix} U & 0 \\ 0 & I \end{bmatrix}\hat{U}'\hat{\Sigma}([V, Q]\hat{V})$

3.2.3 Proposed algorithm

In box 3.2.3 we have described the basics steps of proposed incremental diffusion maps algorithm. Calculation can be done in a very efficient way by storing the previous values of row sum at step 2 and step 4 of the algorithm.

Incremental Diffusion map algorithm

Input:

D: distance matrix of size $N \times N$

T: distance matrix of new points $N + p \times p$

n: dimension of embedding

U,S,V : SVD of markov transition matrix M^t obtained using old points

Diffusion maps paramenets: σ, α

Output: $\hat{U}, \hat{S}, \hat{V}$, DD: Diffusion distance matrix of size $N + p \times N + p$

Steps:

1. Update D by adding new rows and columns from T to get $\hat{D}(N + p \times N + p)$
2. Apply gaussian kernel to get W on \hat{D}
3. Compute new kernel by applying laplacian normalization, $L^{(\alpha)} = D^{-\alpha} W D^{-\alpha}$
4. Calculate $M = (D^{(\alpha)})^{-1} L^{(\alpha)}$
5. Get new rows R and columns C from M
6. Get $\hat{U}, \hat{S}, \hat{V}$ by using update SVD procedure 3.2.2
7. Use \hat{U}, \hat{S} to get new diffusion distance matrix $DD_t(i, j) = \|\Psi_t(x_i) - \Psi_t(x_j)\|^2$

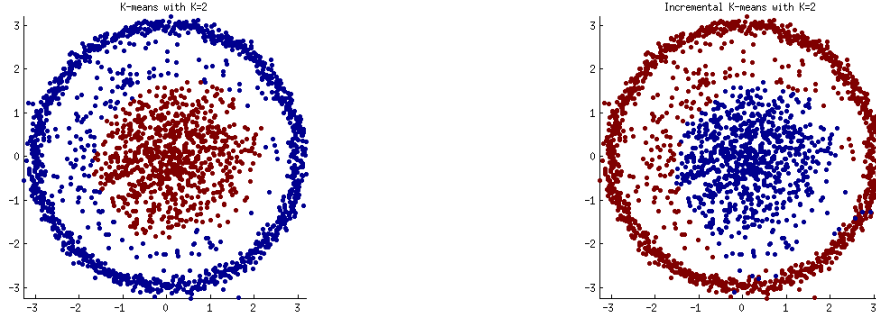
3.2.4 Results

To check the performance of proposed method we have used a toy dataset having two non-linearly separable clusters, total number of points in dataset are 2000, to test the effectiveness of incremental procedure we divide the dataset into 2 parts of 1600 and 400 points. Initial 1600 points were used in batch mode to get initial embedding then we update the embedding using proposed incremental diffusion maps procedure. To simulate real world setting we update the embedding 10 points at a time, calling incremental update 40 time in this case. Once we get the final embedding after updating all 400 points for both batch and incremental setting we use k-means with $k = 2$ to visualize the effectiveness of diffusion distance learned. Final results are plotted in figures 3.1.

All the experiment was done on Sony machine with i3 processor and 4GB RAM. It can be observed that the results are very close with very less run time than batch mode. We have tested this methods on other datasets but due to high approximation error in markov matrix calculation and update svd procedure results are not good.

3.2.5 Summary

From our results we conclude that approximation in markov matrix and svd update leads to bad results in real datasets, since we have implemented the idea using very basic svd update procedure and as there are many other promising svd update procedures like Chen and Candan [20] has been proposed in recent years we believe there there still scope of improvement.



(a) Clusters using batch mode. Run time: 27.5 sec (b) Clusters using incremental mode. Run time 6.4 sec

Figure 3.1: Result incremental diffusion maps

3.3 Online Active Metric Learning for Clustering

In this work we propose an online pairwise constrained metric learning for clustering by actively selecting informative similar and dissimilar pairs. We focus on selecting pairwise constraints for two reasons, first is in real world with large number of clusters it is easier for user to say whether two given points are similar or not then providing actual labels for points and second pairwise constraints is a better choice for updating the clusters. We use Pseudo-Metric Online Learning Algorithm(POLA) [9] combined with our active pairwise selection method to make an online active metric learning method for clustering. We propose a two stage approach, first stage actively select new pair of points which are then posed as a query to the user to label them as similar or dissimilar, in second stage we update our learned metric in online manner using POLA based on new similarity/dissimilarity constraints. Since we update our metric online there is a little learning overhead to clustering algorithm.

3.3.1 Pseudo-Metric Online Learning Algorithm(POLA)

Let \mathcal{X} denotes the feature space. POLA learns a metric of form,

$$d_A(x, x') = \sqrt{(x - x')'A(x - x')}$$

Algorithm receives new samples as similarity and dissimilarity pairs in the form of $z = (x, x', y) \in (\mathcal{X} \times \mathcal{X} \times +1, -1)$, where $y = +1$ if pair (x, x') are similar otherwise $y = -1$. Loss function is defined as,

$$l_\tau(A, b) = \max\{0, y_\tau(d_A(x, x')^2 - b) + 1\} \quad (3.9)$$

where, $b \in \mathbb{R}$ is threshold, if $d_A(x, x')$ is greater than b we predict pairs to be dissimilar otherwise similar. Goal is to learn matrix threshold pair (A_τ, b_τ) which minimize the cumulative loss. At each step algorithm receives pair (x, x', y) where $y = +1$ if pair (x, x') are similar otherwise $y = -1$ and update matrix threshold pair (A_τ, b_τ) in two steps.

1. Projecting current solution (A_τ, b_τ) onto set C_τ which,

$$C_\tau = \{(A, b) \in \mathbb{R}^{(n^2+1)} : l_\tau(A, b) = 0\}$$

C_τ is a set of all matrix-threshold pairs which gives zero loss on (x, x', y) .

2. Then project new matrix-threshold pair to set of all admissible matrix-threshold pairs C_a ,

$$C_a = \{(A, b) \in \mathbb{R}^{(n^2+1)} : A \succeq 0, b \geq 1\}$$

Projecting onto C_τ : We denote matrix-threshold pair as a vector $w \in \mathbb{R}^{n^2+1}$, and $\mathcal{X}_\tau \in \mathbb{R}^{n^2+1}$ is vector of matrix-scalar pair $(-y_\tau v_\tau v_\tau^t, y_\tau)$, where $v_\tau = x_\tau - x'_\tau$. Using this we can rewrite set C_τ as,

$$C_\tau = \{w \in \mathbb{R}^{n^2+1} : w\mathcal{X}_\tau \geq 1\}$$

Now we can write projection of w_τ onto C_τ as,

$$\mathcal{P}_{C_\tau}(w_\tau) = w_\tau + \alpha_\tau \mathcal{X}_\tau \quad (3.10)$$

where, $\alpha_\tau = 0$ if $w_\tau \mathcal{X}_\tau \geq 1$ otherwise $\alpha_\tau = (1 - w_\tau \mathcal{X}_\tau) / \|\mathcal{X}_\tau\|_2^2$. Which we can rewrite as,

$$\alpha_\tau = \frac{l_\tau(A_\tau, b_\tau)}{\|\mathcal{X}_\tau\|_2^2} = \frac{l_\tau(A_\tau, b_\tau)}{\|v_\tau\|_2^4 + 1}$$

and based on this we can update matrix and threshold,

$$A_{\hat{\tau}} = A_\tau - y_\tau \alpha_\tau v_\tau v_\tau^t, \quad b_{\hat{\tau}} = b_\tau + \alpha_\tau y_\tau \quad (3.11)$$

Projecting onto C_a : Projecting b_τ on set $\{b \in \mathbb{R} : b \geq 1\}$ is straightforward and can be achieved as $b_{\tau+1} = \max\{1, b_\tau\}$, for projecting $A_{\hat{\tau}}$ has two cases,

- $y_\tau = -1$: In this case $A_{\hat{\tau}}$ becomes $A_{\hat{\tau}} = A_\tau + \alpha_\tau v_\tau v_\tau^t$ and $\alpha \geq 0$ therefore $A_{\hat{\tau}} \succeq 0$ and hence $A_{\tau+1} = A_{\hat{\tau}}$
- $y_\tau = 1$: In this case we can write $A_{\hat{\tau}} = \sum_{i=1}^n \lambda_i u_i u_i^t$ where u_i is the i^{th} eigenvector of $A_{\hat{\tau}}$ and λ_i is corresponding eigenvalue, we can get $A_{\tau+1}$ by projecting $A_{\hat{\tau}}$ to PSD cone as,

$$A_{\tau+1} = \sum_{i:\lambda_i>0}^n \lambda_i u_i u_i^t$$

For every new sample we update by successively projecting (A_τ, b_τ) to C_τ and C_a .

3.3.2 Proposed Method

In this section we describe our proposed active pair selection method and combine it with POLA to form active metric learning.

Selecting new pairs: Idea behind selecting new pairs is as follows, we want to select those points which can tell whether we should reduce or expand the boundary of current cluster, in this case we are being conservative and trust the overall structure of cluster, at the same time we want to select some points which can expose raider to the metric forcing it to diverge. We propose a heuristic based approach to select these points in a given clustering.

Given a set of points $X \in \mathbb{R}^{n \times d}$ and cluster belongingness matrix $C_{ij} \in \mathbb{R}^{n \times k}$ such that $C_{ij} = 1$ if point x_i belongs to cluster c_j otherwise $C_{ij} = 0$ where $C = c_1, c_2 \dots c_k$ are number of clusters. Figure

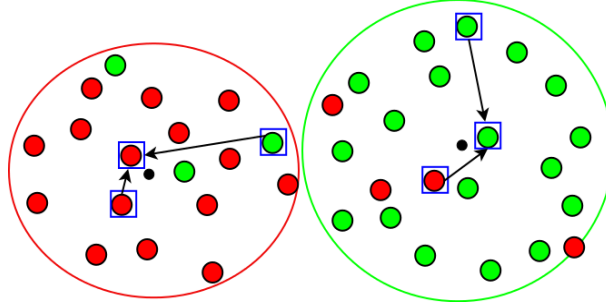


Figure 3.2: Schematic illustration of actively selecting pairs

3.2 shows an illustration with two clusters shown with red and green colors, points inside a box are selected points which are paired with point shown by arrow. For each cluster we select a set of points which are farthest from the cluster center and a set of points which are closest to cluster centre and make pairs as $(cluster_centre, selected_point)$. Since cluster centre may not be an actual data point we replace it with the nearest data point and query these pairs to the user for similarity information. We apply *alpha-trimming* which reduces chances of selecting outliers in case of selecting boundary points and give some space to select points near cluster center.

Algorithm 1 Algorithm to select pairs

```
1: procedure SELECTPAIRS
  Input:  $X, C, p, \hat{S}, \hat{D}, A, r, \alpha$ 
   $X_{n \times d}$  is input feature matrix
   $C_{n \times k}$  is cluster belongingness matrix
   $p$  is number of pairs to select
   $\alpha$  is alpha-trimming parameter
   $A$  is current distance matrix
   $r$  is ratio of near-center & boundary points to select
  Output:  $S$  set of selected similar pairs
   $D$  set of selected dissimilar pairs
2:   Calculate cluster density vector  $W_{k \times 1}$  such that  $w_i$  is density of  $i^{th}$  cluster
3:   for  $i = 1$  to  $k$  do
4:     Select points  $Y \in c_i$  & calculate mean  $\mu$  of all points  $y \in Y$ 
5:     Find point  $c$  closest to  $\mu$ 
6:      $ns \leftarrow p \times r$ 
7:      $nd \leftarrow p - ns$ 
8:      $\hat{Y}' \leftarrow sorted(Y)$ 
9:      $\hat{Y} \leftarrow$  trim top and bottom  $\alpha$  percentage of points
10:    for  $s = 1 : ns$  do
11:       $l \leftarrow \hat{Y}(end - s)$ 
12:      if  $(c, l) \notin \hat{S}$  &  $(c, l) \notin \hat{D}$  then  $t \leftarrow querySimilarity(c, l)$ 
13:        if  $t == 1$  then
14:           $S \leftarrow \hat{S} \cup (c, l)$ 
15:        else
16:           $D \leftarrow \hat{D} \cup (c, l)$ 
17:        end if
18:      end if
19:    end for
20:    for  $d = 1 : nd$  do
21:       $l \leftarrow \hat{Y}(d)$ 
22:      if  $(c, l) \notin \hat{S}$  &  $(c, l) \notin \hat{D}$  then  $t \leftarrow querySimilarity(c, l)$ 
23:        if  $t == 1$  then
24:           $S \leftarrow \hat{S} \cup (c, l)$ 
25:        else
26:           $D \leftarrow \hat{D} \cup (c, l)$ 
27:        end if
28:      end if
29:    end for
30:  end for
31: end procedure
```

Our proposed algorithm for online active metric learning is described in 2.

Algorithm 2 Online active metric learning

 1: **procedure** OAML

 Input: X, p, r, α

 number of cluster k

 N number of iterations

 $X_{n \times d}$ is input feature matrix

 r is ratio of near-center & boundary points to select

 p is number of pairs to select in each iteration

 α is alpha-trimming parameter

 Output: A learned metric

 Cluster belongingness matrix $C_{n \times k}$

 Initialization: $A \leftarrow I; S \leftarrow ; D \leftarrow$

 2: **for** $i = 1$ to N **do**

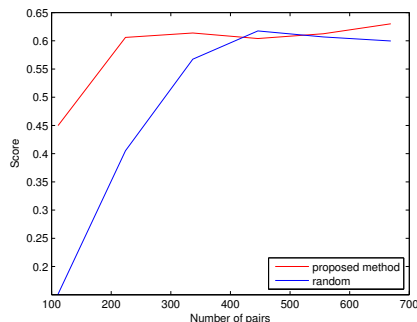
 $\hat{C} \leftarrow kmeans(X, k, A)$

 $[S, D] \leftarrow SELECTPAIRS(X, C, p, S, D, A, r, \alpha)$

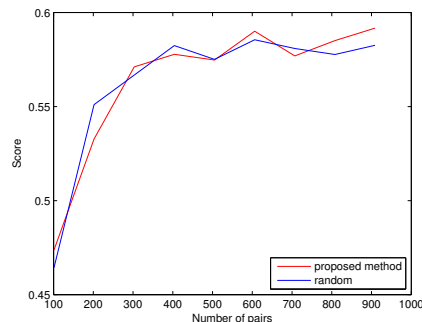
 Update matrix A based on new pairs using POLA, $A \leftarrow pola(A, S, D)$

 3: **end for**

 4: **end procedure**



(a) Silhouette score for Letter recognition dataset



(b) Silhouette score for Magic dataset

Figure 3.3: Results of proposed method.

In algorithm OAML 2 we iteratively select new points and use them to update the current metric using POLA, in each iteration our clusters becomes better by updating the metric.

3.3.3 Results

We have evaluated our procedure on Magic and Letter recognition datasets [18]. We divide each dataset randomly into two parts as learning and evaluation set in the ratio of 30% and 70% respectively. All the results presented are average over 5 runs.

We have used Silhouette measure to validate clustering which is given as $S = \frac{\sum_i s(i)}{n}$, where

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (3.12)$$

and $a(i)$ is average distance of point x_i with points in the same cluster as of x_i and $b(i)$ is lowest average distance to points which are not in same cluster as x_i . In each iteration we apply new pair selection procedure on learning set and update metric using new selected pairs, to evaluate the cluster

performance we cluster the evaluation set using k-means by using learned metric. To best of our knowledge no other method learn online metric using pairwise constraints for a clustering setup we have tested our method against selecting random query pairs with same experimental setup and we have found that our method performs better and with less number of query pairs.

From result we can notice that we can get same score eventually even by selecting random pairs but *label complexity* for our method is much less which what we aim here.

3.3.4 Summary

We proposed a heuristic method to actively select new pairs to update metric in online manner. From results we can see that on our proposed method select more informative points to update metric and can learn faster than selecting random points. In future we would like to improve this method by using a better approach with theoretical bounds.

Chapter 4

Conclusion and future work

We have contributed different methods where each of them solves some challenges related to metric learning for clustering. All the methods are generalized and does not depend on any particular type of data. Although we have produced some interesting results but we think there are places where our proposed methods can be improved.

4.0.5 Directions for Future Work

We have implemented Incremental Diffusion maps approach using very basic incremental SVD method, we believe results can be improved by using a better SVD update method. In Online Active Metric Learning we have proposed a heuristic based approach to select new pairs which may not work in all the cases, this can be improved by a methods like probability or entropy based methods which have a with better theoretical foundations.

Bibliography

- [1] Eric P Xing et al. “Distance metric learning with application to clustering with side-information”. In: *Advances in neural information processing systems*. 2002, pp. 505–512.
- [2] Fei Wang and Jimeng Sun. “Distance metric learning in data mining”. In: *SIAM SDM. Tutorials* (2012).
- [3] Joshua M Lewis et al. “Mapping uncharted waters: Exploratory analysis, visualization, and clustering of oceanographic data”. In: *Machine Learning and Applications, 2008. ICMLA '08. Seventh International Conference on*. IEEE. 2008, pp. 388–395.
- [4] Mikhail Belkin and Partha Niyogi. “Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering.” In: *NIPS*. Vol. 14. 2001, pp. 585–591.
- [5] Sam T Roweis and Lawrence K Saul. “Nonlinear dimensionality reduction by locally linear embedding”. In: *Science* 290.5500 (2000), pp. 2323–2326.
- [6] Ronald R Coifman and Stéphane Lafon. “Diffusion maps”. In: *Applied and computational harmonic analysis* 21.1 (2006), pp. 5–30.
- [7] Brian Kulis. “Metric learning: A survey”. In: *Foundations & Trends in Machine Learning* 5.4 (2012), pp. 287–364.
- [8] Peng Jia et al. “Incremental Laplacian eigenmaps by preserving adjacent information between data points”. In: *Pattern Recognition Letters* 30.16 (2009), pp. 1457–1463.
- [9] Shai Shalev-Shwartz, Yoram Singer, and Andrew Y Ng. “Online and batch learning of pseudo-metrics”. In: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 94.
- [10] Kilian Q Weinberger and Lawrence K Saul. “Distance metric learning for large margin nearest neighbor classification”. In: *The Journal of Machine Learning Research* 10 (2009), pp. 207–244.
- [11] Jason V Davis et al. “Information-theoretic metric learning”. In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 209–216.
- [12] Gautam Kunapuli and Jude Shavlik. “Mirror descent for metric learning: a unified approach”. In: *Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 859–874.
- [13] Ungar Lyle H. Gupta Abhishek A. Foster Dean P. “Unsupervised distance metric learning using predictability”. In: *ScholarlyCommons Technical Reports (CIS)* (2008).
- [14] Jiayan Jiang, Bo Wang, and Zhuowen Tu. “Unsupervised metric learning by self-smoothing operator”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 794–801.

- [15] Sandra Ebert, Mario Fritz, and Bernt Schiele. *Active metric learning for object recognition*. Springer, 2012.
- [16] Fei Wang et al. “Two heads better than one: Metric+ active learning and its applications for it service classification”. In: *Data Mining, 2009. ICDM’09. Ninth IEEE International Conference on*. IEEE. 2009, pp. 1022–1027.
- [17] Derya Birant and Alp Kut. “ST-DBSCAN: An algorithm for clustering spatial–temporal data”. In: *Data & Knowledge Engineering* 60.1 (2007), pp. 208–221.
- [18] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: <http://archive.ics.uci.edu/ml>.
- [19] Hongyuan Zha and Horst D Simon. “On updating problems in latent semantic indexing”. In: *SIAM Journal on Scientific Computing* 21.2 (1999), pp. 782–791.
- [20] Xilun Chen and K Selcuk Candan. “LWI-SVD: low-rank, windowed, incremental singular value decompositions on time-evolving data sets”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 987–996.
- [21] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Fourth Edition*. 4th. Academic Press, 2008. ISBN: 1597492728, 9781597492720.
- [22] Aurélien Bellet, Amaury Habrard, and Marc Sebban. “A survey on metric learning for feature vectors and structured data”. In: *arXiv preprint arXiv:1306.6709* (2013).