ISOMETRY AND CONVEXITY IN DIMENSIONALITY REDUCTION

A Thesis
Presented to
The Academic Faculty

by

Nikolaos Vasiloglou II

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2009

ISOMETRY AND CONVEXITY IN DIMENSIONALITY REDUCTION

Approved by:

Professor Anthony Yezzi, Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor David Anderson and Professor Alexander Gray, Advisor
School of Electrical and Computer Engineering/College of Computing
Georgia Institute of Technology

Professor Hongyuan Zha
College of Computing
Georgia Institute of Technology

Professor Ronald Schafer
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 23 January 2009

to my mother for buying my first book in physics

PREFACE

The size of data generated every year follows an exponential growth. The number of data points as well as the dimensions have increased dramatically the past 15 years. The gap between the demand from the industry in data processing and the solutions provided by the machine learning community is increasing. Despite the growth in memory and computational power, advanced statistical processing on the order of gigabytes is beyond any possibility. Most sophisticated Machine Learning algorithms require at least quadratic complexity. With the current computer model architecture, algorithms with higher complexity than linear $O(N)$ or $O(N \log N)$ are not considered practical.

Dimensionality reduction is a challenging problem in machine learning. Often data represented as multidimensional points happen to have high dimensionality. It turns out that the information they carry can be expressed with much less dimensions. Moreover the reduced dimensions of the data can have better interpretability than the original ones. There is a great variety of dimensionality reduction algorithms under the theory of Manifold Learning. Most of the methods such as Isomap, Local Linear Embedding, Local Tangent Space Alignment, Diffusion Maps etc. have been extensively studied under the framework of Kernel Principal Component Analysis (KPCA).

In this dissertation we study two current state of the art dimensionality reduction methods, Maximum Variance Unfolding (MVU) and Non-Negative Matrix Factorization (NMF). These two dimensionality reduction methods do not fit under the umbrella of Kernel PCA. MVU is cast as a Semidefinite Program, a modern convex nonlinear optimization algorithm, that offers more flexibility and power compared to

KPCA. Although MVU and NMF seem to be two disconnected problems, we show that there is a connection between them. Both are special cases of a general nonlinear factorization algorithm that we developed.

Two aspects of the algorithms are of particular interest: computational complexity and interpretability. In other words computational complexity answers the question of how fast we can find the best solution of MVU/NMF for large data volumes. Since we are dealing with optimization programs, we need to find the global optimum. Global optimum is strongly connected with the *convexity* of the problem. Interpretability is strongly connected with *local isometry*[1] that gives meaning in relationships between data points. Another aspect of interpretability is association of data with labeled information.

The contributions of this thesis are the following:

1. MVU is modified so that it can scale more efficient. Results are shown on 1 million speech datasets. Limitations of the method are highlighted.

2. An algorithm for fast computations for the furthest neighbors is presented for the first time in the literature.

3. Construction of optimal kernels for Kernel Density Estimation with modern convex programming is presented. For the first time we show that the Leave One Cross Validation (LOOCV) function is quasi-concave.

4. For the first time NMF is formulated as a convex optimization problem

5. An algorithm for the problem of Completely Positive Matrix Factorization is presented.

6. A hybrid algorithm of MVU and NMF the isoNMF is presented combining advantages of both methods.

[1] Preservation of local distances

7. The Isometric Separation Maps (ISM) a variation of MVU that contains classification information is presented.

8. Large scale nonlinear dimensional analysis on the TIMIT speech database is performed.

9. A general nonlinear factorization algorithm is presented based on sequential convex programming.

Despite the efforts to scale the proposed methods up to 1 million data points in reasonable time, the gap between the industrial demand and the current state of the art is still orders of magnitude wide.

ACKNOWLEDGEMENTS

The journey for completing this thesis has lasted 8 years, almost as long as they Odyssey. In this trip there was an army of people that supported me locally and globally. I made friends and I lost comrades. The emotional ups and downs made the whole experience interesting, stamping my life forever. It is hard for me to enumerate all these people that helped me and made this thesis possible and I acknowledge that each ones help was necessary for me to fulfill the requirements of a PhD. Going back 2 years, I remember myself being on the verge of dropping out, when my good friend Sourabh Ravindran introduced me to Alexander Gray. I want to thank Sourabh for all the good time we had as labmates and for his help, unfortunately we never had the chance to write a paper together. Alex holds the keystone of this thesis. As a young faculty he inspired a lot of "lost" students like me, blowing the spark of research hidden inside all of us. Very soon he convinced me to continue my PhD and finishing it although I was on the 6th year without any clue about what my thesis would look like. I feel more grateful to him for this wonderful collaboration and for his energy as a coach. Along with Alex, David Anderson and Ronald Schafer my advisors, always believed in me and they were always there to support me financially and morally, even if it took me 3 times to pass the prelim exam and more than two years to publish a paper. I feel privileged to be their student and I believe all 3 of them stand as model advisors. This PhD wouldn't have started at all if Petros Maragos in Greece hadn't endorsed me with his recommendation letters. I would always feel grateful to him for everything he gave me as an undergrad in Greece.

It has been an honor for me to be part of the FASTlab an amazing community of promising PhD students. Special thanks to Garry Boyer and Ryan Riegel for building

the core of the FASTlib library that was necessary for me and a lot of other students. Dongryeol Lee has been a great friend collaborator which I would also like to thank, for our long discussions and all the help he provided to me. Ravi Sastry a promising student is also another person I would like to thank for coauthoring papers together, he sort of reminds me of myself during the first years, but I belive he will find his Ithaca sooner than I did. Pari, Manyu, Hua and Bill (my son) have also provided a great atmosphere in the lab along with a lot of community work. On the other side of the highway, special thanks to Walter Huang who brought the recliner in the lab, making it possible for me to take quick naps and recharge my batteries. Brian, Ken, Tira, Rich, John, Mike and Shyam thank you too, for being great labmates. Professors Shapiro and Nemirovski provided significant mathematical guidance in the last phase of my PhD.

A big company of people balanced my academic life with their friendship. Symeon Nikolaou has been a great friend and I will always feel nostalgic to the nice moments we spent together. Georgios Georgoulas was another milestone in my Odyssey. Kelly Erinn Caine (the smartest PhD student at Georgia Tech) was one of the most interesting people I met at Georgia Tech. Even if I had failed getting my PhD, I would still have earned a great friend, a beautiful mind. Christos Gantsidis was the first one to drag me from electrical engineering to computer science starting my metamorphosis. Antonios Fornaro, Niko Vlantassis, Niko Athanasiadis, Niko Papageorgiou Christina Vlachou, George Stefopoulos, Vassilis Lakafosis, Vagelis Farantatos, Stathis Bacolas, Stathis Velenis, Domniki Assimaki , Lilla Zachou, Dimitris Anagnostou, Manos Tentzeris (for providing academic support the difficult moments), Manos Antonakakis, Nektarios Oreopoulos, Kostis Grigoriou, Efthimia Antonoudi, Maria Konte and Kostas Dovrolis. Among the greek fellows Stelios Kavadias was a big influence on me and he convinced me to repsect managment as a scince. Beyond that he and his family have been great friends. Crhis Markou

(and the family) along with Athan Sambanis offered a family atmosphere here in Atlanta. I have to thank both of them for the endless and uncountable grill sessions.

There are also some Georgia Tech employees that by doing their job right made my life easier here at Tech. Marilou Mycko in the academic office was always very good at making things work. Christy Ellis has been a model administrator and I think she is being rewarded for that. Finally Rosvelt Hardy the Janitor at TSRB has been the most hard working Georgia Tech employee, justifying his salary to the last nickel, by keeping the working environment clean.

On the west coast I want to thank Manos Pontikakis (for the lovely summer at Google), Alexandros Dimakis (for all he gave me at Berkeley), Shahid Chouldry (for opening the door to Google), Ayman Farahat, Harry Boukas, Angelos Stavrou (for teaching me C++), Mat Hans (for openning the door to HP, and for his advice to start programming, which I regret I didn't adopt earlier).

My Odyssey evolved simultaneously in both sides of the Atlantic Ocean. On the other side of the Atlantic my family and my friends contributed in many ways so that I can look back proud of what I accomplished these 8 years. My mother who bought me the first physics book and bought me the ticket to come in Atlanta is the first person I feel I owe the most, specially during 2004 where I got sick. My sisters Myrto, Areti and Ismini along with my aunt Lina have always been on my side taking care of me and making my summers in Greece unforgetable.

I have great repsect for my high school professors, Nikolaos Krasakis, Thanassis Mol, Kostas Mantzaris, Elisavet Kampani Despina Kalamaridou, Stavroula Resvani, who built my personality and believed in me since they first met me, They put the foundation and some of them had the opportunity to see my finishing what they started. Unfortunately some of them didn't make it and I consider them as the toll of this PhD. Kostas Mantzaris along with Yannis Tzortzis left us early during the first years of my PhD. Their absence filled me with sadness and I wish they could see me

graduating.

I am also grateful to HP Labs and Google for providing the grants to support me during all these years

At last I feel grateful to my wife Vicki for standing on my side during these years. I realize how difficult and painful it has been for her to see me switching from dissapointment and devastation to excitemnt so often.

TABLE OF CONTENTS

DEDICATION iii

PREFACE iv

ACKNOWLEDGEMENTS vii

LIST OF TABLES xv

LIST OF FIGURES xvi

SUMMARY xx

I INTRODUCTION 1

II LOCAL AND NON-LOCAL GEOMETRY AND SEMIDEFINITE PROGRAMMING 6

2.1 Fast computation of local and non-local neighborhoods 7

2.1.1 Kd-trees 7

2.1.2 Ball trees 8

2.1.3 Nearest Neighbor Algorithm 10

2.1.4 The k nearest neighbors algorithm 13

2.1.5 All nearest Neighbors and the dual-tree algorithm 14

2.1.6 Furthest neighbor and the dual tree algorithm 15

2.1.7 The k-in-between neighbor algorithm 17

2.2 Semidefinite Programming (SDP) 19

2.2.1 Convex programming and the semidefinite cone 19

2.2.2 Preserving distances 21

2.2.3 Ellipsoidal volumes and density preservation 22

2.2.4 Geometric means and leave one out cross validation 23

III MAXIMUM VARIANCE UNFOLDING, SCALABILITY AND EXTENSIONS 25

3.1 Maximum Variance Unfolding, the convex SDP case 27

3.2 The non-convex Maximum Variance Unfolding 29

3.3 Maximum Furthest Neighbor Unfolding 30
3.4 Stochastic Proximity Embedding 31
3.4.1 Evaluation of SPE 33
3.5 Implementation of MVU-like methods and Experiments 33
3.5.1 Implementation Issues of the Augmented Lagrangian and L-BFGS 34
3.5.2 Datasets 35
3.6 Extensions of MVU 37
3.6.1 Maximization of the Leave One Out Cross Validation (LOOCV) 37
3.6.2 Preserving volumes/densities instead of distances 38
3.7 Summary 40
IV NON-NEGATIVE MATRIX FACTORIZATION AS A SPECIAL CASE OF MVU. YET ANOTHER EMBEDDING PROBLEM 52
4.1 Convexity in Non-Negative Matrix Factorization under the positive completeness. 55
4.1.1 Solving the optimization problem of NMF. 57
4.2 Convex relaxations of the NMF problem. 58
4.2.1 A simple convex upper bound with Singular Value Decomposition. 58
4.2.2 Relaxation with a positive semidefinite cone. 59
4.2.3 Approximating the SDP cone with smaller ones. 61
4.2.4 NMF as a convex multi-objective problem. 63
4.2.5 Augmenting the relaxations with sparsity constraints 64
4.2.6 An algorithm for rank constrained problems 66
4.2.7 Experiments 68
4.2.8 An algorithm for solving the problem of Completely Positive Factorization 69
4.3 Global and local solutions of non-convex NMF. 71
4.3.1 NMF as a Generalized Geometric Program and its Global Optimum. 72
4.4 Isometric NMF. 75

4.4.1 Convex isoNMF. 76
4.4.2 Non-convex formulation of isoNMF. 76
4.5 Experimental Results 78
4.6 Summary 79
V LEARNING ISOMETRIC SEPARATION MAPS 85
5.1 Isometric Separation Maps (ISM) 87
5.2 Dimensionality Minimization with ISM 89
5.3 Transductive SVMs 90
5.4 Summary 92
VI NONLINEAR MATRIX FACTORIZATIONS, A GENERAL FRAMEWORK FOR DIMENSIONALITY REDUCTION 102
6.1 Matrix factorizations and dimensionality reduction 103
6.2 Casting MVU, NMF and more as a general rank-constrained semidefinite program 104
6.2.1 MVU as a special case 106
6.2.2 LOOCV unfolding as a special case 108
6.2.3 NMF as a special case 109
6.3 Extending to factorizations with any nonlinear dot product or even divergence 111
6.4 Moving even further, automatic construction of the nonlinear operator g 112
6.5 Summary 113
VII DIMENSIONALITY REDUCTION OF LARGE SPEECH CORPORA . 115
7.1 MFCC Features 117
7.2 Principal Component Analysis 117
7.3 Experiments 118
7.3.1 Preliminary experiments 118
7.3.2 300K points experiments 119
7.3.3 1 Million points experiments 120
7.3.4 Weaknesses, limitations of MVU/MFNU 121

7.3.5 Future work 122
7.4 Summary 123
VIII CONCLUSION 130
8.1 Directions for future research 131
IX AUTHOR'S PUBLICATIONS 132
REFERENCES 135
VITA 142

LIST OF TABLES

1 Dataset description 35

2 Classic NMF, the relative root mean square error, sparsity and distance error for the four different datasets (cbcl normalized and plain, statue and orl) 79

3 isoNMF, the relative root mean square error, sparsity and distance error for the four different datasets (cbcl normalized and plain, statue and orl) 79

4 ISM SVM classification score versus k-neighborhood for the First Experiment 93

5 Traditional SVM Classification Score versus k-neighborhood 93

6 ISM SVM Classification Score versus k-neighborhood For the Whole Dataset 94

7 ISM SVM Classification Score versus k-neighborhood For the Whole Dataset 94

LIST OF FIGURES

1 The parent node of a 2 dimensional kd-tree 9

2 A two node 2 dimensional kd-tree 9

3 Nearest neighbor with pruning 12

4 True nearest neighbor is out of the leaf 12

5 Kd tree with no possible pruning 13

6 A two dimensional pathological kd-tree 13

7 In this case the partition is very bad and almost for every point pruning is not feasible 14

8 The query node in red after top down recursion ends up in a leaf of the reference tree. Then every point in the query node (red) finds with the naive method its candidate nearest neighbor. Then for all of them we compare all the candidate nearest distances and find the maximum r_{max}. Now we know that if there is any node in distance greater than r_{max} there is no point in checking for candidate nearest neighbors. As we see in the right Figure the dashed box doesn't intersect with the bounding box of the leaf. This means that we can prune the yellow boxes. 16

9 A two dimensional kd-tree 17

10 Simulation of the dual tree algorithm 17

11 Pseudo-code for the dual-tree all nearest neighbor algorithm 18

12 Pseudo-code for the dual-tree all furthest neighbor algorithm 19

13 Pseudo-code for the in-between-k-neighbor. The `RemoveExtraFurthestNeighbors()` removes the extra m extra neighbors by finding the $m+1$ furthest neighbor. m is typically small, less than the maximum number of points in a leaf. 20

14 MVU maximizes the distances of points from the origin 30

15 MFNU maximizes distances beteen furthest neighbors. As we can see points get pushed from different directions. This fact helps avoiding local minima 31

16 The Stochastic Proximity Embedding Algorithm 32

17 The classic MVU algorithm a)Scaling performance b)Iterations required for the optimization 42

18 Convex MVU vs non-convex MFNU. For the convex MVU we run experiments up to 600 points and then extrapolated 43

19 The MFNU performance, a) Scaling of the MFNU b)Maximization results of the Maximum Furthest Neighbors objective c) Iterations required for the optimization d) Number of constrained kN (solid line), consolidated constraints (dashed line) 44

20 The MFNU algorithm with auto-tuning of k-neighborhoods, a)Scaling of the algorithm b)Iterations required for the optimization c)Number of constrained kN (solid line), consolidated constraints (dashed line) . 45

21 Unfolded Swiss rolls 10K, 20K, 40K (top to bottom row), (left column) MFNU, (center column)MFNU with auto-tuning for k-neighborhoods, (right column)MVU. All images have been sampled showing only 4000 points, for visual clarity 46

22 (left column) corel color moments, (right column) corel color histogram, (a)4-point neighborhood, (b)5-point neighborhood, (c)7-point neighborhood 47

23 The 3 first components of the Max LOOCV unfolding 256-point swiss roll, along with the eigenvalue spectrum. In this experiment a k=5 neighborhood is preserved. 48

24 The 3 first components of the Max LOOCV unfolding 512-point swiss roll, along with the eigenvalue spectrum. In this experiment a k=5 neighborhood is preserved. 49

25 Top: The first 3 components of the DPM for a 400-point swiss roll, Bottom: The first 3 components of Kernel PCA. In this experiment a k=4 neighborhood is preserved. The bandwidth of the Gaussian is medium to small. 50

26 Top: The spectrum of DPM and Kernel PCA for small bandwidth. Bottom: The same spectra for large bandwidth. 51

27 3 rice seeds original gray scale image 69

28 The 6 components of rank reduction convex NMF 70

29 The 6 components of non-convex NMF 71

30 The 3 rice seeds image reconstructed from the 6 NMF components acquired with the rank reduction convex NMF 72

31 The 3 rice seeds image reconstructed from the 6 NMF components acquired with non-convex NMF 73

32 The singular values of the 3 rice seeds 74

33 The six first svd components of the 3 rice seeds 75

34 (a)Some images from the cbcl face database (b)The same images after variance normalization, mean set to 0.25 and thresholding in the interval [0,1] (c)The synthetic statue dataset from the isomap website [40] (d)472 images from the orl faces database [42] 80

35 Top row: 49 Classic NMF prototype images. Bottom row: 49 isoNMF prototype images (a, c) CBCL-face database with mean variance normalization and thresholding, (b, d) CBCL face database without preprocessing. 81

36 Top row: 49 Classic NMF prototype images. Bottom row: 49 isoNMF prototype images (a, c) statue database , (b, d) orl-faces database . . 82

37 Scatter plots of two largest components of classic NMF(in blue) and Isometric NMF(in red) for (a)cbcl faces (b)isomap faces (c)orl faces . 83

38 In this set of figures we show the spectrum of classic NMF (solid line) and Isometric NMF (dashed line) for the three datasets (a)cbcl face (b)isomap statue (c)orl faces. Although isoNMF gives much more compact spectrum we have to point that the basis functions are not orthogonal, so this Figure is not comparable to SVD type spectrums . 84

39 a)A three dimensional swiss roll painted with color gradient. b)The same swiss roll with two classes on it, black and green c)Unfolded swiss roll (a) with MVU/MFNU (no class information). The color gradient shows that local distances has been preserved. d) Unfolded swiss roll (b) with MVU/MFNU. The two classes are not linearly separable. e,f) Views of the swiss roll (a) with ISM. The class structure was taken from (b). The intension of this Figure is to show how the points are mapped so that the local neighborhoods are preserved. g,h)Views of the (b) manifold after ISM. Now points are painted with the class colors to show that they are linearly separable 96

40 Top: Three classes laying on a swiss roll. Bottom: After unfolding them with MVU the classes are not linearly separable. Isometric Separation Maps managed to map this manifold in a 12-dimensional space such that the classes were linearly separable by 3 hyperplanes 100% of the time and the 5-neighborhood distances were preserved with 0.1% relative root mean square error 97

41 In this Figure we illustrate the PCA (SVD) spectrum of the unfolded swiss roll of Figure 40. As we can see it is pretty rich. 98

42 Top: Three classes laying randomly on a swiss roll. Bottom: After unfolding them with MVU the classes are not linearly separable. Isometric Separation Maps managed to map this manifold in a 12-dimensional space such that the classes were linearly separable by 3 hyperplanes. The optimization algorithm terminated with feasibility error 0.4% for 5-neighborhood distance preservation, while 99.83% of the points were correctly classified. The goal of this experiment was to verify experimentally that ISM can lift any strange dataset to a high dimensional space, such that classes are linearly separable 99

43 In this Figure we illustrate the PCA (SVD) spectrum of the unfolded swiss roll of Figure 42. As we can see it is pretty rich. Despite the bad structure of the classes, the ISM algorithm was able to map it on a 12 dimensional space. 100

44 (a) A trivial one dimensional manifold (b) The separation hyperplane from an SVM with a gaussian kernel (c) ISM will do a homeomorphic transformation on the manifold so that a linear hyperplane can do perfect separation 101

45 Nonlinear Convex Factorization 106

46 122

47 (a) The magnitude of the principal components for the whole TIMIT dataset (b) Cumulative percentage energy of the eigenvalues. 124

48 (a) The magnitude of the principal components for the 20,000 TIMIT dataset (with variance normalization) (b) Cumulative percentage energy of the eigenvalues. 125

49 (a) Embedding of the 100,000 TIMIT dataset (without variance normalization) with the MFNU method. (b) Embedding of the same dataset with PCA 126

50 Left: PCA of the 39 dimensional TIMIT datapoints, Right: PCA spectrum of the 15 dimensional unfolded 127

51 Left:PDFs of the 39 dimensional TIMIT data points, Right: PDFs of the of the 15 dimensional Unfolded 128

52 Left:PDFs of the 39 dimensional TIMIT data points for *uh*, Right: PDFs of the of the 15 dimensional Unfolded *uh* 128

53 Right:PDFs of the 39 dimensional TIMIT data points for *sh*, Left: PDFs of the of the 15 dimensional Unfolded *sh* 129

SUMMARY

In this dissertation we address the problem of dimensionality reduction for large datasets based on the algorithms of Maximum Variance Unfolding (MVU) and Non-Negative Matrix Factorization (NMF). In chapter 1 we give a short review of the existing dimensionality reduction methods. Following, in chapter 2 we introduce efficient multidimensional structures for computing near and distant neighborhoods between points, along with some new algorithms for computing furthest neighbors. In the same chapter we give an overview of Semidefinite Programming (SDP) and show how geometric and statistical properties of data can be SDP representable. In chapter 3 an extensive computational analysis of MVU is given. Techniques for scaling MVU are shown. We verify experimentally a novel variation called Maximum Furthest Neighbor Unfolding (MFNU) that shows very good performance. In the same chapter extensions of MVU are presented based on the Kernel Density Estimation of the data. Next in chapter 4 an extensive study of NMF as a global optimization problem is presented. For the first time in the literature we show that NMF can be cast as a convex program. An extension of NMF with local isometric constraints, the isoNMF, is introduced and compared to NMF. In chapter 5 Isometric Separation Maps (ISM) are introduced as a dimensionality reduction tool. ISM supplements MVU with classification information, leading to a transductive SVM. Comparisons with traditional SVMs are illustrated showing comparable performance. In chapter 6 we present a unified framework for dimensionality reduction as a nonlinear factorization problem. We show that all the algorithms presented in this dissertation along with many others more complex can be cast as a rank constraint problem, that its local solution can be found with a sequence of convex programs. Although the algorithm

presented is of polynomial complexity it is not yet scalable. Finally in chapter 7 MVU is applied on 1 million speech data points, revealing its strengths and the weaknesses.

CHAPTER I

INTRODUCTION

Most of the time data is represented as $m-$dimensional Euclidean points where m might be high. For example speech waveforms are converted to $m-$dimensional vectors through Fast Fourier Transform or Linear Prediction Coefficients or Mel Frequency Cepstrum Coefficients MFCC [70]. m can typically range from 40 to a few hundreds. Another example of high dimensional data appears in the user-movie rating data. Assume N users that can possibly rate m movies. This information can be represented in an $N \times m$ matrix. In other words any user can be represented with an $m-$dimensional vector. The dimensionality of the data m is also known as *extrinsic* dimensionality. In many cases the extrinsic dimensionality of the data is significantly large. Often though, dimensions are nonlinearly correlated. That means the original information included in m dimensions can be represented with $k < m$ dimensions. In the literature k is referred as the *intrinsic* dimensionality. Discovering the dependance between dimensions not only leads to compression of the data, but also extracts information. Several methods have been proposed for discovering relationships between dimensions. They are divided in two categories, those that assume linear dependance and those ones that do not. The first category is very well represented by Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) [31]. On the other side a great variety of non-linear dimensionality reduction methods has been developed in the last 10 years. The most important ones are the following:

1. Isomap [89]
2. Locally Linear Embedding (LLE) [74]
3. Laplacian EigenMaps (LE) [5]
4. Hessian EigenMaps (HE) [20]
5. Local Tangent Space Alignment (LTSA) [101]

6. Diffusion Maps (DM) [17]

7. Maximum Variance Unfolding (MVU) [97].

8. Non-Negative Matrix Factorization (NMF) [55]

In [30] authors showed that most of these methods along with Multidimensional Scaling (MDS) [49, 50] are a special case of Kernel PCA [75, 10]. Each of the above methods defines a data-dependant kernel that emphasizes different properties of the data. All of the above methods are based on preserving some properties defined on a local neighborhood, with the exception of MDS that in its original form was based on all pair distances. Once the Kernel matrix is built, the embedding can be retrieved from the eigenvectors. Isomap extends MDS by building the Kernel matrix based on geodesic distances rather than the Euclidean ones. Laplacian EigenMaps is based on the Laplacian of the graph based on the Gaussian kernel. It turns out that the Laplacian of the graph is a positive semidefinite kernel matrix. The Laplacian Eigenmaps developed almost simultaneously with the Diffusion Maps are based on the diffusion kernel. Instead of computing the geodesic distances based on the shortest path on the Euclidean path it computes the time a diffusion process needs to reach from point A to point B. Diffusion maps in general squeezes local distances and increases distances between points that are far away. These techniques deform the manifold and are ideal for classification. Maximum Variance Unfolding solves a semidefinite program that tries to spread points in space while constraining local distances. The method gives similar results with Isomap. The only fundamental difference is that MVU doesn't explicitly compute geodesic distances. In general MVU gives better embedding than Isomap [96]. This is because Isomap computes geodesic distances approximately causing some deformation that often leads to slightly higher dimensional embedding compared with MVU. Both methods behave better than the others when it comes to intrinsic dimensionality estimation as studied in [97, 57].

LLE tries to reconstruct points from its neighbors. LLE can also be interpreted as a diffusion between points allowing negative and positive reactions , in contrast to Diffusion Maps. Hessian LLE is a blend of LLE and Laplacian EigenMaps. The basic advantage over LLE is that it gets rid of the convexity constraint of LLE. LTSA tries to reconstruct the tangent space locally on every point and then aligns them with affine transformations in order to reconstruct the geodesic distances in a Euclidean space. LTSA has the same goal with ISOMAP, but provides more rigorous error analysis. In many applications, it makes more sense to preserve ranks between distances rather than the exact distances. This idea was first introduced as a variation of MDS. Although the problem was cast as a non-convex problem, in [1] it was posed as a semidefinite one. In [66] a simpler formulation of the problem as a quadratic problem was presented achieving superior results. Moreover, the method provided a solution for out-of-sample extension. The common thing about all the above methods compared to traditional kernels such as the Gaussian is that each element of the kernel matrix K_{ij} does not depend on the inputs x_i, x_j but on all the other training data.

All of the above methods have their strengths and their weaknesses,. MVU, though, seems to give the most compact embedding[1] compared to other methods at least for toy data sets [97, 57], but there are still cases that it fails [78]. Moreover the comparisons are not extensive so it would be wrong to claim that MVU is the best method. Unlike MVU all the other methods listed above have already been extensively studied and unified under the framework of kernel PCA [75, 10], but NMF and MVU have not. This motivates us to unify all of them under a general dimensionality reduction algorithm. Another challenge is MVU's polynomial complexity due to the underlying semidefinite program. In this dissertation we scale MVU up to million point datasets. We investigate whether it is possible to preserve the convex nature of

[1]Experiments have shown that it tends to recover the intrinsic dimension of manifolds for which the intrinsic dimension is known. With the term compact we mean that the svd spectrum of the embedded dimensions is very concentrated

the algorithm and the same time scale it.

Some applications require that the data are non-negative, so any nonlinear transformation on the dimensions should also give non-negative components. In [55] authors present a method that reduces the dimensionality of the data so that non-negativity is preserved. The method provides sparse representation of the data and in some cases it projects the data in vectors that are meaningful. For example when NMF is applied in a set of images it extracts parts of the images that are meaningful [55]. A major drawback of the algorithm is the underlying non-convex problem. In this chapter 4 we present convex relaxations of NMF along with an algorithm that finds the global optimum. In this dissertation we investigate if a convex formulation of the problem exists. Convexity is a desirable property in machine learning optimization problems, because it guarantees global solutions. Another challenge also addressed is whether NMF and MVU can be combined together and give an algorithm that shares the properties of both.

CHAPTER II

LOCAL AND NON-LOCAL GEOMETRY AND SEMIDEFINITE PROGRAMMING

All nonlinear dimensionality projection methods mentioned in chapter 1 share a common step. Local nearest neighbors have to be computed for every point. The local geometry can be expressed in terms of dot products, distances, volumes and densities. Preservation of local geometry and deformation of the non-local geometry leads to different dimensionality reduction methods. In this chapter we show how local and non-local geometry can be computed in a fast and scalable way. Moreover we show how geometry can be expressed with convex constraints with the help of the recently developed theory for semidefinite programming [64, 90, 99].

2.1 Fast computation of local and non-local neighborhoods

In this section we discuss the state of the art method for computing all k-nearest neighbor method and we also introduce a variant of this method for computing furthest neighbors. We introduce multidimensional trees, a hierarchical partition of a dataset so that data points are grouped close to their neighbors. It has been proven that there is not a globally optimal partition strategy [28] and it is heavily dependant on the distribution of the data points as well as on the intrinsic dimensionality. In the next section we present a particular class of multidimensional trees, the binary trees and more specifically kd-trees and ball (metric) trees. The key concept in binary trees is partitioning a set recursively in two sets at every step.

2.1.1 Kd-trees

The first multidimensional tree in the literature was the kd-tree [8]. After 30 years it still performs comparably and in some cases, even better than modern trees. At every level a partition (pivot) dimension is chosen according to a user defined criterion and the data is split in two subsets according to this dimension. Kd-trees are off-line trees. In order to build a kd-tree all the data must be available before building the tree in contrast to other trees that can dynamically add or delete points.

Several strategies have been recommended for finding the splitting dimension.

The most popular ones are splitting on the dimension with the maximum variance or maximum range. However both of these strategies are heuristics and there is no underlying theory behind them that guarantees better performance of the trees. Another issue once the splitting dimension is found, is the actual numeric value of the split value. In other words, the ratio of the points going in the two resulting sets. The mid value of the range or the mid value of the variance will in general give two sets that in most of the cases will have different number of points. Another strategy is to find the value that gives equally sized sets. This strategy in general gives balanced trees, while the first one leads to unbalanced trees. In the following sections we show that empirically it is preferable to have unbalanced splits on the tree, since it favors pruning in the process of finding the nearest neighbors.

Data structures for kd-trees Kd-trees consist of two types of data structures, nodes and leafs. The basic data needed for the nodes are the hyper-rectangles, Figure 1. Hyper-rectangles are the k-dimensional rectangles that describe the minimum bounding box for a given set of points. Leafs hold the data points, Figure 2. The number of points on a leaf is user defined. The tree method for finding the nearest neighbor is faster than the naive only after a threshold of points. This threshold is used as the maximum number of points per leaf. Bounding boxes (Hyper-rectangles) and points are the minimum amount of information required to be stored on nodes on the leaf. Extra information about the data, such as the centroid of the box, higher order moments, etc., also called Cached Statistics can speed up tremendously algorithms other than nearest neighbor. The algorithm of nearest neighbor search with kd-trees will be described in the following section (see Figure 3).

2.1.2 Ball trees

One of the main disadvantage of kd-trees is that they can only handle problems associated with the Euclidian metric, or metrics that can be cast in terms of the

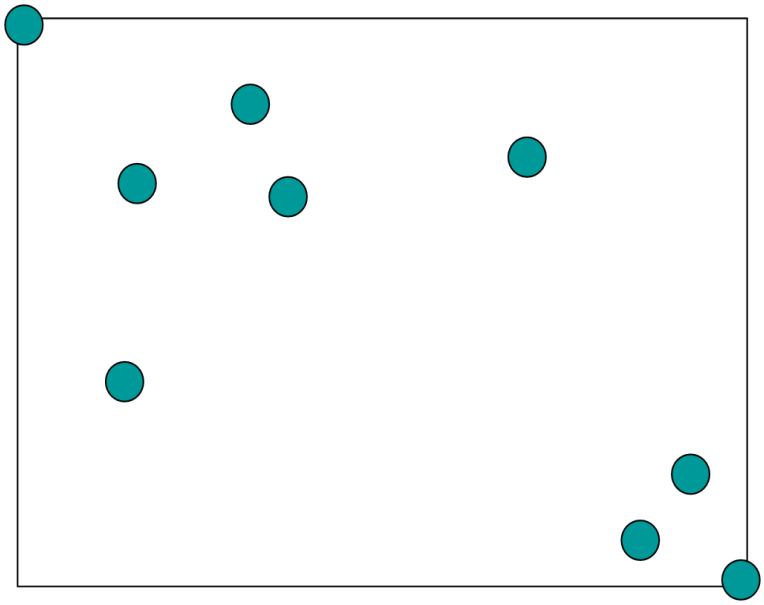


Figure 1: The parent node of a 2 dimensional kd-tree

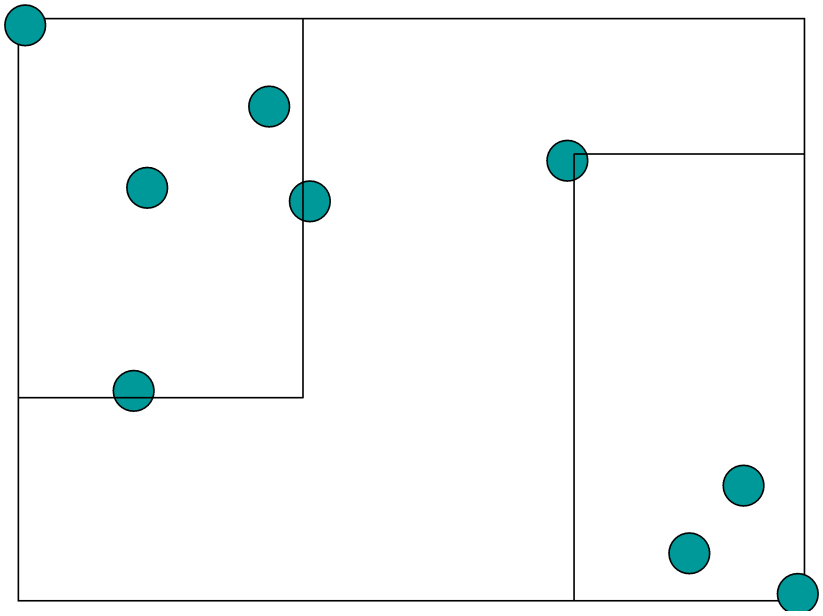


Figure 2: A two node 2 dimensional kd-tree

Euclidian metric such as the Hamming distance. In many cases, though, the data is expressed in cartesian coordinates but similarity cannot be expressed with the euclidian metric. For example the cosine distance is used in many applications of machine learning such as document retrieval.

Metric or Ball trees [63] provide a solution to this problem. Instead of using hyper-rectangles, they use hyper-spheres. Hyper-spheres are generalized k-dimensional spheres:

$$x, x_0 \in \Re^k, d(x, x_0) = r, r \in \Re \tag{1.1}$$

where $d(x, y)$ defines a distance metric in a k dimensional space. At every split two points that are far away are chosen as anchor points. Every other point of the initial set is classified according to the minimum distance from the anchor points. The choice of anchor points is done heuristically. One method that works in practice is choosing a point randomly and then find the furthest point. This is the first anchor point. The next anchor point is chosen as the furthest point from the first.

2.1.3 Nearest Neighbor Algorithm

The purpose of using a tree in the nearest neighbor problem is to gradually find bounds for the neighbor distance and prune parts of the search space. In general trees give bounds for data partitions that can be used in other problems.

Initially we start the search from the root node. At each step we recurse down to the child node that contains the query point. In kd-trees we check the pivot dimension of the query point and if it is smaller than the pivot value we recurse on the left child otherwise on the right. In ball trees we recurse to the child that is closer to the query point. The recursion continues until we reach a leaf. Then the distances of the query point and the points on the leaf are computed. The closest point is a candidate nearest neighbor Figure 3. Moreover we know that the actual nearest neighbor can not be further than the current nearest neighbor. So in the end of the top down recursion we have an upper bound for the nearest neighbor. Note that the top down recursion is logarithmic to the number of points assuming a balanced tree. The next step is to backtrack. At each step in backtracking we check if the candidate nearest neighbor is in the current leaf/node. There are two ways to check that. First of all

when we are on a leaf we check whether the hyper-sphere centered around the query point with radius equal to the distance between the query point and the candidate nearest neighbor d_{min} intersects the bounding box, Figure 4 5. If not then this means it is the nearest neighbor. If it does, then we have to check all the bounding boxes that intersect the hyper-sphere. For kd-trees and ball trees it is fairly easy to detect that. The algorithm is outlined below:

1. Recurse to the leaf that contains the point. To do that compare at every node the split value of the splitting dimension of the point and recurse to the right direction.

2. On the leaf compute the nearest neighbor with the naive method. The distance to the nearest point on the leaf (candidate nearest neighbor) is the upper bound for the nearest neighbor distance r_{max}, Figure 3.

3. If the sphere centered on the query point and radius r_{max} doesn't intersect the bounding box terminate. The candidate nearest neighbor is the actual nearest neighbor.

4. If not, then backtrack and visit all subtrees recursively to the leaf and update the candidate nearest neighbor if you find a point that is closer than r_{max}, Figure 5.

5. Every time you backtrack check if the sphere centered on the query point with the radius r_{max} doesn't intersect the bounding box. If it doesn't, then terminate otherwise continue.

In general for natural datasets where data tend to cluster, it is possible to heavily prune and achieve logarithmic performance in search. In some cases when the partitioning is not very good, trees behave poorly (Figure 6, 7) and eventually have

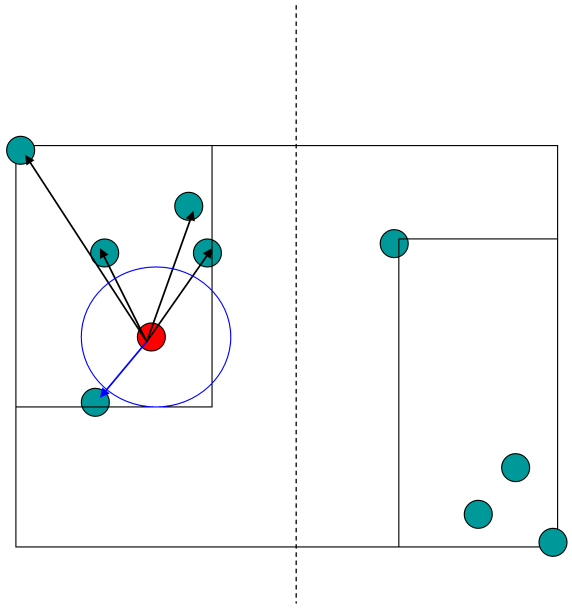


Figure 3: Nearest neighbor with pruning

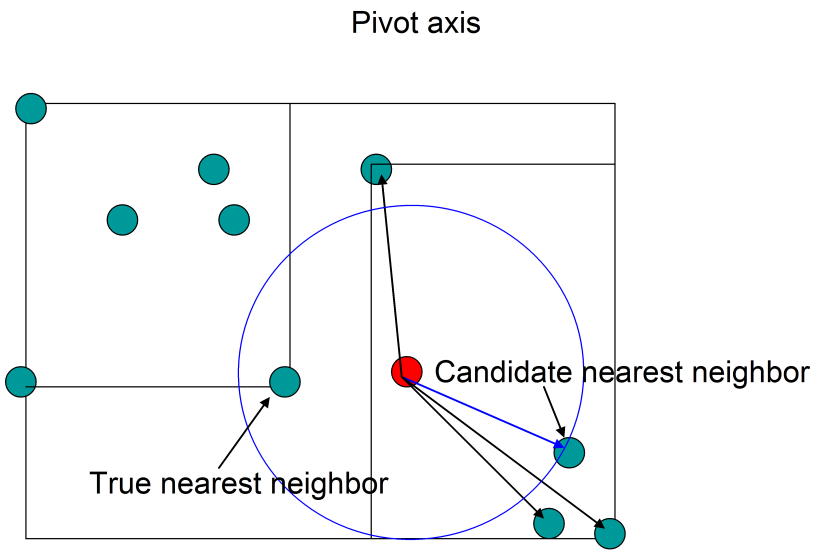


Figure 4: True nearest neighbor is out of the leaf

to search almost the whole data-set and they behave worse than the naive method since they have extra recursion overhead.

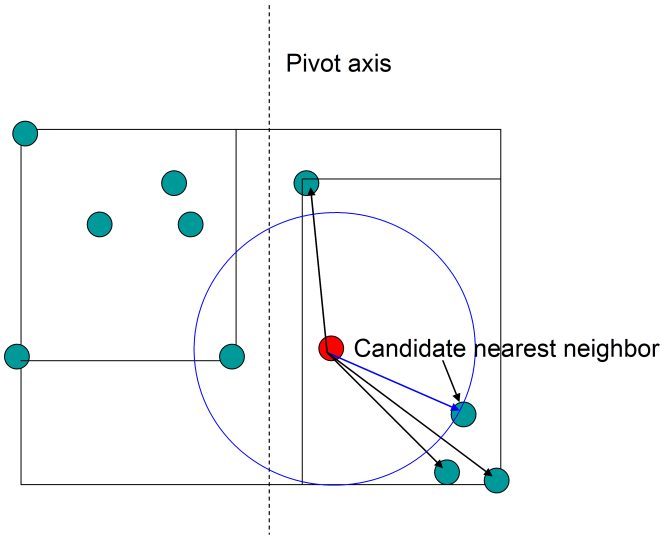


Figure 5: Kd tree with no possible pruning

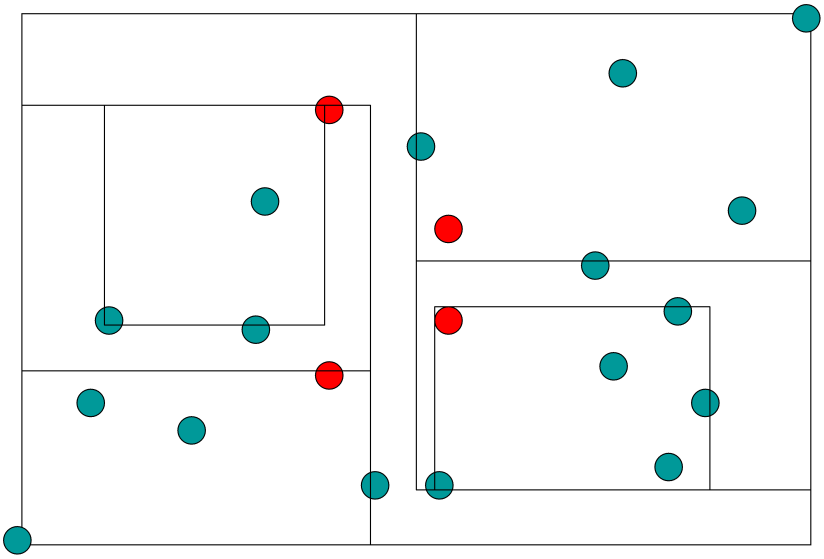


Figure 6: A two dimensional pathological kd-tree

2.1.4 The k nearest neighbors algorithm

In machine learning it is very rare that only the nearest neighbor is necessary. In practice we need more than one, we need k. The k nearest algorithm differs from the one described above in step 2. When hitting a leaf we compare the query point with

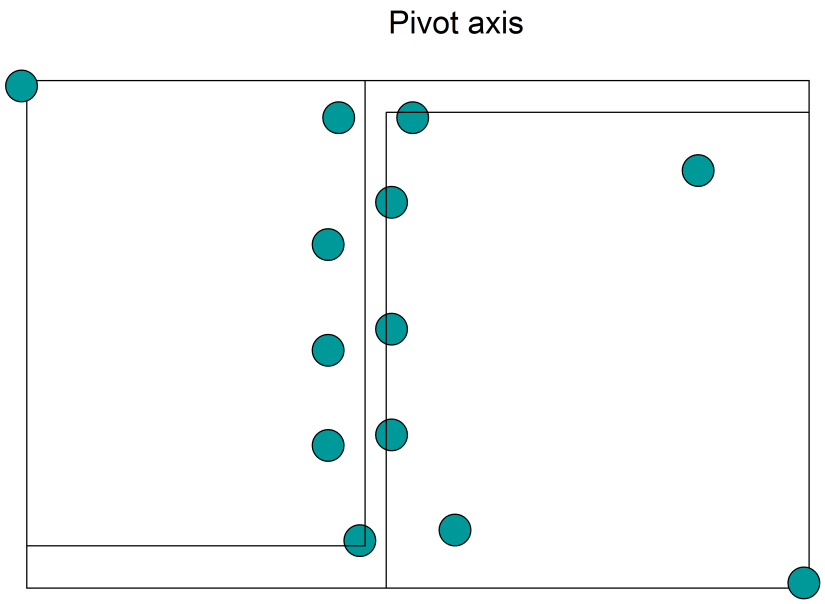


Figure 7: In this case the partition is very bad and almost for every point pruning is not feasible

every reference point and we keep a sorted list with the k_{th} nearest neighbors. Every time the algorithm visits a leaf it updates the list and it always keeps the k_{th} nearest neighbors. In order to be able to prune, r_{max} is always set to the distance of the k_{th} neighbor computed so far.

2.1.5 All nearest Neighbors and the dual-tree algorithm

The problem of computing all nearest neighbors appears often in many machine learning problems, usually in the training phase. As we will see in the following sections the formation of the kernel matrix involves all nearest neighbor computations. The all-nearest neighbor problem is a special case of a more general class of problems called N-body problems [27].

Having in mind the tree based algorithm, it is natural to assume that finding all nearest neighbors can be solved by executing N nearest neighbor queries resulting in an $O(N \log N)$ complexity. We will refer to this as the single tree all nearest neighbor algorithm. However, it is not difficult to notice that there are redundant computations. For example, consider a query set stored on a kd-tree and a reference

set stored on another kd-tree as well. If we do the top down recursion for a point then we notice that most likely this top down recursion might be exactly the same for all the points in the same query leaf. It is also possible that the top down recursion might be the same for a whole query subtree (see Figure 8).

This observation leads us to the conclusion that instead of comparing distances between points in the search it is more useful to compare bounding boxes. The algorithm is a four-way recursion Figures 9, 10 11,. In general, the dual tree algorithm as we call it since the query and the reference set lie on trees (they might share the same tree if the query and the reference set are the same), gives linear complexity over the number of data. This is empirical complexity. In Figure 8 we give an example of dual-tree pruning.

2.1.6 Furthest neighbor and the dual tree algorithm

The problem of finding the nearest neighbors has many applications in different areas such as statistics, machine learning, physics simulation, etc. In chapter 3, we show that distances between furthest "neighbors" can also be useful in manifold learning. It is not the first time that furthest neighbor is used, it is a very common term in computational geometry [26]. Although it does not make sense to call "neighbors" points that they are far away, we will still call furthest neighbor of point x the point y that its distance from x is maximized. Finding the furthest neighbor with the naive method has the same complexity with finding the nearest neighbor. Use of multidimensional trees can speed up the computation the same way it does with the nearest neighbor. In the top down recursion the algorithm always chooses the node that is further away. At the end of the top down recursion, the exhaustive search finds the point that has the maximum distance with the query point. The point found is a candidate furthest neighbor and its distance from the query point is less or equal to the true furthest neighbor. In the back tracking steps (bottom up recursion)

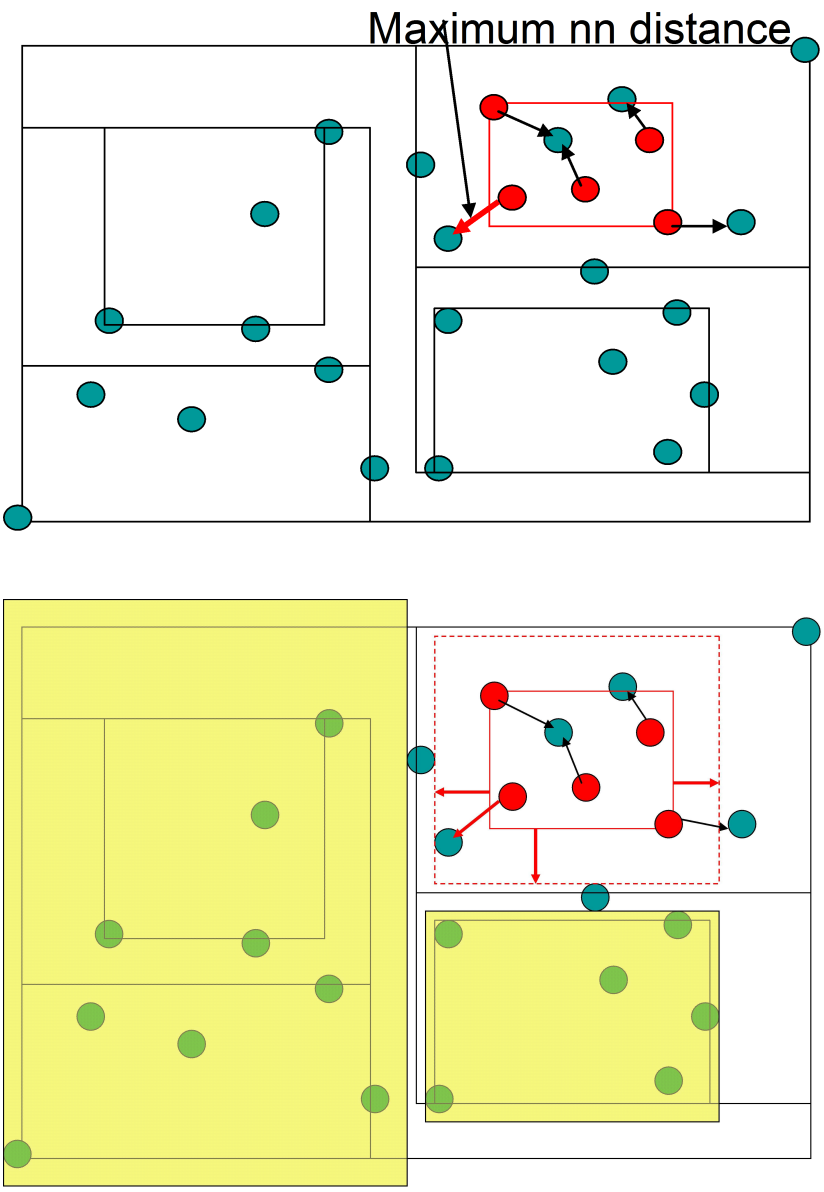


Figure 8: The query node in red after top down recursion ends up in a leaf of the reference tree. Then every point in the query node (red) finds with the naive method its candidate nearest neighbor. Then for all of them we compare all the candidate nearest distances and find the maximum r_{max}. Now we know that if there is any node in distance greater than r_{max} there is no point in checking for candidate nearest neighbors. As we see in the right Figure the dashed box doesn't intersect with the bounding box of the leaf. This means that we can prune the yellow boxes.

nodes that are closer than the candidate furthest neighbor are pruned. The algorithm just described is a single tree furthest neighbor one. It is very simple to apply the framework of the dual-tree algorithm for furthest neighbor computation. In Figure 12

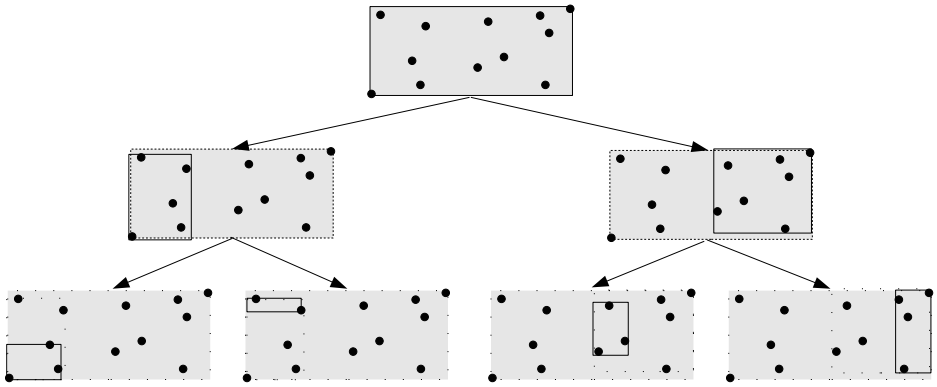


Figure 9: A two dimensional kd-tree

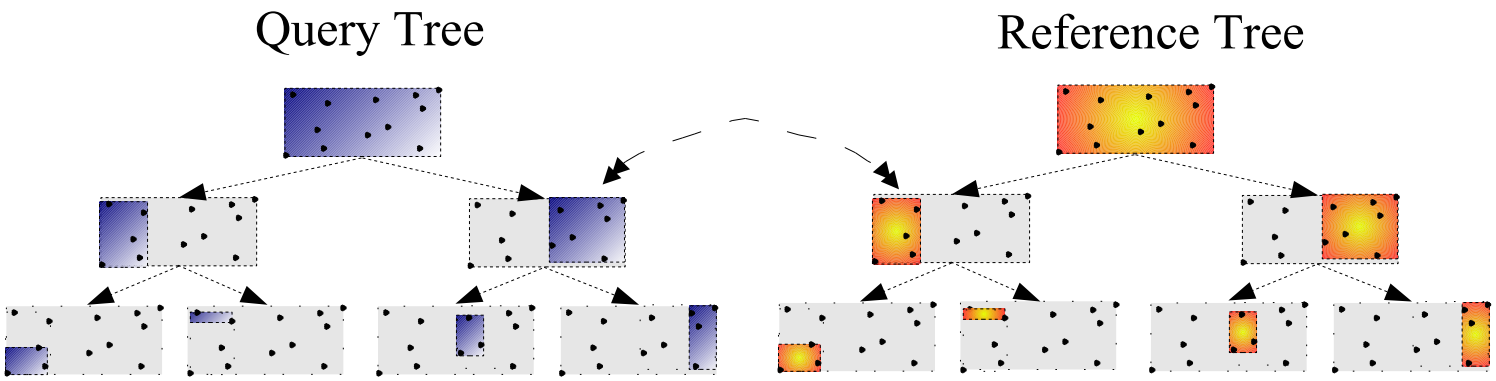


Figure 10: Simulation of the dual tree algorithm

we show the pseudo code. This algorithm can compute all pair furthest distances very fast, speeding up the Gonzalez algorithm mentioned above and improving MVU's performance as shown in chapter 3.

2.1.7 The k-in-between neighbor algorithm

We already presented algorithms for computing k-nearest/furthest neighbors, for fairly small k. There are applications where the k_{th} neighbor is essential but not the preceding ones. If k is large (fraction of N) then the algorithm described in section 2.1.4 is slow as it needs to keep a sorted list of k neighbors. Every time the algorithm visits a leaf that has m points the algorithm has to sort a list of $k + m$ point distances and keep only the k first elements. If $k >> m$ sorting should not be very complex, since the list is sorted before the insertion of the new m distances.

```
recurse(q : KdTree, r : KdTree) {
  if (max_nearest_neighbor_distance_in_node(q)
        < distance(q, r) {
     /* prune */
  } else if (IsLeaf(q)==true and IsLeaf(r)==true) {
    /* search for every point in q node its
    /* nearest neighbor in the r node */
    /* at leaves we must resort to     */
    /* exhaustive search O(n^2) */
    /*update the maximum_nearest_distance_in_node(q)*/
  } else if (IsLeaf(q)==false and IsLeaf(r)=true {
    /*choose the child that is closest to r */
    /* and recurse first */
    recurse(closest(r, q.left, q.right), r)
    recurse(furthest(r, q.left, q.right), r)
  } else if (IsLeaf(q)==true and IsLeaf(r)==false) {
    /* choose the child that is closest to q  */
    /* and recurse first */
    recurse(q, closest(q, r.left, r.right))
    recurse(q, furthest(q, r.left, r.right))
  } else {
    recurse(q.left,closest(q.left, r.left, r.right));
    recurse(q.left,furthest(q.left, r.left, r.right));
    recurse(q.right,closest(q.right, r.left, r.right));
    recurse(q.right,furthest(q.right, r.left, r.right));
  }
}
```

Figure 11: Pseudo-code for the dual-tree all nearest neighbor algorithm

So for a single tree algorithm the algorithm behaves fairly well. For the dual tree algorithm though the above approach is not practical since it requires keeping N sorted lists of k length. Since k is a fraction of N the memory requirements are of $O(N^2)$. In Figure 13 we present a single tree algorithm for finding the $k - neighbor$ with minimal memory requirements. The main idea is that for every point we need to keep a subtree with all the candidate k neighbors. Every time the subtree is overfilled with m, the $m + 1$ furthest neighbors are found and the m of them are discarded.

```
recurse(q : KdTree, r : KdTree) {
  if (min_furthest_neighbor_distance_in_node(q)
        < distance(q, r) {
     /* prune */
  } else if (IsLeaf(q)==true and IsLeaf(r)==true) {
    /* search for every point in q node its
    /* furthest neighbor in the r node */
    /* at leaves we must resort to     */
    /* exhaustive search O(n^2) */
    /*update the minimum_furthest_distance_in_node(q)*/
  } else if (IsLeaf(q)==false and IsLeaf(r)=true {
    /*choose the child that is furthest to r */
    /* and recurse first */
    recurse(furthest(r, q.left, q.right), r)
    recurse(closest(r, q.left, q.right), r)
  } else if (IsLeaf(q)==true and IsLeaf(r)==false) {
    /* choose the child that is furthest to q  */
    /* and recurse first */
    recurse(q, furthest(q, r.left, r.right))
    recurse(q, closest(q, r.left, r.right))
  } else {
    recurse(q.left,furthest(q.left, r.left, r.right));
    recurse(q.left,closest(q.left, r.left, r.right));
    recurse(q.right,furthest(q.right, r.left, r.right));
    recurse(q.right,closest(q.right, r.left, r.right));
  }
}
```

Figure 12: Pseudo-code for the dual-tree all furthest neighbor algorithm

2.2 Semidefinite Programming (SDP)

In this section we give an overview of semidefinite programming (SDP) [64, 99, 90, 19] as a tool for expressing geometric constraints.

2.2.1 Convex programming and the semidefinite cone

Convex optimization programs have the well known advantage of converging to globally optimal solutions, that is why they are very appealing in machine learning

```
recurse(q : QueryPoint, r : KdTree, k: Integer, ktree: KdTree) {
  if (ktree.CurrentKdistance < Distance(q, r)) {
    return;
  }
  if (NumOfPointsInTheSubtree(r) <= k) {
    ktree.AddTree(r);
  } else {
    if (IsLeaf(r)) {
      ktree.AddTree(r);
      ktree.RemoveExtraFurthestNeighbors();
    } else {
      recurse(q, ClosestNode(q, r), k);
      recurse(q, FurthestNode(q, r), k);
    }
  }
}

KdTree::AddTree(rtree : KdTree) {
  internal_list.Add(rtree);
}

KdTree::RemoveExtraFurthestNeighbors(ktree : KdTree) {
   list=FindkFurthestNeighbors(ktree);
   Delete(list);
}
```

Figure 13: Pseudo-code for the in-between-k-neighbor. The `RemoveExtraFurthestNeighbors()` removes the extra m extra neighbors by finding the $m+1$ furthest neighbor. m is typically small, less than the maximum number of points in a leaf.

and in engineering in general. The following optimization problem:

$$\min_x f(x) \tag{2.2}$$
$$g_i(x) \leq 0$$
$$h_i(x) = 0$$
$$x \in \Re^m$$

is convex provided that $f(x), g_i(x)$ are convex and $h_i(x)$ is linear. Not all optimization problems can be cast as convex ones. Capabilities of convex programming were mainly limited to linear and quadratic forms. Recent advances in the convex properties of

the semidefinite cone, along with the development of interior point methods opened the door for expressing non-linear programs as convex. Moreover, a lot of non-convex (often combinatorial) programs can be relaxed to convex ones, with the help of semidefinite constraints [99]. Another advantage of the semidefinite constraint is that it is a convex cone that encapsulates the 2nd order and the linear cones. Almost any convex constraint can be expressed or approximated with the help of the semidefinite cone. The very recently developed primal-dual algorithms have computational advantage over the conventional ones [100]. Unfortunately, the primal-dual method is only applicable to small scale problems, since its complexity is cubic, although in some special cases it can be significantly less than that.

In this thesis we will use the symbol $\succeq$ as the positive semidefinite operator. For example $K \succeq 0$ implies that K matrix is positive semidefinite. Also $Z \succeq X$ implies that $Z - X \succeq 0$ meaning that $Z - X$ is positive semidefinite. Given $A, B \in \Re^{N \times N}$ we define the dot product between matrices as $A \bullet B = \text{Trace}(AB^T)$.

2.2.2 Preserving distances

Given a set of points stored in a matrix $X \in \Re^{N \times m}$ the Gram (or dot product matrix) is defined as:

$$G = XX^T \tag{2.3}$$

Notice that by definition $G \succeq 0$ is always positive semidefinite. The Gram matrix is a more abstract representation of the points. Several datasets may map to the same Gram matrix. If $G \succeq 0$ is given then X can be recovered with Singular Value Decomposition, up to an orthogonal transformation. In many algorithms we prefer to represent points with their Gram matrix. One of the great advantages of the Gram matrix is that the distances between two points can be expressed with a linear function. Consider points i, j, where $x_i, x_j \in \Re^{m \times 1}$ then the squared distance d_{ij} is:

$$d_{ij} = x_i^T x_i + x_j^T x_j - x_i^T x_j - x_j^T x_i \tag{2.4}$$

If we want to preserve distances in an optimization problem ,the above constraint is non-convex with respect to x_i, x_j. If we instead use G then the distance d_{ij} can be expressed in the following way:

$$d_{ij} = G_{ii} + G_{jj} - G_{ij} - G_{ji} \tag{2.5}$$

Now the distance between points is linear in G. In general given G we can uniquely retrieve the distance matrix $D \in \Re^{N \times N}$ that contains the squared distances between all pairs [19].

$$D = \delta(G)\mathbf{1}^T + \mathbf{1} + \delta(G)^T - 2G \tag{2.6}$$

where $\delta(G)$ is vector with the diagonal elements of G and $\mathbf{1}$ is a vector full of ones.

2.2.3 Ellipsoidal volumes and density preservation

Another interesting property of G is that it can be uniquely defined given all pairs distances between points [19].

$$G = -\frac{1}{2}(D - \frac{1}{N}D\mathbf{1}\mathbf{1}^T - \frac{1}{N}\mathbf{1}\mathbf{1}^T D + \frac{1}{N^2}\mathbf{1}\mathbf{1}^T D\mathbf{1}\mathbf{1}^T) \tag{2.7}$$

So instead of working with the Gram matrix, it is equivalent to work with the distance matrix. The relationship between the distance matrix and the Gram matrix was pointed in the original work of metric Multidimensional Scaling.

Nemirovski [64] showed that finding the minimum volume ellipsoidal that contains a set of points is an SDP. Given a set of k points $x_i \in \Re^m, i = 1, \ldots, m$, S is the corresponding convex hull and E is an ellipsoidal containing S. The analytic expression of E is:

$$E = \{x | (x - c_*)^T D_* (x - c_*) \leq 1\} \tag{2.8}$$

where c_*, D_* are given by the solution (t_*, Z_*, z_*, s_*) of the following semidefinite

problem:

$$\max_{t,s,z,Z} \quad t \tag{2.9}$$

$$\text{such that:} \tag{2.10}$$

$$t \leq (\det Z)^{\frac{1}{m}}$$

$$Z \succeq 0$$

$$\begin{bmatrix} s & z^T \\ z & Z \end{bmatrix} \succeq 0$$

$$x_i^T Z x_i - 2x_i^T z + s \leq 1, i = 1, \ldots, k$$

$D_* = Z_*, c_* = Z_*^{-1} z_*$.

As we will see later in chapter 3 by approximating the volume of a convex hull of a neighborhood with an ellipsoid we can have an estimate of the density locally.

2.2.4 Geometric means and leave one out cross validation

Often the Gram Matrix might correspond to a generalized dot product as introduced in modern kernel learning theory [76, 79]. In other words there are positive definite kernel functions that generate positive definite matrixes once applied on pairs of points. Probably the most popular kernel is the Gaussian one:

$$k(x_1, x_2) = e^{-||x_1 - x_2||_2^2/\sigma^2} \tag{2.11}$$

where $x_1, x_2 \in \Re^m$ and σ is a real non-zero number representing the bandwidth of the Gaussian. The Gaussian falls under broad class of functions known as Mercer Kernels, that admit a factorization of the form:

$$k(x_1, x_2) = \sum_{i=1}^{N} \phi_i(x_1)\phi_i(x_2) \tag{2.12}$$

where N can be infinity. Kernels have been used in Kernel Density Estimation (KDE) [80] although it is not really necessary to belong in the Mercer family. A very common metric for evaluating the performance of KDE is the Leave One Out Cross Validation

(LOOCV). Given a set of points $x_i \in \Re^m, \quad i = 1, \ldots, N$, the KDE for every point is given by:

$$\hat{f}(x_j) = \sum_{i \neq j} k(x_j, x_i) \tag{2.13}$$

and the LOOCV is given by :

$$LOOCV = \prod_{j=1}^{N} \hat{f}(x_j) \tag{2.14}$$

From equation 2.13 we see that the KDE evaluation on a point is linear on the kernel terms. So instead of using the coordinate representation of a dataset it is better to use the Gram matrix as we also pointed earlier. The expression of LOOCV as pointed in equation 2.14 is neither convex or concave. Yet it easy to bring it in a concave form just by taking the Nth root:

$$\sqrt[N]{LOOCV} = \sqrt[N]{\prod_{j=1}^{N} \hat{f}(x_j)} \tag{2.15}$$

The above form is concave and it admits a second order cone representation [64].

CHAPTER III

MAXIMUM VARIANCE UNFOLDING, SCALABILITY AND EXTENSIONS

Maximum Variance Unfolding (MVU) is one of the-state-of-the-art Manifold Learning (ML) algorithms [95]. In his original paper Weinberger showed that MVU is empirically the best unfolding method that recovers the intrinsic dimension of a manifold compared to other known ML algorithms. These results were also verified in [57]. An improved version of MVU is the Minimum Volume Embedding (MVE) [78] that can handle cases where the data are not on a manifold. It is more general than MVU and it gives lower level embedding when the manifold assumption is not valid. MVE has a more computationally heavy objective function.

Unfortunately MVU and MVE are only scalable up to a few hundred points because they are cast as a Semidefinite Programs (SDP) [90] and their complexity is cubic. Weinberger [94] suggested some solutions to the scalability of MVU by selecting landmark points. This improves scalability but it is still bound on the efficiency of the sampler and on the SDP solver. If the solver can handle N points and the sampler can reduce the points by $1/m$ then the maximum dataset that can be handled is mN. Moreover sampling without error bounds on the final results can be dangerous as there is no guarantee on the induced error. In [98] another approach was given that shares the same limitations.

In [51], a non-convex formulation of MVU known as NCMVU was presented based on the non-convex SDP framework developed in [16]. This method [51] has linear complexity per iteration based on L-BFGS algorithm [65], but nothing can be said about the overall complexity. Experiments in [**?**] showed that it speeds MVU significantly with the disadvantage that it may find non-optimal solutions that correspond to local minima. Despite the speedup the method could only scale up to a few thousand points. This is mainly because MVU, like any other ML technique, requires the computation of all k-nearest neighbors which has quadratic complexity. Another problem of non-convex MVU is auto-tuning the optimization parameters that can distort the optimal solution. A different approach for reducing the complexity

of Semidefinite Programming was presented in [73], where the semidefinite constraint was replaced by diagonal dominance on the Gram matrix. This restriction reduces the problem to a Linear Program which is more scalable.

The first bottleneck was addressed with the dual-tree algorithm described in chapter 2 that has empirically linear complexity and speeds up the computation of neighborhoods significantly. For the second problem (local minima) we introduce a different objective function for unfolding that is based on the distances of the furthest neighbors, Maximum Furthest Neighbor Unfolding MFNU. In the experiments we show that it behaves better than NCMVU in terms of local optima. We also derive an upper bound for NCMVU and MFNU that helps in auto-tuning the optimization parameters. In our experiments we were able to unfold 10-dimensional 100K point datasets in 5 hours. This is the largest dataset to be unfolded based on MVU or its variants on standard machine learning benchmarks. In chapter 7 we present results on even larger datasets. The largest set ever processed with ML was done in [87], that involved 18M images, but different techniques were used.

This chapter is organized as follows: In section 3.1 and 3.2 the convex and non-convex MVU algorithms are outlined. MFNU is presented in section 3.3 and in the end (sections 3.5, 3.5.2) implementational issues and experiments are discussed.

3.1 Maximum Variance Unfolding, the convex SDP case

Weinberger formulated the problem of isometric unfolding as a Semidefinite Programming algorithm [95]. According to his experiments MVU has the best performance compared to the other state of the art Manifold Learning methods.

Given a set of data $X \in \Re^{N \times d}$, where N is the number of points and d is the dimensionality. The dot product or Gram matrix is defined as $G = XX^T$. The goal is to find a new Gram matrix K such that $rank(K) < rank(G)$ in other words $K = \hat{X}\hat{X}^T$ where $\hat{X} \in \Re^{N \times d'}$ and $d' < d$. Now the dataset is represented by $\hat{X}$

which has fewer dimensions that X. The requirement of isometric unfolding is that the Euclidean distances in $\Re^{d'}$ for a given neighborhood around every point have to be the same as in $\Re^d$. This is expressed in:

$$K_{ii} + K_{jj} - K_{ij} - K_{ji} = G_{ii} + G_{jj} - G_{ij} - G_{ji}, \forall i, j \in I_i \tag{1.16}$$

where I_i is the set of the indices of the neighbors of the ith point. From all the K matrices MVU chooses the one with the maximum variance. So the algorithm is presented as an SDP:

$$\begin{aligned} &\max_K \ \text{Trace}(K) \\ &\quad \text{subject to} \\ &\qquad A_{ij} \bullet K &&= d_{ij} \quad \forall i, j \in I_i \\ &\qquad \mathbf{1} \bullet K_{ij} &&= 0 \\ &\qquad K &&\succeq 0 \end{aligned}$$

where A_{ij} has the following form:

$$\begin{bmatrix} 1 & 0 & \dots & -1 & \dots & 0 \\ 0 & \ddots & 0 & \dots & 0 & 0 \\ \vdots & 0 & \ddots & 0 & \dots & 0 \\ -1 & \dots & 0 & 1 & \dots & 0 \\ \vdots & 0 & \dots & 0 & \dots & 0 \\ 0 & \dots & \dots & 0 & \dots & 0 \end{bmatrix} \tag{1.17}$$

$$\mathbf{1} = \begin{bmatrix} 1 & \dots & \dots & 1 \\ \vdots & \ddots & \dots & 1 \\ \vdots & \dots & \ddots & 1 \\ 1 & \dots & \dots & 1 \end{bmatrix} \tag{1.18}$$

and

$$d_{ij} = G_{ii} + G_{jj} - G_{ij} - G_{ji} \tag{1.19}$$

The last constraint is just a centering constraint for the Gram matrix. The new dimensions $\hat{X}$ are the eigenvectors of K. In general MVU gives Gram matrices that have compact spectrum at least more compact than traditional linear Principal Component Analysis (PCA). Unfortunately this method can handle datasets of no more than hundreds of points because of its complexity.

3.2 The non-convex Maximum Variance Unfolding

By replacing the constraint $K \succeq 0$ [16] with an explicit rank constraint $K = RR^T$ the problem becomes non-convex and it is reformulated to

$$\max_R RR^T \tag{2.20}$$

$$\text{subject to} \tag{2.21}$$

$$A_{ij} \bullet RR^T = d_{ij} \tag{2.22}$$

$$\mathbf{1} \bullet RR^T = 0 \tag{2.23}$$

The above problem can be solved with the augmented Lagrangian method [65].

$$\mathcal{L} = -RR^T - \sum_{i=1}^{N} \sum_{\forall j \in I_i} \lambda_{ij}(A_{ij} \bullet RR^T - d_{ij}) + \frac{\sigma}{2} \sum_{i=1}^{N} \sum_{\forall j \in I_i} (A_{ij} \bullet RR^T - d_{ij})^2$$

Because our goal is to minimize the Lagrangian, the objective function is $-RR^T$ and not RR^T

The derivative of the augmented Lagrangian is:

$$\frac{\partial \mathcal{L}}{\partial R} = -2R - 2\sum_{i=1}^{N} \sum_{\forall j \in I_i} \lambda_{ij} A_{ij} \bullet R + 2\sigma \sum_{i=1}^{N} \sum_{\forall j \in I_i} (A_{ij} \bullet RR^T - d_{ij})R \tag{2.24}$$

Gradient descent is a possible way to solve the minimization of the Lagrangian, but it is rather slow. The Newton method is also prohibitive. The Hessian of this problem

is a sparse matrix. Although the cost of the inversion might be high it is worth investigating. In our experiments we used the limited memory BFGS (L-BFGS) method [59, 65] that is known to give a good rate for convergence.

3.3 Maximum Furthest Neighbor Unfolding

Weinberger presented a physical explanation of MVU by simulating every point with a metallic ball connected to its neighbors with a rod. Every rod can freely rotate around the ball. Every ball tries to move far away from the origin bound to its neighbors. Mathematically this is expressed as variance maximization. Instead of maximizing

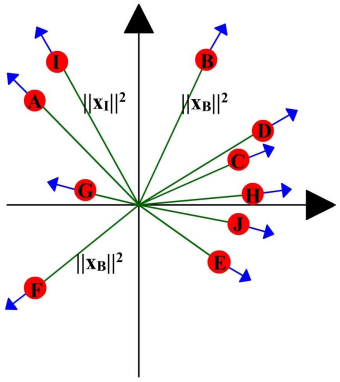


Figure 14: MVU maximizes the distances of points from the origin

the variance which is equivalent to maximizing the distance of every point from the origin Figure 14 we propose maximization of the distance between furthest neighbors Figure 15. So the objective function becomes

$$\max \sum_{i=1}^{N} C_i \bullet RR^T \tag{3.25}$$

where C_i selects the pair of furthest neighbor, it has similar structure with (1.17). This formulation as we will see later leads to better unfolding of the manifold, bypassing local minima. Computing the furthest neighbors is an N-body problem too, meaning that the naive method would be of $O(N^2)$ complexity. It turns out that the dual-tree and single tree algorithms can efficiently compute the furthest neighbors too as already shown in chapter 2.

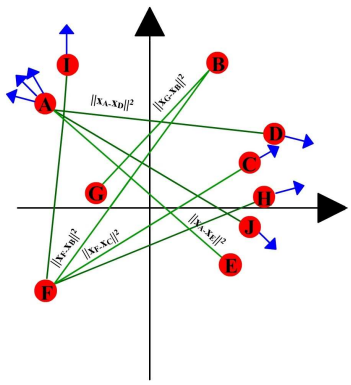


Figure 15: MFNU maximizes distances beteen furthest neighbors. As we can see points get pushed from different directions. This fact helps avoiding local minima

As it can be seen from 4.89 the derivative of the Lagrangian becomes bigger if the first term R is changed to $R(B_{ij}RR^T - f_{ik})$ where f_{ik} is the the distance of the furthest neighbor of i and B_{ij} has similar structure with A_{ij}

Another formulation would be to maximize the variance but make sure that the distance of the furthest neighbors $B_{ij}RR^T$ in the embedded domain is larger than the distance in the Euclidean domain d_{ij}. This condition has been extensively used in manifold learning, specially when ML is intended to be the preprocessing step for classification. Taking account this constraint we exploit more information about the manifold. As we will see further this constraint makes optimization even faster than the simple MVU, because it will give extra push to points that are initially close but we eventually want them to be far away.

3.4 Stochastic Proximity Embedding

Stochastic Proximity Embedding (SPE) [2] has also been suggested as a scalable dimensionality reduction method. The goal of SPE is to minimize a stress function of the distances in the initial domain and in the embedded on. Let d_{ij} be the distance in the initial domain $\Re^d$, and d'_{ij} in the embedded domain $\Re^m$ then SPE minimizes

$$E = \sum \frac{\frac{(d'_{ij}-d_{ij})^2}{d_{ij}}}{\sum d_{ij}}, \forall i,j \quad where \quad d_{ij} < d_c \tag{4.26}$$

d_c is a user defined distance that represents local neighborhoods of the manifold

This method tries to match the distances on the lower dimensional space $\Re^m$ with the distances in the original space $\Re^m$ on local neighborhoods. The original algorithm worked on range d_c neighborhoods while we work on k-neighborhoods which doesn't make a difference. The algorithm is outlined in Figure 16.

1. pick two points randomly i, j.
2. if
   (a) i is in the neighborhood of j
   (b) or i is not in the neighborhood of j and $d_{ij} < d'_{ij}$ go to 3
3. Update

$$y_i = y_i - \lambda \frac{d'_{ij} - d_{ij}}{d'_{ij} + \epsilon}(y_i - y_j) \tag{4.27}$$

$$y_j = y_j - \lambda \frac{d'_{ij} - d_{ij}}{d'_{ij} + \epsilon}(y_j - y_i) \tag{4.28}$$

$$\tag{4.29}$$

4. if step 2 fails go to one
5. Loop for a number of iterations proportional to the number of point in the set

Figure 16: The Stochastic Proximity Embedding Algorithm

The above procedure is repeated for a 100 times in each step λ (see Figure 16) decreases linearly from 2 to 0.1. Two are the main drawbacks of this method. The adhoc parameters and the fact that most of the time is spent in choosing points that might not lead to updates. The use of the dual-tree algorithm for off-line computation of the neighborhoods accelerates the algorithm since all iterations lead to updates. The formulation of SPE is actually the solution of the feasibility problem of the augmented Lagrangian with the addition of the inequality constraints:

$$A_{ij} \bullet RR^T > d_c \tag{4.30}$$

requiring that the distances in the embedded domain between points that are not in

the neighborhood should always be greater than the distances in the original domain.

3.4.1 Evaluation of SPE

One fundamental problem for SPE is that it picks points randomly and most of the time no update is done because points are not in the neighborhood. For that reason we precomputed nearest neighbors and used this information to make sure that the pairs chosen always lead to updates because they satisfy the condition (2.1). Moreover we also used furthest neighbors that, most of the time satisfy condition (2.2). The results where very poor since the algorithm converged slowly to a manifold that wasn't perfectly unfolded. We noticed that running the algorithm several times improved the results at high cost of speed. Another drawback of the procedure is the number of ad-hoc parameters that cannot be easily determined automatically. The method suffers from the same problems of stochastic gradient descent algorithms.

3.5 Implementation of MVU-like methods and Experiments

Our experiments target datasets from 5K up to a 100K points. All the algorithms were developed in C++ as part of the Fastlib library [25], that uses data structures based on BLAS and LAPACK that are known to be optimal for linear algebra operations. Moreover Fastlib has several other optimizations ideal for machine learning algorithms. Fastlib also contains algorithms for kd-trees and fast all-nearest neighbor implementation. All the experiments were run on identical dual Xeon 3.00GHz, 64-bit processors with 8 GB RAM and with the hyperthreading off. The following objective functions were tested:

1. Maximum Variance Unfolding

2. Maximum Furthest Neighbors Unfolding

3.5.1 Implementation Issues of the Augmented Lagrangian and L-BFGS

Before presenting the results the authors think it is necessary to mention some details on the parameters of the optimization methods. As mentioned above, the augmented Lagrangian method has some parameters that need to be tuned. First of all the memory of the L-BFGS method was chosen to be 10.

The sigma (penalty parameter) is the most critical one since if it is very small then the method does not converge. The solution is moving away from the feasible domain. If sigma is very high then the method moves very quickly to feasible domain without giving the opportunity for the variance to be maximized. Our strategy was to always start with a small sigma and if the objective function exceeds an upper bound sigma is increased. Eventually sigma will get the right value. It turns out that a reasonable upper bound for the objective (distance of furthest neighbors) in the optimization problem is the following:

$$B = d_{max} * N^2 \tag{5.31}$$

where d_{max} is the maximum nearest neighbor distance in the set. The geodesic distance between two furthest neighbors cannot be greater than Nd_{max}. This upper bound is also valid for the maximum variance case since the variance is the distance from the axis origin which is always less than the distance of the furthest neighbors.

Another parameter of the method is the k-neighborhood. This can be set ad-hoc or it can be tuned. In the experiments, the technique of leave-one-out cross validation was used described in [91]. As it will be shown in the following sections, a bad choice of the k-parameter can give wrong results.

The only parameter that still remains ad-hoc is the norm gradient accuracy for optimization for a fixed-penalty parameter (sigma). As a heuristic we found out that when the norm gradient is less than sigma the inner optimization should terminate, although we noticed that it could have been set at a higher value. The whole

Table 1: Dataset description

dataset	points	dimension
Swiss roll	up to 100,000	3
TIMIT MFCC features	100,000	39
Corel color histogram	68,040	32
Corel color moments	68,040	9
Corel textures	68,040	16

optimization algorithm terminates when the feasibility error is below a certain value.

3.5.2 Datasets

For our experiments we used the following datasets:

1. A three dimensional Swiss roll ranging from 1000 points up to 100K
2. Speech features from the TIMIT database
3. The Corel image features

Detailed descriptions of the datasets can be found at [38], while a short descriptions can be found in table 1.

The goal of the experiments was to visualize the datasets so we tried low dimensional embedding to 2, 3 dimensions or up to the dimension for which the optimization method would give a satisfactory feasibility error.

Swiss roll experiments. Swiss roll has been a benchmark for manifold learning, but the experiments have been limited to few hundreds of points. In these experiments the goal is to investigate the scalability of both the MVU and MFNU and the quality of the results. In Figure 3.5.2 we see the tremendous difference between convex MVU and NCMFNU, as it is 6 orders of magnitude slower for 100K points.

Maximum Furthest Neighbors Unfolding. In Figure 19.a we see that the algorithm scales in a quasi-linear way. The jump between 50 and 70 thousand points is because we had to increase the number of neighbors. In Figure 19.c we see that

the number of iterations has a linear trend which means that the whole complexity of the algorithm has to be quadratic asymptotically. The reason why we don't notice clear quadratic behavior is because all linear operations of LBFGS don't scale linearly because of the BLAS implementation. BLAS has almost constant behavior for small vectors and asymptotically linear. In Figure 19.c we see that the objective function increases linearly, which is something expected and a good way to verify that the algorithm works, although the results were also visually inspected in order to verify convergence of the algorithm. In Figure 19.d, the number of constraints versus the number of points is plotted. When the k-neighborhoods are computed it is obvious that some of them will be duplicated. In this picture we see that the necessary constraints also grow linearly but they are always less than the imposed ones (solid line).

Maximum Furthest Neighbors Unfolding with auto-tuning. In Figure 20, it is noteworthy that in the middle column the unfolded Swiss roll seems very well unfolded. The scaling seems to be be almost linear until 45000 points.

Maximum Variance Unfolding. In Figure 17 we see the scaling of MVU. By comparing Figure 17.a and 19.a we can see that MVU is faster than MFNU. Unfortunately in fig 21 we see that the MVU gives very poor results and gets trapped into local minima more easily.

Corel dataset. In Figure 22 the results from MFNU are depicted for the color moments and color histograms. We tested different k-neighborhoods and dimensions. It turns out that both datasets can be embedded in 3 dimensions. All the experiments run in reasonable time from a few minutes up to 2 hours.

TIMIT dataset. The TIMIT dataset is a benchmark speech dataset. After sampling we extracted 100,000 39-dim points that correspond to the Mel-frequency cepstrum coefficients of 25msec frames with 12.5msec overlap. The dimension was reduced with PCA and the first 10 principal components were found to correspond to

96% of the total sum of eigenvalues. It took about 5 hours to unfold it in 5 dimensions. Optimization is the most intensive part as it took 95% of the time. The part of computing the nearest neighbors took only 15 minutes with the dual tree algorithm. Just as a comparison, if we are using the naive method it would take 14hours just to compute the neighbors and the whole algorithm would take 14+5=19hours.

Although the dataset was unfolded in 5 dimensions, it turns out that 3 are the dominant ones. This is a very useful result since it shows clearly that there are many redundant dimensions on the initial 10 dimensional dataset.

3.6 Extensions of MVU

In this chapter, we showed experimentally the power of MVU and with the help of furthest neighbors we reformulated the objective so that the optimization is more robust when it comes to the non-convex case. In this section, we will introduce some variations of MVU that can be useful for different types of applications. Our goal is not to show experimental results on real data as it is not yet possible to scale these methods.

3.6.1 Maximization of the Leave One Out Cross Validation (LOOCV)

As discussed in chapter 2, the Gram matrix K can represent any Mercer Kernel function that can be used for Kernel Density Estimation (KDE). The only restriction is that the kernel should be positive everywhere. Therefore the Kernel matrix has two useful applications, computing distances between points and computing densities on points. Our goal is to embed a dataset in a Euclidean space where the coordinates can be used to compute geodesic distances and kernel densities. The formulation of the problem is the following:

$$\max_K \sqrt[N]{\prod_{j=1}^{N} f_j} \tag{6.32}$$

$$s.t.$$

$$A_{ij} \bullet K = d_{ij} \quad \forall i,j \in I_i$$

$$f_j = \sum_{i \neq j} K_{ji}, \quad \forall i = 1, \ldots, N$$

$$K \geq 0$$

$$K \succeq 0$$

The objective function requires approximately $\log_2(N)$ second order cones. In Figures 23 and 24 examples of max LOOCV are shown. Compared to the MVU the spectrum is slightly wider, but considerably more compact to Kernel PCA. Another interesting part is that it deforms the manifold and it squeezes points into clusters.

3.6.2 Preserving volumes/densities instead of distances

In [67] Ozakin showed that for a large class of manifolds it is not possible to preserve the distances. For example, a 3 dimensional ice-cream cone cannot be embedded in 2 dimensions. The same holds for the 3 dimensional sphere. He proved that instead of distances it makes more sense to preserve densities or measures over the manifolds. Based on that he developed an algorithm for embedding a dataset based on density preservation. The algorithm is based on the solution of a non-convex problem. According to the previous analysis the problem can be cast as a semidefinite program:

$$\max_K \text{Trace}(K) \tag{6.33}$$

$$s.t.$$

$$\hat{f}_j = \sum_i K_{ji}, \quad \forall j = 1, \ldots, N$$

$$K \geq 0$$

$$K \succeq 0$$

where $\hat{f}(x_j)$ is the KDE on point x_j estimated with any of KDE method on the original dataset. We also propose a different way to compute densities. The density can be defined as the number of points per volume. Assume that for every point we have found the k-nearest neighbors. For that set it is possible to compute the minimum volume inclusive ellipsoid as shown in chapter 2.

$$E = \{x|(x - c_*)^T D_*(x - c_*) \leq 1\} \tag{6.34}$$

An approximation of density can be:

$$\hat{f}(x_j) = \frac{k}{\det {D_*}^{-1}} \tag{6.35}$$

An alternative way that has been suggested in the past is using a sphere instead of an ellipsoid (not the minimum volume one). This corresponds to centering the sphere on a point and setting the radius equal to the distance of the k-nearest neighbor. Although this is not the optimal packing sphere is computationally cheap.

The formulation in equation 6.33 is weak and it does not generate interesting results. This is mainly because it does not include any information about the geometry of the data. It is easy to verify (we also verified it experimentally with SEDUMI [86]) that the KDE values will be placed on the diagonal giving a very high rank and a trivial solution. So instead we prefer to maximize the distance between neighbors. In other words we are trying to spread the neighbors around the points, but they are constrained by the fact that the sum of their dot products must be equal to the density value.

$$\begin{aligned} \max_K \sum_{(i,j)\in I} A_{ij} \bullet K & \\ s.t. & \\ \hat{f}_j = \sum_i K_{ji}, \quad \forall j = 1, \ldots, N & \\ K_{ij} \geq 0, (i,j) \in I & \\ K \succeq 0 & \end{aligned} \tag{6.36}$$

We call the above algorithm Density Preservation Map (DPM). The optimization of 6.36 can be seen as a reduction process of Kernel matrices. For example a matrix that contains all pair Gaussian kernel evaluations has typically high rank. The rank is controlled by the bandwidth (sigma) of the Gaussian. Large bandwidths give lower rank matrices, while low ones, increase the rank. In general the optimal bandwidth [80, 29] is small, so the rank is expected to be high. In many cases what is really interesting about a kernel matrix is that the sum of its rows corresponds to the density on each point. So with the formulation of equation 6.36 we can construct a matrix that the sum of only k elements of every row matches a given density. Essentially we are stripping information out of the matrix and implicitly we reduce its rank. For the optimization problem only N constraints are necessary. It is well known that the rank of the SDP solution depends on the number of constraints. In Figure 25 we see the results of DPM compared to Kernel PCA. In Figure 26 we show the corresponding eigenvalue spectra. In both cases where the bandwidth is either small or large DPM has more compact spectrum.

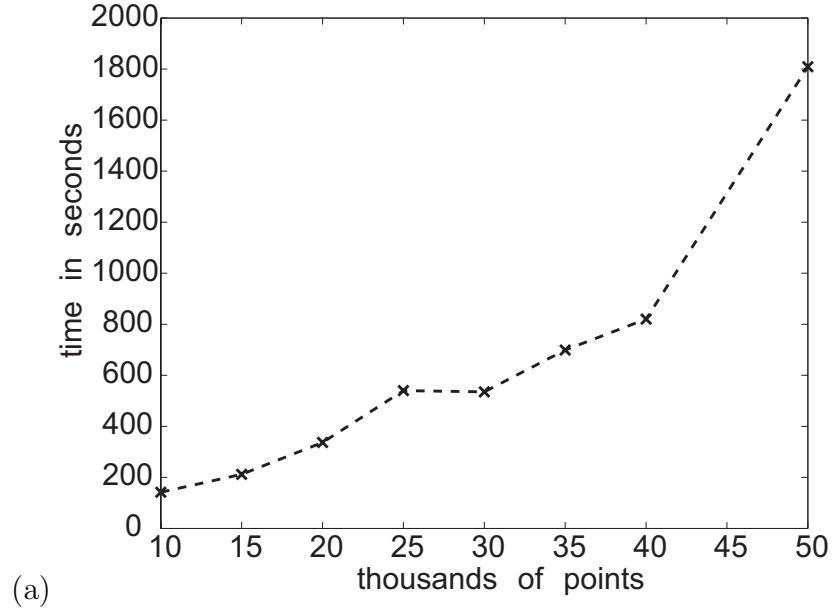
3.7 Summary

In this chapter we presented implementational issues of Maximum Variance Unfolding and tested it over medium size to large datasets. The contribution of this chapter is primarily the modification of the objective function from Max Variance to Max Furthest Neighbor distance, that turned out to have better performance in terms of overcoming local optima. The MFNU method significantly outperforms by far SPE. We also presented some heuristics for auto-tuning of the augmented Lagrangian, by estimating an upper bound for the objective function. Our experiments showed that it is not yet clear if the method can scale linearly, as it depends on the dataset pathology, and on the number of the necessary constraints. Another critical parameter for the scalability of any Manifold Learning Method is the k-neighborhood. Recent work

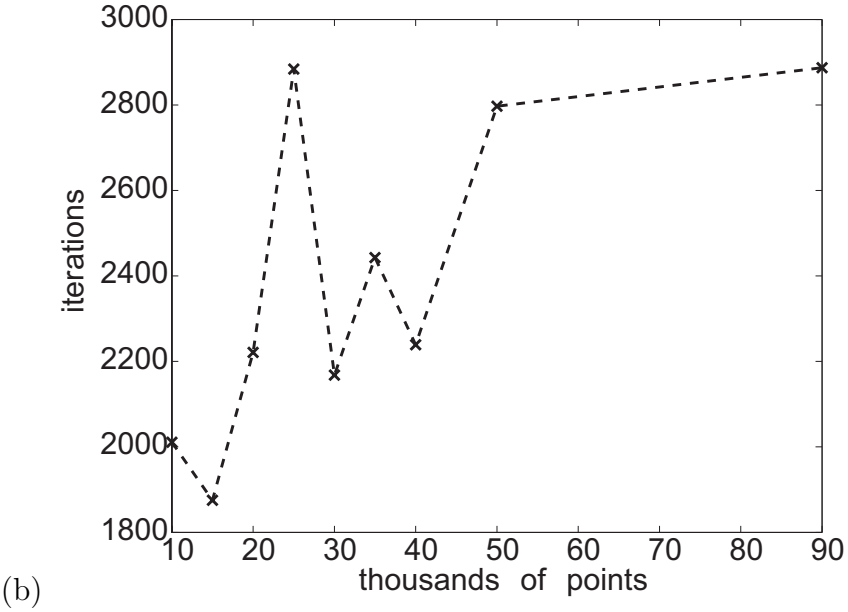
presented in [82] shows that k should be on the order of $O(N^{\frac{1}{d+1}})$, where d is the intrinsic dimensionality. Essentially this adds more complexity on the problem.

This is the first time that MVU is applied on medium to large size datasets, revealing some dimensional aspects. As a future direction we believe that analytical computation of the Hessian and investigation of the right preconditioner for the Newton Method should be considered.

Extensions of MVU on the KDE domain were also introduced and preliminary experiments showed promising results for discovering new kernels suitable for KDE.



(a)



(b)

Figure 17: The classic MVU algorithm a)Scaling performance b)Iterations required for the optimization

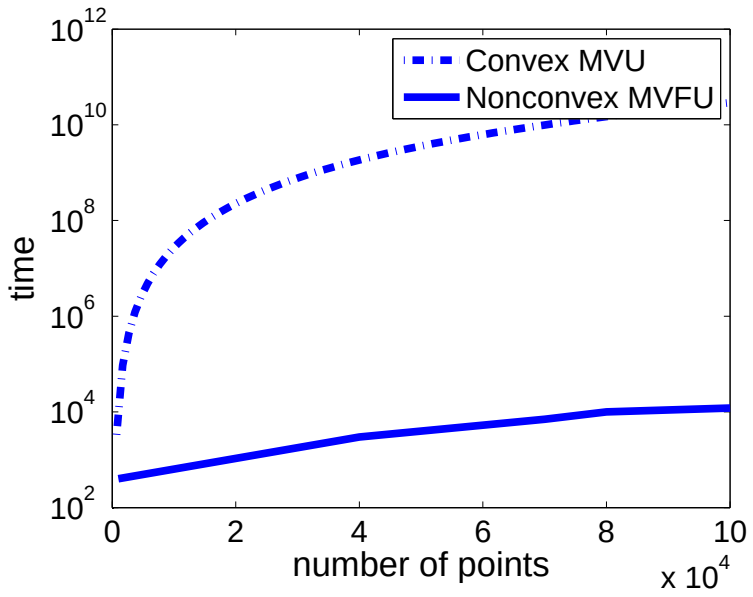


Figure 18: Convex MVU vs non-convex MFNU. For the convex MVU we run experiments up to 600 points and then extrapolated

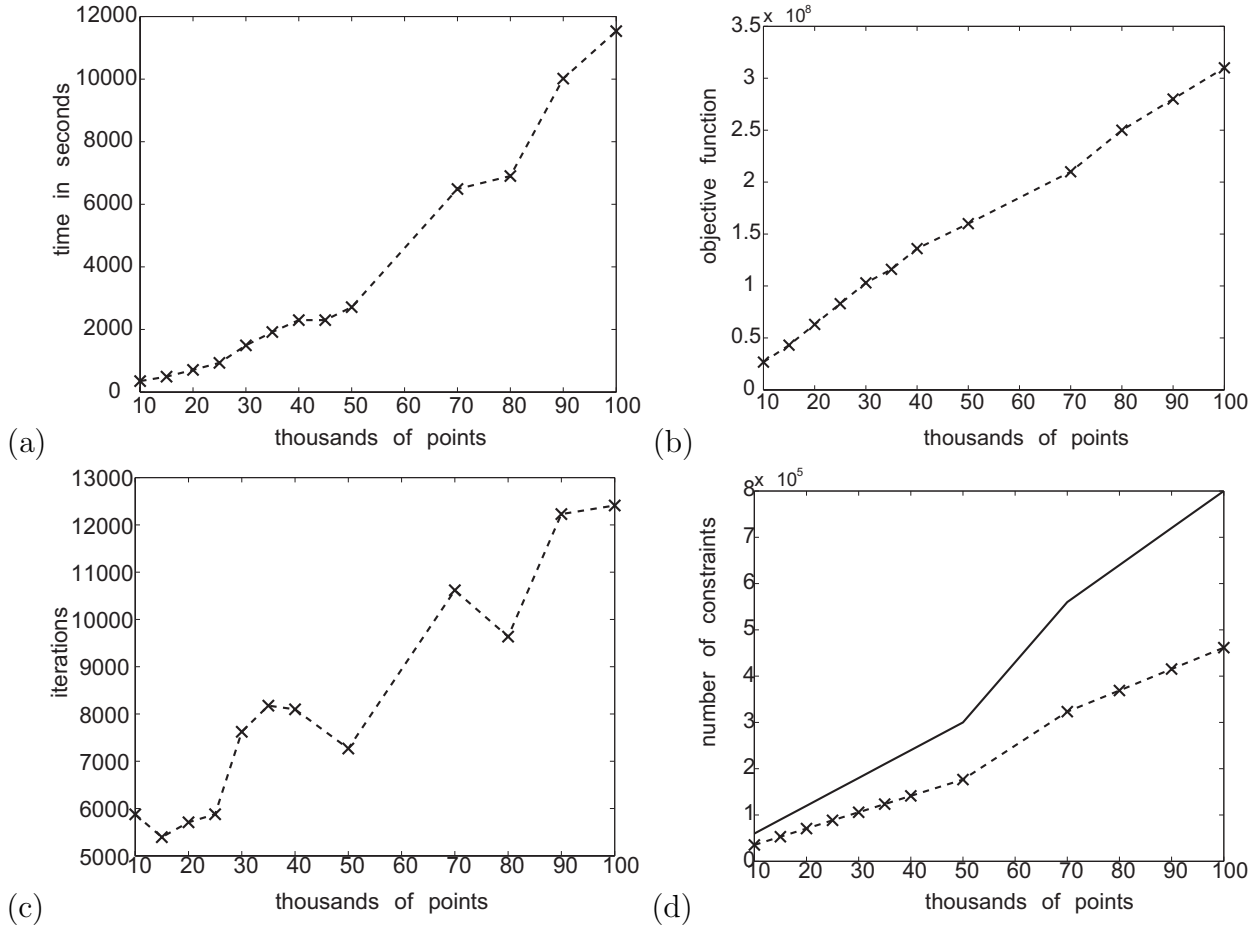


Figure 19: The MFNU performance, a) Scaling of the MFNU b)Maximization results of the Maximum Furthest Neighbors objective c) Iterations required for the optimization d) Number of constrained kN (solid line), consolidated constraints (dashed line)

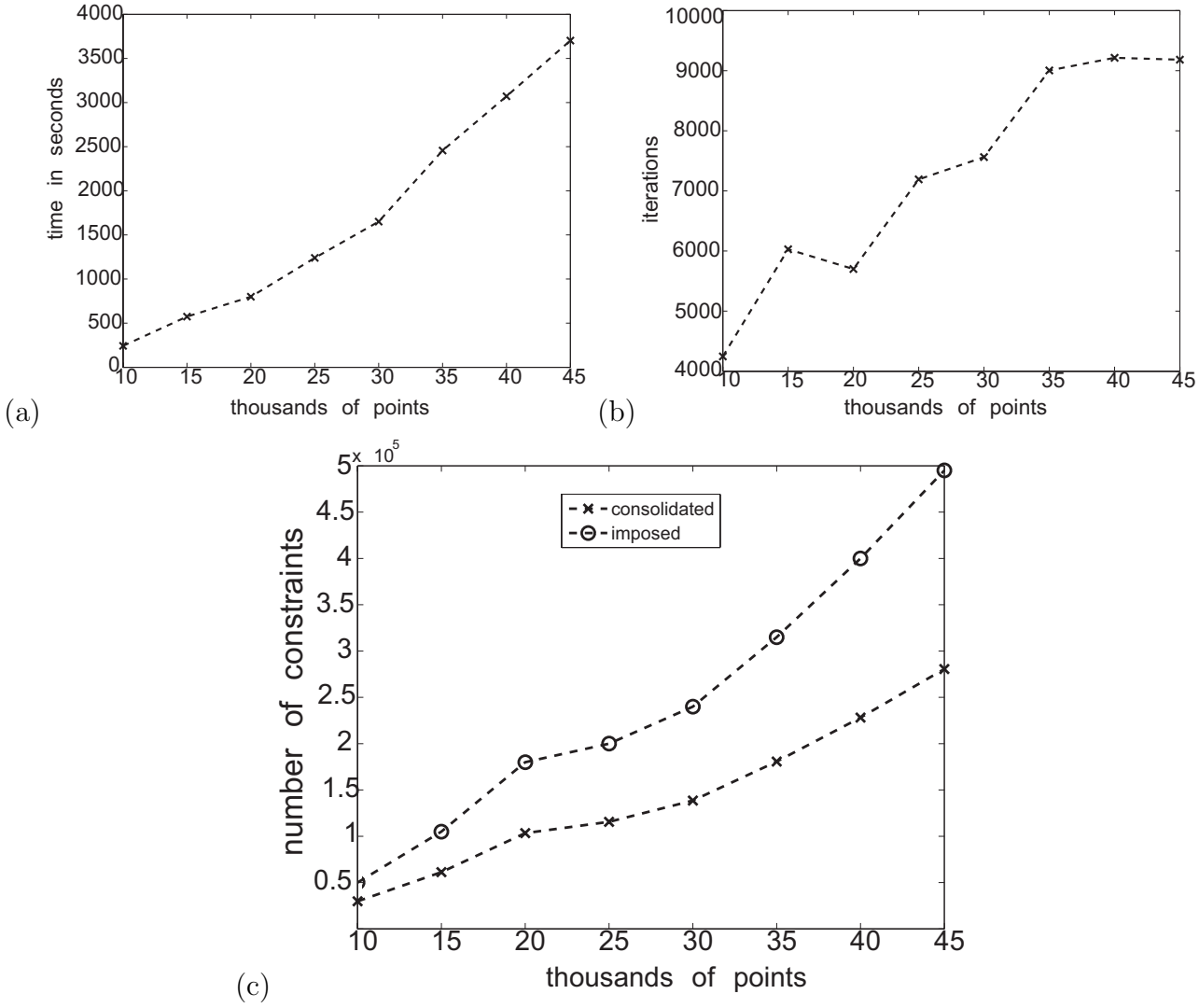


Figure 20: The MFNU algorithm with auto-tuning of k-neighborhoods, a)Scaling of the algorithm b)Iterations required for the optimization c)Number of constrained kN (solid line), consolidated constraints (dashed line)

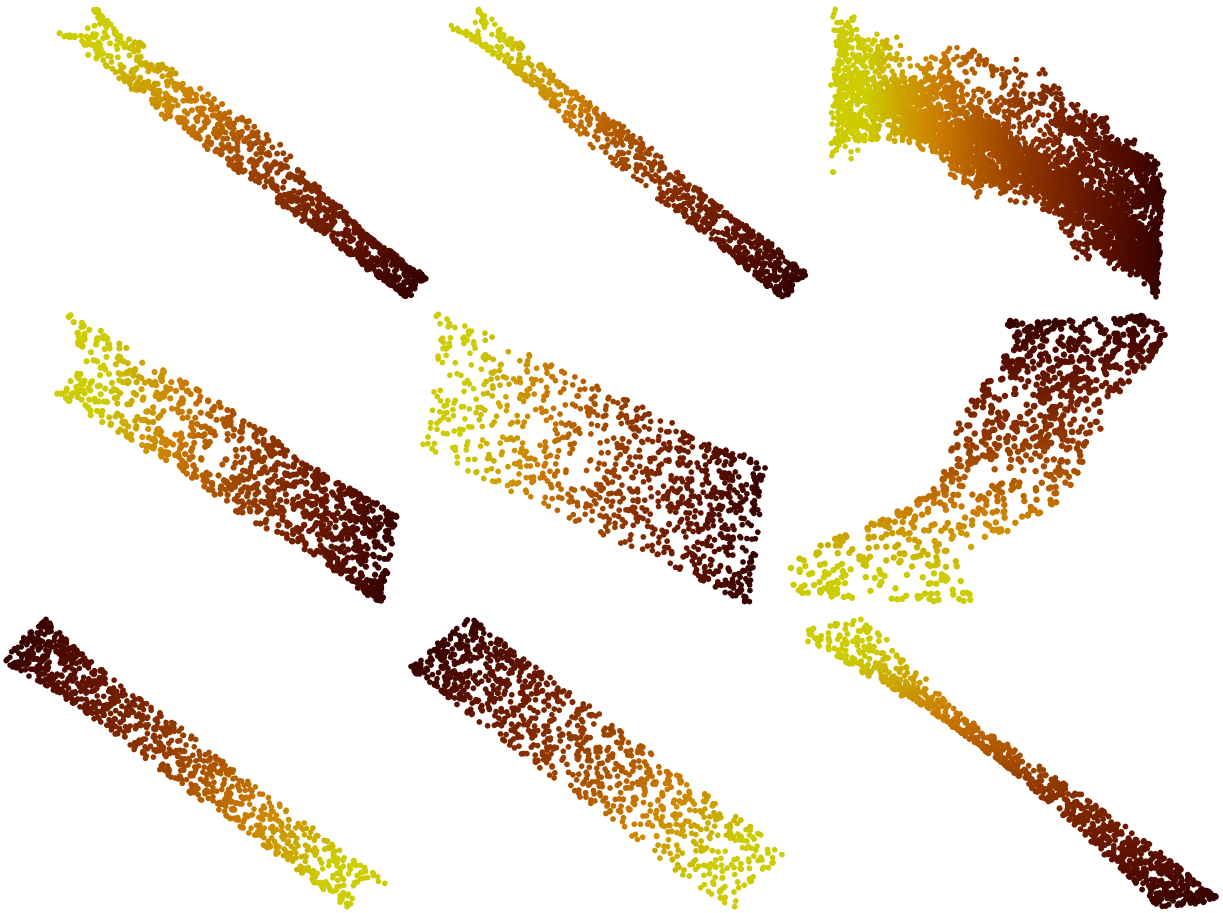


Figure 21: Unfolded Swiss rolls 10K, 20K, 40K (top to bottom row), (left column) MFNU, (center column)MFNU with auto-tuning for k-neighborhoods, (right column)MVU. All images have been sampled showing only 4000 points, for visual clarity

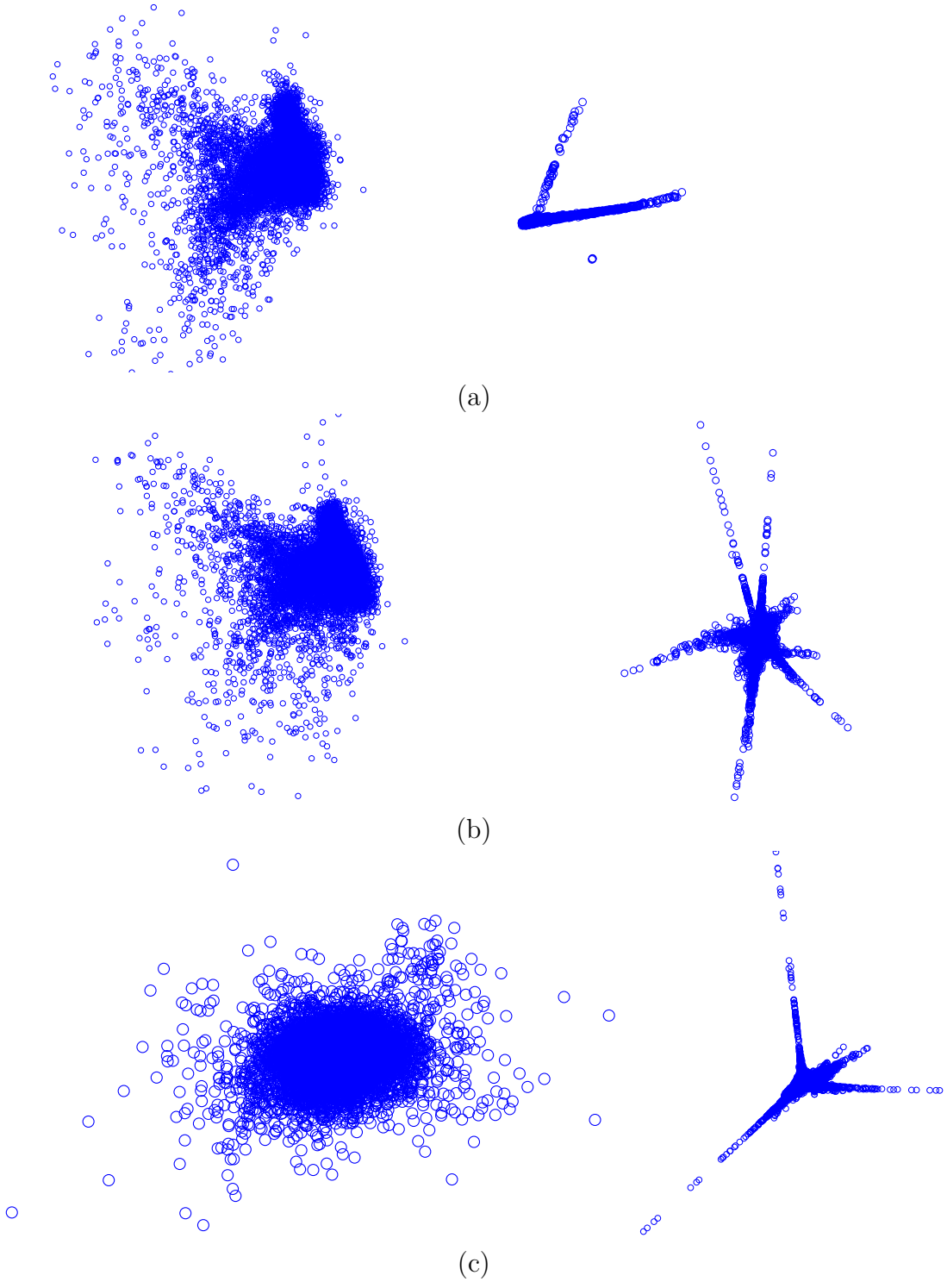


Figure 22: (left column) corel color moments, (right column) corel color histogram, (a)4-point neighborhood, (b)5-point neighborhood, (c)7-point neighborhood

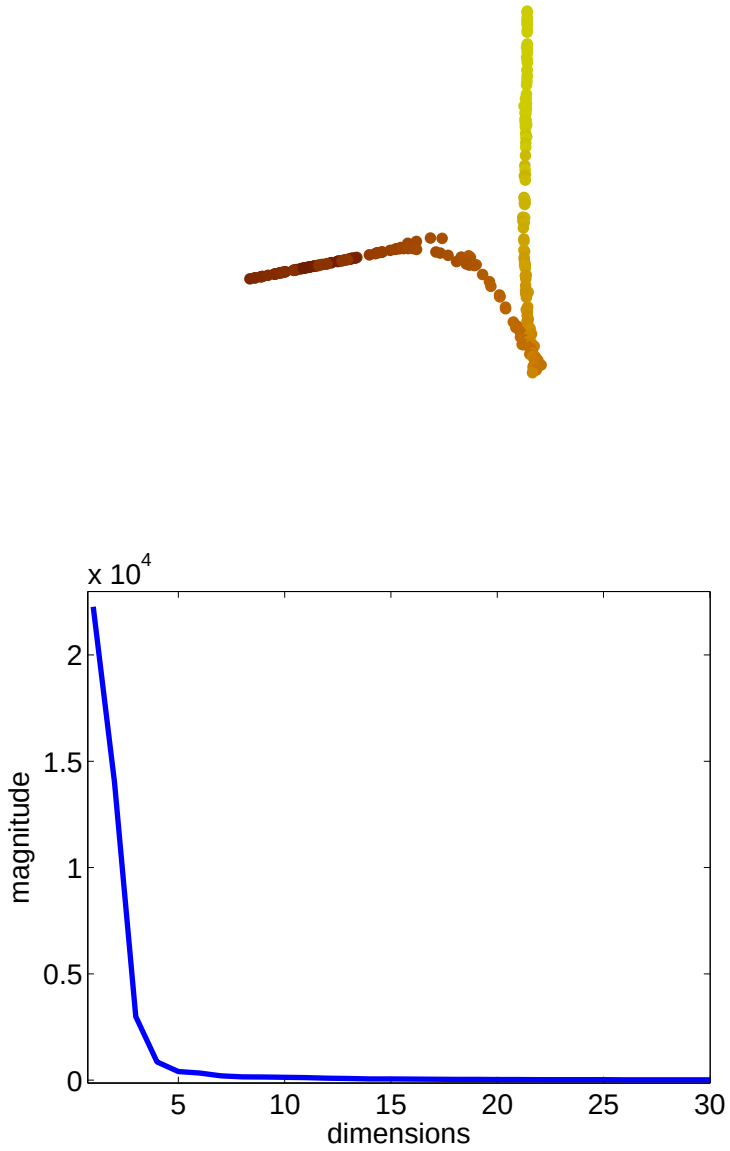


Figure 23: The 3 first components of the Max LOOCV unfolding 256-point swiss roll, along with the eigenvalue spectrum. In this experiment a k=5 neighborhood is preserved.

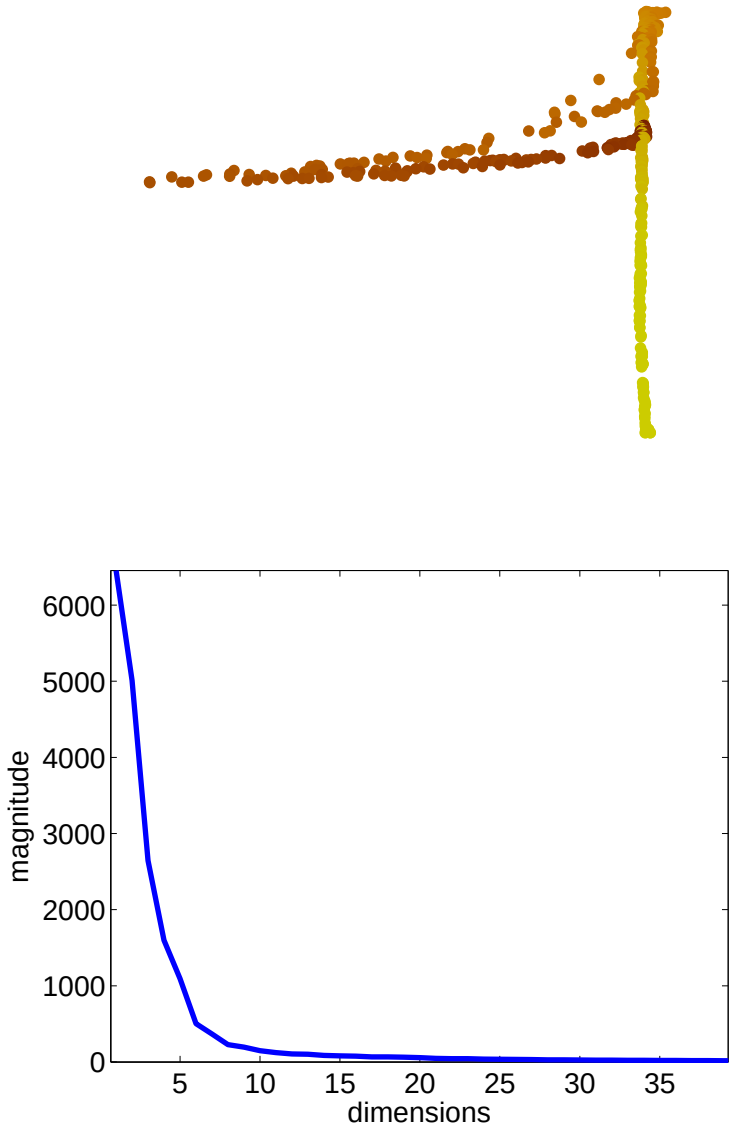


Figure 24: The 3 first components of the Max LOOCV unfolding 512-point swiss roll, along with the eigenvalue spectrum. In this experiment a k=5 neighborhood is preserved.

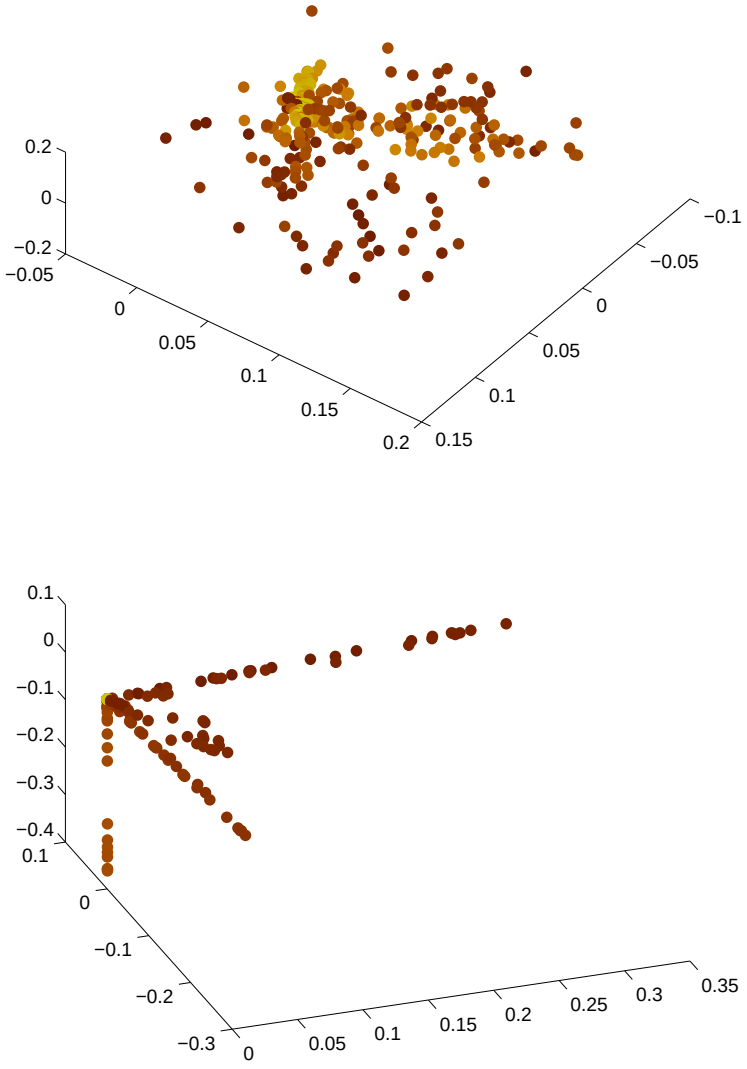


Figure 25: Top: The first 3 components of the DPM for a 400-point swiss roll, Bottom: The first 3 components of Kernel PCA. In this experiment a k=4 neighborhood is preserved. The bandwidth of the Gaussian is medium to small.

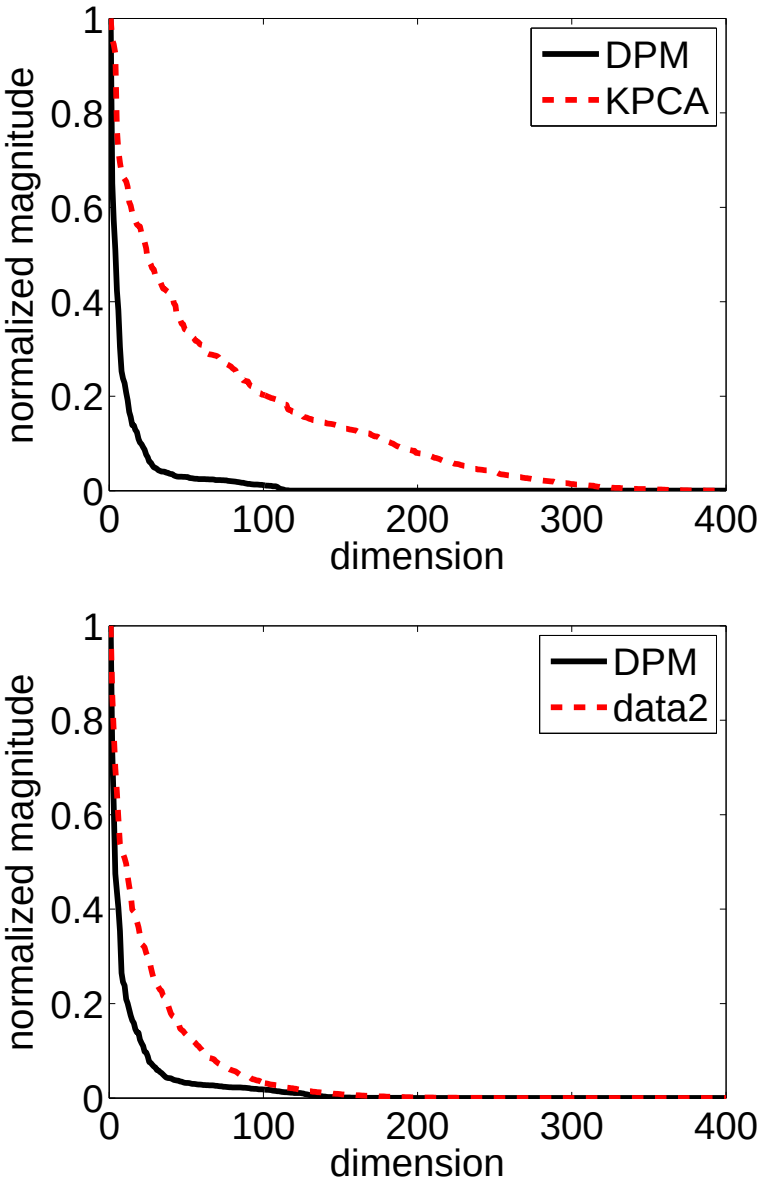


Figure 26: Top: The spectrum of DPM and Kernel PCA for small bandwidth. Bottom: The same spectra for large bandwidth.

CHAPTER IV

NON-NEGATIVE MATRIX FACTORIZATION AS A SPECIAL CASE OF MVU. YET ANOTHER EMBEDDING PROBLEM

In this chapter we are going to address Non-Negative Matrix Factorization [55], [68], another dimensionality reduction method. Through a different reformulation of the problem we will show that this is special case of MVU. We will show that the same framework of semidefinite programming can be used in order to find NMF.

NMF is a dimensionality reduction method of much recent interest which can, for some common kinds of data, sometimes yield results which are more meaningful than those returned by the classical method of Principal Component Analysis (PCA), for example (though it will not in general yield better dimensionality reduction than PCA, as we'll illustrate later). For data of significant interest such as images (pixel intensities) or text (presence/absence of words) or astronomical spectra (magnitude in various frequencies), where the data values are non-negative, NMF can produce components which can themselves be interpreted as objects of the same type as the data which are added together to produce the observed data. In other words, the components are more likely to be sensible images or documents or spectra. This makes it a potentially very useful interpretive data mining tool for such data.

A second important interpretive usage of dimensionality reduction methods is the plot of the data points in the low-dimensional space obtained (2-D or 3-D, generally). Multidimensional scaling methods and recent nonlinear manifold learning methods focus on this usage, typically enforcing that the distances between the points in the original high-dimensional space are preserved in the low-dimensional space (isometry constraints). Then, apparent relationships in the low-dimensional plot (indicating for example cluster structure or outliers) correspond to actual relationships. A plot of the points using components found by standard NMF methods will in general produce misleading results in this regard, as existing methods do not enforce such a constraint.

Another major reason that NMF might not yield reliable interpretive results is that current optimization methods [47, 56] might not find the actual optimum, leading to poor performance in terms of both of the above interpretive usages. This

is because its objective function is not convex, and so unconstrained optimizers are used. Thus, obtaining a reliably interpretable NMF method requires understanding its optimization problem more deeply – especially if we are going to actually create an additionally difficult optimization problem by adding isometry constraints.

In section 4.1 we first study at a fundamental level the optimization problem of standard NMF. We relate for the first time the NMF problem to the theory of Completely Positive Factorization, then using that theory, we show that every non-negative matrix has a non-trivial exact non-negative matrix factorization of the form W=VH, a basic fact which had not been shown until now. Using this theory we also show that a convex formulation of the NMF optimization problem exists, though a practical solution method for this formulation does not yet exist. We then (section 4.2) explore four novel formulations of the NMF optimization problem toward achieving a global optimum: convex relaxation using the positive semidefinite cone, approximating the semidefinite cone with smaller ones, convex multi-objective optimization, and g eneralized geometric programming (section 4.3). We highlight the difficulties encountered by each approach. Following we show that the quality of the relaxations can be improved by cross-validating over sparsity constraints. We also apply a rank reduction method based on a heuristic that finds a local optimum of NMF.

In section 4.4 we show a practical algorithm for an isometric NMF (*isoNMF* for short), representing a new data mining method capable of producing both interpretable components and interpretable plots simultaneously. This method combines the advantages of MVU and its variants presented in chapter 3 and NMF. We use ideas for efficient optimization and efficient neighborhood computation presented in chapter 2 to obtain a practical scalable method.

In section 4.5 we demonstrate the utility of isoNMF in experiments with datasets used in previous papers. We show that the components it finds are comparable to

those found by standard NMF, while it additionally preserves distances much better, and also results in more compact spectra.

4.1 Convexity in Non-Negative Matrix Factorization under the positive completeness.

Given a non-negative matrix $V \in \Re_+^{N\times m}$ the goal of NMF is to decompose it in two matrices $W \in \Re_+^{N\times k}$, $H \in \Re_+^{k\times m}$ such that $V = WH$. Such a factorization always exists for $k \geq m$. The factorization has a trivial solution where $W = V$ and $H = I_m$. Determining the minimum k is a difficult problem and no algorithm exists for finding it. In general we can show that NMF can be cast as a Completely Positive (CP) Factorization problem [9].

DEFINITION 4.1.1. *A matrix $A \in \Re_+^{N\times N}$ is Completely Positive if it can be factored in the form $A = BB^T$, where $B \in \Re_+^{N\times k}$. The minimum k for which $A = BB^T$ holds is called the CP rank of A.*

Not all matrices admit a completely positive factorization even if they are positive definite and non-negative. Notice though that for every positive definite non-negative matrix a Cholesky factorization always exists, but there is no guarantee that the Cholesky factors are non-negative too. Up to now there is no algorithm of polynomial complexity that can decide if a given positive matrix is CP. A simple observation can show that A has to be positive definite, but this is a necessary and not a sufficient condition.

THEOREM 4.1.1. *If $A \in \Re_+^{N\times N}$ is CP then rank(A) $\leq$ cp-rank(A) $\leq \frac{N(N+1)}{2} - 1$*

The proof can be found in [9]p.156. It is also conjectured that the upper bound can be tighter $\frac{N^2}{4}$.

THEOREM 4.1.2. *if $A \in \Re_+^{N\times N}$ is diagonally dominant[1], then it is also CP.*

[1] *A matrix is diagonally dominant if $a_{ii} \geq \sum_{j\neq i} |a_i j|$*

The proof of the theorem can be found in [46]. Although CP factorization ($A = BB^T$) doesn't exist for every matrix, we prove that non-trivial NMF ($A = WH$) always exists.

THEOREM 4.1.3. *Every non-negative matrix $V \in \Re_+^{N \times m}$ has a non-trivial, non-negative factorization of the form $V = WH$.*

Proof. Consider the following matrix:

$$Z = \begin{bmatrix} D & V \\ V^T & E \end{bmatrix} \tag{1.37}$$

We want to prove that there always exists $B \in \Re_+^{N \times k}$ such that $Z = BB^T$. If this is true then B can take the form:

$$B = \begin{bmatrix} W \\ H^T \end{bmatrix} \tag{1.38}$$

Notice that if D and E are arbitrary diagonally dominant completely positive matrices, then B always exists. The simplest choice would be to chose them as diagonal matrices where each element is greater or equal to the sum of rows/columns of V. Since they are diagonally dominant according to 4.1.2 Z is always CP. Since Z is CP then B exists so do W and H. □

Although theorem 4.1.2 also provides an algorithm for constructing the CP-factorization, the cp-rank is usually high. A corollary of theorems 4.1.1 (cp-rank$(A) \geq$ rank(A)) and 4.1.3 (existence of NMF) is that SVD has always a more compact spectrum than NMF.

There is no algorithm known yet for computing an exact NMF despite its existence. In practice, scientists try to minimize the norm [37, 56] of the factorization error.

$$\min_{W,H} ||V - WH||_2^2 \tag{1.39}$$

This is the objective function we use in the experiments for this paper.

4.1.1 Solving the optimization problem of NMF.

Although in the current literature it is widely believed that NMF is a non-convex problem and only local minima can be found, we will show in the following subsections that a convex formulation does exist. Despite the existence of the convex formulation, we also show that a formulation of the problem as a generalized geometric program, which is non-convex, could give a better approach for finding the global optimum.

4.1.1.1 NMF as a convex conic program.

THEOREM 4.1.4. *The set of Completely Positive Matrices $\mathcal{K}^{\mathcal{CP}}$ is a convex cone.*

Proof. See [9]p.71.

It is always desirable to find the minimum rank of NMF since we are looking for the most compact representation of the data matrix V. Finding the minimum rank NMF can be cast as the following optimization problem:

$$\min_{\mathcal{W},\mathcal{H}} \quad \text{rank} \begin{bmatrix} \mathcal{W} & V \\ V^T & \mathcal{H} \end{bmatrix} \tag{1.40}$$

subject to:

$$\mathcal{W} \in \mathcal{K}^{\mathcal{CP}}$$

$$\mathcal{H} \in \mathcal{K}^{\mathcal{CP}} \tag{1.41}$$

Since minimizing the rank is non-convex, we can use it's convex envelope that according to [72] is the trace of the matrix. So a convex relaxation of the above

problem is:

$$\min_{\mathcal{W},\mathcal{H}} \quad \text{Trace}\left(\begin{bmatrix} \mathcal{W} & V \\ V^T & \mathcal{H} \end{bmatrix}\right) \tag{1.42}$$

$$\text{subject to:} \tag{1.43}$$

$$\mathcal{W} \in \mathcal{K}^{\mathcal{CP}}$$

$$\mathcal{H} \in \mathcal{K}^{\mathcal{CP}} \tag{1.44}$$

After determining $\mathcal{W}, \mathcal{H}$, W and H can be recovered by CP factorization of $\mathcal{W}, \mathcal{H}$, which again is not an easy problem. In fact there is no practical barrier function known yet for the CP cone so that Interior Point Methods can be employed. Finding a practical description of the CP cone is an open problem. So although the problem is convex, there is no algorithm known for solving it.

4.2 Convex relaxations of the NMF problem.

In the following subsections we investigate convex relaxations of the NMF problem with the Positive Semidefinite Cone [64].

4.2.1 A simple convex upper bound with Singular Value Decomposition.

Singular Value Decomposition (SVD) can decompose a matrix in two factors U, V:

$$A = UV \tag{2.45}$$

Unfortunately the sign of the SVD components of $A \geq 0$ cannot be guaranteed to be non-negative except for the first eigenvector [62]. However if we project U, V on the nonnegative orthant ($U, V \geq 0$) we get a very good estimate (upper bound) for NMF. We will call it clipped SVD, (CSVD). CSVD was used as a benchmark for the relaxations that follow. It has also been used as an initializer for NMF algorithms [54].

4.2.2 Relaxation with a positive semidefinite cone.

In the minimization problem of equation 4.2.2 where the cost function is the L_2 norm, the nonlinear terms $w_{il}h_{lj}$ appear. A typical way to get these terms [64] would be to generate a large vector $z = [W'(:); H(:)]$, where we use the MATLAB notation ($H(:)$ is the column-wise unfolding of a matrix). If $Z = zz^T$ ($rank(Z) = 1$) and $z > 0$ is true, then the terms appearing in $||V - WH||_2$ are linear in Z. In the following example eq. 2.46, 2.47 (see next page) where $V \in \Re^{2\times 3}, W \in \Re^{2\times 2}, H \in \Re^{2\times 3}$ we show the structure of Z. Terms in bold are the ones we need to express the constraint $V = WH$.

$$z = \begin{bmatrix} w_{11} \\ w_{12} \\ w_{21} \\ w_{22} \\ h_{11} \\ h_{21} \\ h_{12} \\ h_{22} \\ h_{13} \\ h_{23} \end{bmatrix} \tag{2.46}$$

$$Z = \begin{bmatrix} w_{11}^2 & w_{11}w_{12} & w_{11}w_{21} & w_{11}w_{22} & \mathbf{w_{11}h_{11}} & w_{11}h_{21} & \mathbf{w_{11}h_{12}} & w_{11}h_{22} & \mathbf{w_{11}h_{13}} & w_{11}h_{23} \\ w_{12}w_{11} & w_{12}^2 & w_{12}w_{21} & w_{12}w_{22} & w_{12}h_{11} & \mathbf{w_{12}h_{21}} & w_{12}h_{12} & \mathbf{w_{12}h_{22}} & w_{12}h_{13} & \mathbf{w_{12}h_{23}} \\ w_{21}w_{11} & w_{21}w_{12} & w_{21}^2 & w_{21}w_{22} & \mathbf{w_{21}h_{11}} & w_{21}h_{21} & \mathbf{w_{21}h_{12}} & w_{21}h_{22} & \mathbf{w_{21}h_{13}} & w_{21}h_{23} \\ w_{22}w_{11} & w_{22}w_{12} & w_{22}w_{21} & w_{22}^2 & w_{22}h_{11} & \mathbf{w_{22}h_{21}} & w_{22}h_{12} & \mathbf{w_{22}h_{22}} & w_{22}h_{13} & \mathbf{w_{22}h_{23}} \\ h_{11}w_{11} & h_{11}w_{12} & h_{11}w_{21} & h_{11}w_{22} & h_{11}^2 & h_{11}h_{21} & h_{11}h_{12} & h_{11}h_{22} & h_{11}h_{13} & h_{11}h_{23} \\ h_{21}w_{11} & h_{21}w_{12} & h_{21}w_{21} & h_{21}w_{22} & h_{21}h_{11} & h_{21}^2 & h_{21}h_{12} & h_{21}h_{22} & h_{21}h_{13} & h_{21}h_{23} \\ h_{12}w_{11} & h_{12}w_{12} & h_{12}w_{21} & h_{12}w_{22} & h_{12}h_{11} & h_{12}h_{21} & h_{12}^2 & h_{12}h_{22} & h_{12}h_{13} & h_{12}h_{23} \\ h_{22}w_{11} & h_{22}w_{12} & h_{22}w_{21} & h_{22}w_{22} & h_{22}h_{11} & h_{22}h_{21} & h_{22}h_{12} & h_{22}^2 & h_{22}h_{13} & h_{22}h_{23} \\ h_{13}w_{11} & h_{13}w_{12} & h_{13}w_{21} & h_{13}w_{22} & h_{13}h_{11} & h_{13}h_{21} & h_{13}h_{12} & h_{13}h_{22} & h_{13}^2 & h_{13}h_{23} \\ h_{23}w_{11} & h_{23}w_{12} & h_{23}w_{21} & h_{23}w_{22} & h_{23}h_{11} & h_{23}h_{21} & h_{23}h_{12} & h_{23}h_{22} & h_{23}h_{13} & h_{23}^2 \end{bmatrix} \tag{2.47}$$

Now the optimization problem is equivalent to:

$$\min_{Z} \quad \sum_{i=1,j=1}^{i=N,j=m} \sum_{l=1}^{k} (Z_{ik+l,Nk+jk+l} - V_{ij})^2 \tag{2.48}$$
$$\text{subject to:}$$
$$\text{rank(Z)} = 1$$

This is not a convex problem but it can be easily be relaxed to [22]:

$$\min_{Z} \text{Trace}(Z) \tag{2.49}$$
$$\text{subject to:}$$
$$\begin{aligned} A \bullet Z &= V_{ij} \\ Z &\succeq 0 \\ Z &\succeq zz^T \\ Z &\geq 0 \end{aligned}$$

where A is a matrix that selects the appropriate elements from Z. Here is an example for a matrix A that selects the elements of Z that should sum to the V_{13} element:

$$A_{13} = \begin{bmatrix} & 0 & 0 & 0 & 0 & 1 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 1 \\ \mathbf{0} & & & & & & \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{0} & & & \mathbf{0} & & & \end{bmatrix} \tag{2.50}$$

In the second formulation (2.49) we have relaxed $Z = zz^T$ with $Z \succeq zz^T$. The objective function tries to minimize the rank of the matrix, while the constraints try to match the values of the given matrix V. After solving the optimization problem the solution can be found on the first eigenvector of Z. The quality of the relaxation depends on the ratio of the first eigenvalue to sum of the rest. The positivity of Z will

guarantee that the first eigenvector will have elements with the same sign according to the Peron Frobenious Theorem [62]. Ideally if the rest of the eigenvectors are positive they can also be included. One of the problems of this method is the complexity. Z is $(N+m)k \times (N+m)k$ and there are $\frac{((N+m)k)((N+m)k-1)}{2}$ non-negative constraints. Very quickly the problem becomes unsolvable.

In practice the problem as posed in 2.48 always gives W and H matrices that are rank one[2]. After testing the method exhaustively with random matrices V that either had a product $V = WH$ representation or not the solution was always rank one on both W and H. This was always the case with any of the convex formulations presented in this paper. This is because there is a missing constraint that will let the energy of the dot products spread among dimensions. This is something that should characterize the spectrum of H.

The H matrix is often interpreted as the basis vectors of the factorization and W as the matrix that has the coefficients. It is widely known that in nature spectral analysis is giving spectrum that decays either exponentially $e^{-\lambda f}$ or more slowly $1/f^{\gamma}$. Depending on the problem we can try different spectral functions. In our experiments we chose the exponential one. Of course the decay parameter λ is something that should be set adhoc. We experimented with several values of λ, but we couldn't come up with a systematic, heuristic and practical rule. In some cases the reconstruction error was low but in some others not. Another relaxation that was necessary for making the optimization tractable was to reduce the the non-negativity constraints only on the elements that are involved in the equality constraints.

4.2.3 Approximating the SDP cone with smaller ones.

A different way to deal with the computational complexity of SDP (eq. 2.49) is to approximate the big SDP cone $(N+m)k \times (N+m)k$ with smaller ones. Let W_i be the

[2]Unfortunately the non-zero eigenvalues are more than one. As it was highlighted only the first one can be included. For some reason when we construct W and H from the first eigenvector they are always rank 1

i_{th} row of W and H_j the j_{th} column of H. Now $z_{ij} = [W_i(:)'; H_j(:)]$ ($2k$ dimensional vector) and $Z_{ij} = z_{ij} z_{ij}^T$ ($2k \times 2k$ matrix), or

$$Z_{ij} = \begin{bmatrix} W_i^T W_i & W_i^T H_j \\ W_i^T H_j & H_j H_j^T \end{bmatrix} \tag{2.51}$$

or it is better to think it in the form:

$$Z_{ij} = \begin{bmatrix} \mathcal{W}_i & \mathcal{Z}_{WH} \\ \mathcal{Z}_{WH} & \mathcal{H}_j \end{bmatrix} \tag{2.52}$$

and once $\mathcal{W}, \mathcal{H}$ are found then W_i, H_j can be found from SVD decomposition of $\mathcal{W}, \mathcal{H}$ and the quality of the relaxation will be judged upon the magnitude of the first eigenvalue compared to the sum of the others. Now the optimization problem becomes:

$$\begin{aligned} \min \sum_{i=1}^{N} \sum_{j=1}^{m} \text{Trace}(Z_{ij}) & \\ Z_{ij} &\geq 0 \\ Z_{ij} &\succeq 0 \\ A_{ij} \bullet Z_{ij} &= v_{ij}, \quad \forall i, j \end{aligned} \tag{2.53}$$

The above method has Nm constraints. In terms of storage it needs

- $(N + m)$ symmetric positive definite $k \times k$ matrices for every row/column of W, H, which is $\frac{(N+m)k(k+1)}{2}$
- Nm symmetric positive definite $k \times k$ matrices for every $W_i H_j$ product, which is $\frac{(Nm)k(k+1)}{2}$

In total the storage complexity is $O((N+m+Nm)\frac{k(k+1)}{2})$ which is significantly smaller by an order of magnitude from $O(\frac{(N+m)k((N+m)k-1)}{2})$ which is the complexity of the previous method. There is also significant improvement in the computational part. The SDP problem is solved with interior point methods [64] that require the inversion

of a symmetric positive definite matrix at some point. In the previous method that would require $O((N+m)^3k^3)$ steps, while with this method we have to invert Nm $2k \times 2k$ matrices, that would cost $Nm(2k)^3$. Because of their special structure the actual cost is $(Nm)k^3 + \max(N,m)k^3 = (Nm + \max(N,m))k^3$.

We know that $\mathcal{W}_i, \mathcal{H}_j \succeq 0$. Since Z_{ij} is PSD and according to Schur's complement on eq. 2.52:

$$\mathcal{H}_j - \mathcal{Z}_{WH}\mathcal{W}_i^{-1}\mathcal{Z}_{WH} \succeq 0 \tag{2.54}$$

So instead of inverting (2.52) that would cost $8k^3$ we can invert 2.54. This formulation gives similar results with the big SDP cone and most of the cases the results are comparable to the CSVD.

4.2.4 NMF as a convex multi-objective problem.

A different approach would be to find a convex set in which the solution of the NMF lives and search for it over there. Assume that we want to match $V_{ij} = W_i H_j = \sum_{l=1}^{m} W_{il}H_{lj}$. In this section we show that by controlling the ratio of the L_2/L_1 norms of W, H it is possible to find the solution to NMF. Define $W_{il}H_{lj} = V_{ij,l}$ and $\sum_{l=1}^{k} V_{ij,l} = V_{ij}$. We form the following matrix that we require to be PSD:

$$\begin{bmatrix} 1 & W_{il} & H_{lj} \\ W_{il} & t_{il} & V_{ij,l} \\ H_{lj} & V_{ij,l} & t_{jl} \end{bmatrix} \succeq 0 \tag{2.55}$$

If we use the Schur complement we have:

$$\begin{bmatrix} t_{il} - W_{il}^2 & V_{ij,l} - W_{il}H_{lj} \\ V_{ij,l} - W_{il}H_{lj} & t_{jl} - H_{lj}^2 \end{bmatrix} \succeq 0 \tag{2.56}$$

An immediate consequence is that

$$t_{il} \geq W_{il}^2 \tag{2.57}$$

$$t_{jl} \geq H_{ll}^2 \tag{2.58}$$

$$(t_{il} - W_{il}^2)(t_{jl} - H_{lj}^2) \geq (V_{ij,l} - W_{il}H_{lj})^2 \tag{2.59}$$

In the above inequality we see that the L_2 error $\sum_{i=1}^{N}\sum_{j=1}^{m}\sum_{l=1}^{k}(V_{ij,l} - W_{il}H_{lj})^2$ becomes zeros if $t_{il} = W_{il}^2, t_{jl} = H_{il}^2, \quad \forall t_{il}, t_{jl}$. In general we want to minimize t while maximizing $||W||^2$ and $||H||^2$. L_2 norm maximization is not convex, but instead we can maximize $\sum W_{il}, \sum H_{lj}$ which are equal to the L_1 norms since everything is positive. This can be cast as convex multi-objective problem [3] on the second order cone [12].

$$\begin{aligned} \min \quad & \begin{bmatrix} \sum_{i=1}^{i=N}\sum_{l=1}^{k} t_{il} + \sum_{j=1}^{j=m}\sum_{l=1}^{k} t_{lj} \\ -\sum_{i=1}^{i=N}\sum_{l=1}^{k} W_{il} - \sum_{j=1}^{j=m}\sum_{l=1}^{k} H_{lj} \end{bmatrix} \\ & \text{subject to :} \\ & \begin{bmatrix} t_{il} - W_{il}^2 & V_{ij,l} - W_{il}H_{lj} \\ V_{ij,l} - W_{il}H_{lj} & t_{jl} - H_{lj}^2 \end{bmatrix} \succeq 0 \end{aligned} \tag{2.60}$$

Unfortunately multi-objective optimization problems even when they are convex they have local minima that are not global too. An interesting direction would be to test the robustness of existing multi-objective algorithms on NMF.

4.2.5 Augmenting the relaxations with sparsity constraints

A common problem in all the previous relaxations is a constraint on the distribution of the eigenvalues of the problem. A different way to constraint the problem is through the expected sparsity of the result. Hoyer [37] showed that the sparsity $s(x)$ of a vector $x \in \Re^{m\times 1}$ can be expressed through its L_1 and L_2 norms.

$$s(x) = \frac{\sqrt{m} - \frac{||x||_1}{||x||_2}}{\sqrt{m} - 1} \tag{2.61}$$

If x has only one non zero element $s(n)$ evaluates to zero and if all the elements are all the same (up to the sign) it evaluates to 1. The above method has been tested and it is a good measure for the sparsity. If sparsity is given then a linear constraint between L_1 and L_2 norms would suffice to solve the problem. Unfortunately in all the convex

[3] also known as vector optimization

relaxations we have optimizations that correspond to the L_1 norm and to the square of L_2. That means we can only express constraints of the form $L_1 = (\sqrt{m}-(\sqrt{m}-1)s)L_2^2$. Consider the following formulation:

$$Z = \begin{bmatrix} I & W^T & H^T \\ W & \mathcal{W} & V \\ H & V^T & \mathcal{H} \end{bmatrix} \tag{2.62}$$

where $V \in \Re_+^{N\times m}, \quad W \in \Re_+^{N\times k}, \quad H \in \Re_+^{m\times k}$. If the rank of Z is k then there exists a matrix

$$z = \begin{bmatrix} I \\ W \\ H \end{bmatrix} \tag{2.63}$$

such that $Z = zz^T$. This relaxation along with the others suffers from rank one solution. To remedy this problem we enforce sparsity constraints. Sparsity constraints can be enforces either on the W or H matrix. It is more common to enforce sparsity constraints on the W matrix, specially when looking for sparse representation of data. The optimization problem becomes now:

$$\begin{aligned} &\min_Z \text{Trace}(Z) && (2.64)\\ &\quad s.t. \\ &\qquad A_{ij}^V \bullet Z = V_{ij}, i = 1,\ldots,N, \quad j = 1,\ldots,k \\ &\qquad A_{ij}^W \bullet Z \geq 0, i = 1,\ldots,N, \quad j = 1,\ldots,k \\ &\qquad A_{ij}^H \bullet Z \geq 0, i = 1,\ldots,m, \quad j = 1,\ldots,k \\ &\qquad \sum_{i=k+1}^{N}\sum_{j=1}^{k} Z_{ij} = (\sqrt{m}-(\sqrt{m}-1)s)\sum_{i=k+1}^{N} Z_{ii} \end{aligned}$$

where A^V, A^W, A^H are matrices that select elements of W, H, V. The last constraint is an approximation to the sparsity constrain on W as indicated in equation 2.61. In some problems it is useful to constraint the L_2 norm to 1, so the last constraint

becomes an exact sparsity constraint. In reality the sparsity of the results is not known or estimated ahead of time, so in practice the we solve the above problem for different values of s and we pick the solution with the minimum error. A similar approach for solving sparse NMF with a sequence of 2nd order conic programs and Reverse Convex Programming was presented in [33, 32]. The method computes sequentially updates on W, H and it is different to ours.

4.2.6 An algorithm for rank constrained problems

In his excellent book [19] Dattorro presented an iterative algorithm for rank constraint optimization problems. The algorithm presented is a sequence of semidefinite programs. Assume the following SDP problem:

$$\min_{G \in \Re^{N \times N}} \text{Trace}(G) \tag{2.65}$$

$$s.t. \tag{2.66}$$

$$A_i \bullet G = b_i, \forall i = 1, \ldots, p$$

$$B_i \bullet G \geq c_i, \forall i = 1, \ldots, q$$

$$G \succeq 0$$

$$\text{rank}(G) \leq n;$$

In [19] Dattorro proved that a local solution for the problem can be given by the following sequential algorithm:

1. Set $R = I$

2. Let G^* be the optimal solution of the following SDP:

$$\min_{G \in \Re^{N \times N}} G \bullet R \tag{2.67}$$

$$s.t. \tag{2.68}$$

$$A_i \bullet G = b_i, \forall i = 1, \ldots, p$$

$$B_i \bullet G \geq c_i, \forall i = 1, \ldots, q$$

$$G \succeq 0$$

$$\tag{2.69}$$

3. Find R^* by solving the following SDP:

$$\min_{R \in \Re^{N \times N}} G^* \bullet R \tag{2.70}$$

$$I \succeq W \succeq 0$$

$$\text{Trace}(R) = N - n$$

4. if $G^* \bullet R^* \leq \epsilon$ terminate. Otherwise set $R = R^*$ and go to step 2.

The algorithm is known to behave well for small problems [4]. Following we apply and evaluate the above algorithm in the NMF problem as discussed in the previous section without the sparsity constraint.

1. Set $R = I$

[4] According to [19] the method fails in large scale problems due to numerical errors of the current interior point solvers

2. Let Z^* be the optimal solution of the following SDP:

$$\min_{Z} Z \bullet R \tag{2.71}$$

$$s.t.$$

$$A_{ij}^V \bullet Z = V_{ij}, i = 1, \ldots, N, \quad j = 1, \ldots, k$$

$$A_{ij}^W \bullet Z \geq 0, i = 1, \ldots, N, \quad j = 1, \ldots, k$$

$$A_{ij}^H \bullet Z \geq 0, i = 1, \ldots, m, \quad j = 1, \ldots, k \tag{2.72}$$

3. Find R^* by solving the following SDP:

$$\min_{R \in \Re^{N \times N}} Z^* \bullet W \tag{2.73}$$

$$I \succeq W \succeq 0$$

$$\text{Trace}(Z) = N - k$$

4. if $Z^* \bullet R^* \leq \epsilon$ terminate. Otherwise set $R = R^*$ and go to step 2.

4.2.7 Experiments

In this section we evaluate the performance of the sparsity constraint optimization as well as the algorithm 4.2.6 described in the previous section. Experiments on random matrices with algorithm 2.64 showed poor performance, compared to the standard non-convex solver. On the other hand the rank reduction algorithm 4.2.6 was tested on several synthetic random matrices with known factorization and successfully recovered the NMF components. Compared to the standard NMF it converged much faster in a relative error on the order of $1e-8$ for small matrices. For larger matrices the semidefinite program, becomes slow and for even larger ones intractable. Specially the second one in the algorithm 4.2.6 has 2 positive semidefinite inequalities $R \succeq 0, Z \succeq 0$ and a matrix equality $Z = I - W$ that requires $O((N+M+k)^2)$ equalities. We examine the NMF of a picture that contains 3 rice seeds Figure 27.

On this particular image the convex rank reduction algorithm gives the smaller error with the non-convex solver (relative error 5.2% ¡7%). In Figure 28, 29 we show the 6 components of both methods that are not very different. The reconstructed images from both NMF algorithms is shown in Figure 30, 31 For comparison we also show (Figure 33) the 6 first components that give 20% relative reconstruction error (Figure 32). As it can be seen the svd components are meaningless, while the NMF ones identify areas where the rice seeds exist.

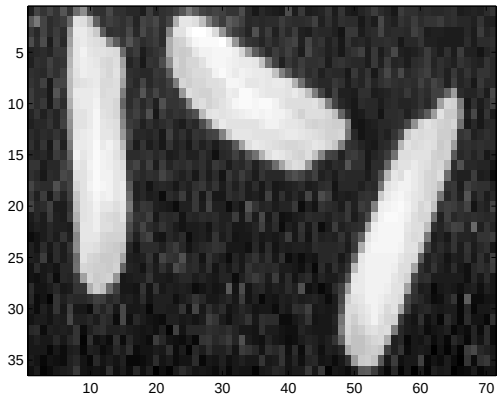


Figure 27: 3 rice seeds original gray scale image

Although the rank constraint algorithm can guarantee local but not global convergence, it tends to be very robust. We tested it extensively in over 1000 matrices with known factorization and it always found the solution. A very similar heuristic for reducing the rank of a matrix was introduced in the Minimum Volume Embedding (MVE) [78]. The authors verified the robustness with a great variety of experiments.

4.2.8 An algorithm for solving the problem of Completely Positive Factorization

We started this chapter with the theory of Completely Positive Matrices (CPM). The theory of CPMs gave us a guarantee that NMF has a convex formulation, but never provided an algorithm for computing it. Up to now CP factorization of a matrix in polynomial time is an open problem. In this section we show that a rank constrained

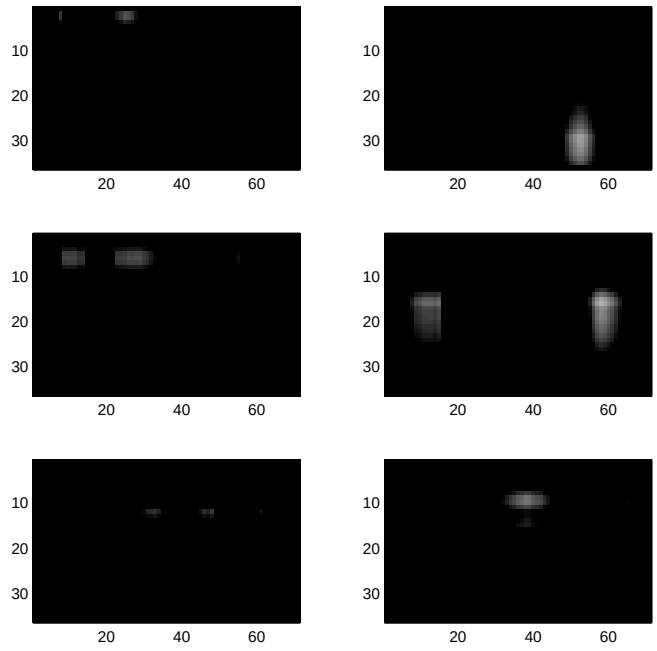


Figure 28: The 6 components of rank reduction convex NMF

algorithm presented in section 4.2.6 can solve the problem of CP factorization and find a local solution.

Given matrix $A \in \Re_+^{N \times N}$, we want to test whether matrix $B \in \Re_+^{N \times k}$ such that $A = BB^T$, exists. Consider the matrix:

$$Z = \begin{bmatrix} I_k & B^T \\ B & A \end{bmatrix} \tag{2.74}$$

Matrix B can be found by solving the feasibility problem:

$$\text{find } B \tag{2.75}$$

$$\text{such that:} \tag{2.76}$$

$$B \geq 0 \tag{2.77}$$

$$\text{rank}(Z) \leq k \tag{2.78}$$

$$Z \succeq 0 \tag{2.79}$$

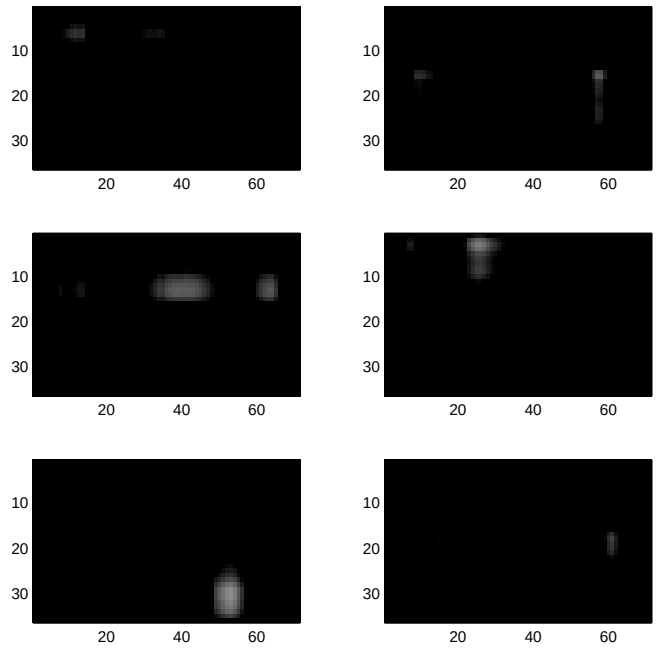


Figure 29: The 6 components of non-convex NMF

The above problem can be solved with the algorithm presented in section 4.2.6. We know from theorem 4.1.1 that $\text{rank}(A) \leq k \leq \frac{N(N+1)}{2} - 1$ (also known as CP-rank). In order to decide if a matrix is CP we need to solve 2.75 $\frac{N(N+1)}{2} - 1$ times in the worst case. Since the solution of 2.75 is local and not global if the algorithm converges to $Z^* \bullet R^* = \tau \neq 0$ then we don't know if this is because of infeasibility of the problem or of local convergence.

4.3 Global and local solutions of non-convex NMF.

In the previous sections we gave several convex formulations and relaxations of the NMF problem that unfortunately are either unsolvable or they are not scalable at all.

In practice the non-convex formulation of eq. 4.2.2 (classic NMF objective) along with other like the KL distance between V and WH are used in practice [56]. All of them are non-convex and several methods have been recommended, such as alternating least squares, gradient decent or active set methods [47]. In our

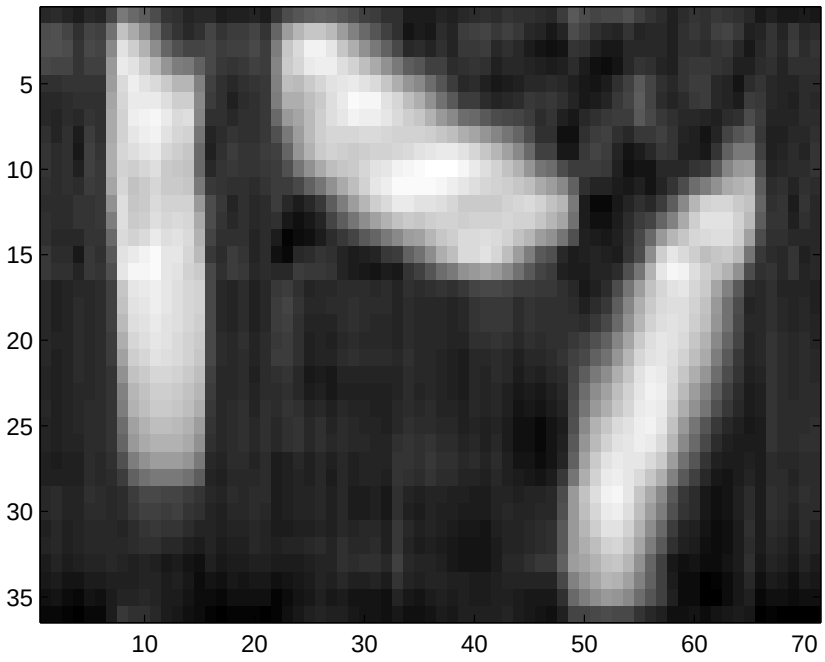


Figure 30: The 3 rice seeds image reconstructed from the 6 NMF components acquired with the rank reduction convex NMF

experiments we used the L-BFGS method that scales very well for large matrices.

4.3.1 NMF as a Generalized Geometric Program and its Global Optimum.

The objective function (eq. 4.2.2) can be written in the following form:

$$||V - WH||_2^2 = \sum_{i=1}^{N} \sum_{j=1}^{m} \sum_{l=1}^{k} (V_{ij} - W_{il} H_{lj})^2 \tag{3.80}$$

The above function is twice differential so according to [36] the function can be cast as the difference of convex (d.c.) functions. The problem can be solved with general off-the-shelf global optimization algorithms. It can also be formulated as a special case of dc programming, the generalized geometric programming. With the following

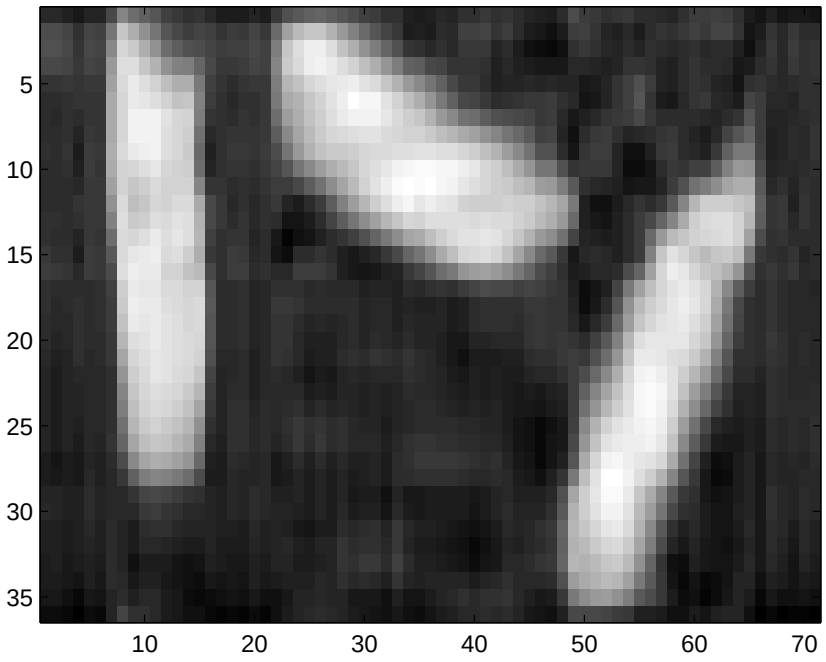


Figure 31: The 3 rice seeds image reconstructed from the 6 NMF components acquired with non-convex NMF

transformation $W_{il} = e^{\tilde{w}_{il}}, H_{lj} = e^{\tilde{h}_{lj}}$ the objective becomes:

$$||V - WH||_2^2 = \sum_{i=1}^{N}\sum_{j=1}^{m}\sum_{l=1}^{k}\left(V_{ij} - e^{\tilde{w}_{il}+\tilde{h}_{lj}}\right)^2 \tag{3.81}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{m} V_{ij}^2 + \sum_{i=1}^{N}\sum_{j=1}^{m}\left(\sum_{l=1}^{k} e^{\tilde{w}_{il}+\tilde{h}_{lj}}\right)^2$$

$$-2\sum_{i=1}^{N}\sum_{j=1}^{m} V_{ij}\left(\sum_{l=1}^{k} e^{\tilde{w}_{il}+\tilde{h}_{lj}}\right)$$

The first term is constant and it can be ignored for the optimization. The other two terms:

$$f(\tilde{w}_{il}, \tilde{h}_{lj}) = \sum_{i=1}^{N}\sum_{j=1}^{m}\left(\sum_{l=1}^{k} e^{\tilde{w}_{il}+\tilde{h}_{lj}}\right)^2 \tag{3.82}$$

$$g(\tilde{w}_{il}, \tilde{h}_{lj}) = 2\sum_{i=1}^{N}\sum_{j=1}^{m} V_{ij}\left(\sum_{l=1}^{k} e^{\tilde{w}_{il}+\tilde{h}_{lj}}\right) \tag{3.83}$$

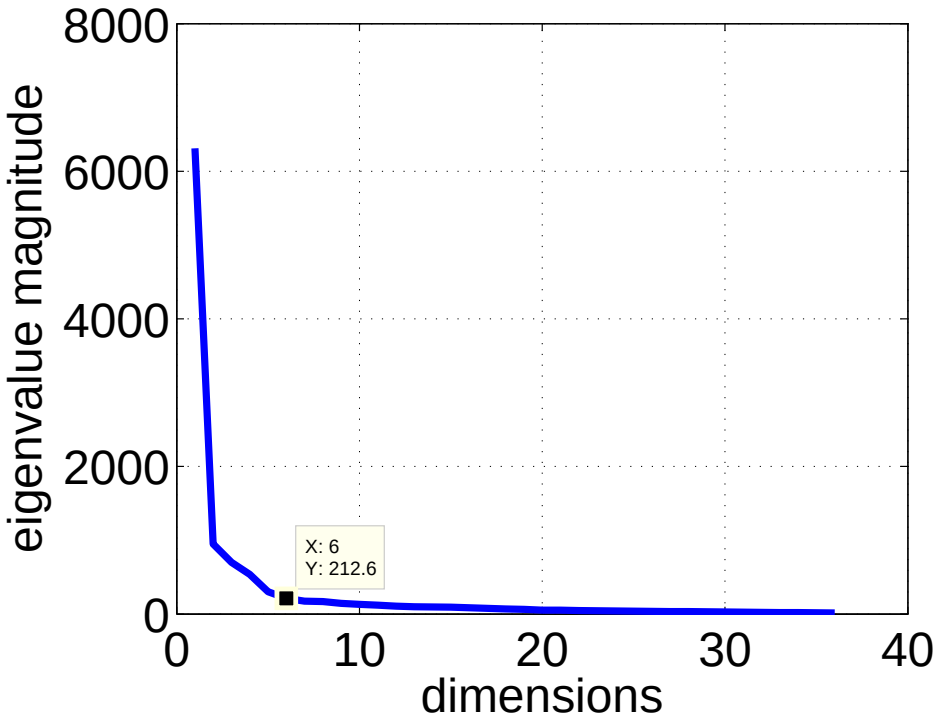


Figure 32: The singular values of the 3 rice seeds

are convex functions also known as the exponential form of posinomials [5] [12]. For the global solution of the problem

$$\min_{\tilde{W},\tilde{H}} \quad f(\tilde{W},\tilde{H}) - g(\tilde{W},\tilde{H}) \tag{3.84}$$

the algorithm proposed in [23] can be employed. The above algorithm uses a branch and bound scheme that is impractical for high dimensional optimization problems as it requires too many iterations to converge. It is worthwhile though to compare it with the local non-convex NMF solver on a small matrix. We tried to do NMF of order 2 on the following random matrix:

$$\begin{bmatrix} 0.45 & 0.434 & 0.35 \\ 0.70 & 0.64 & 0.43 \\ 0.22 & 0.01 & 0.3 \end{bmatrix}$$

[5] Posynomial is a product of positive variables exponentiated in any real number

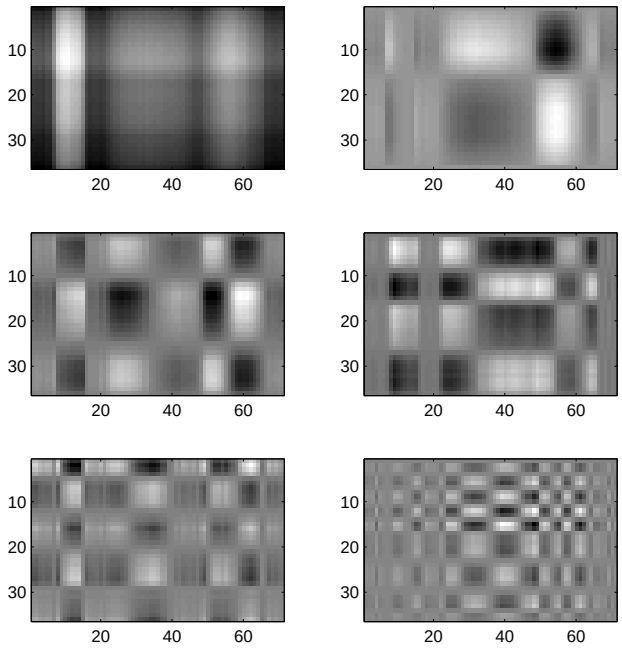


Figure 33: The six first svd components of the 3 rice seeds

After 10000 restarts of the local solver the best error we got was 2.7% while the global optimizer very quickly gave 0.015% error, which is 2 orders of magnitude less than the local optimizer.

Another direction that is not investigated in this paper is the recently developed algorithm for Difference Convex problems by Tao [88] that has been applied successfully to other data mining applications such as Multidimensional Scaling. [3].

4.4 Isometric NMF.

NMF and MFNU are both optimization problems. The goal of isoNMF is to combine these optimization problems in one optimization problem. MFNU has a convex and a non-convex formulation, while for NMF only a non-convex formulation that can be solved is known.

4.4.1 Convex isoNMF.

By using the theory presented in section 4.1.1.1 we can cast isoNMF as a convex problem:

$$\begin{aligned} \max_{\tilde{W},\tilde{H}} \quad & \sum_{i=1}^{N} B_i \bullet Z \\ & \text{subject to:} \\ & A_{ij} \bullet \tilde{W} = d_{ij} \\ & Z = \begin{bmatrix} \tilde{W} & V \\ V^T & \tilde{H} \end{bmatrix} \\ & Z \in \mathcal{K}^{\mathcal{CP}} \\ & \tilde{W} \in \mathcal{K}^{\mathcal{CP}} \\ & \tilde{H} \in \mathcal{K}^{\mathcal{CP}} \end{aligned} \tag{4.85}$$

Then W, H can be found by the completely positive factorization of $\tilde{W} = WW^T, \tilde{H} = HH^T$. Again this problem although it is convex, there is no polynomial algorithm known for solving it. The rank reduction framework of section 4.2.6 can also be applied but it is not yet scalable.

4.4.2 Non-convex formulation of isoNMF.

The non convex isoNMF can be cast as the following problem:

$$\begin{aligned} \max \quad & \sum_{i=1}^{N} B_i \bullet WW^T \\ & \text{subject to:} \\ & A_{ij} \bullet WW^T = d_{ij} \\ & WH = V \\ & W \geq 0 \\ & H \geq 0 \end{aligned} \tag{4.86}$$

The augmented lagrangian with quadratic penalty function is the following:

$$\begin{aligned}\mathcal{L} &= -\sum_{i=1}^{N} B_i \bullet WW^T \\ &-\sum_{i=1}^{N}\sum_{\forall j\in I_i} \lambda_{ij}(A_{ij}\bullet WW^T - d_{ij}) \\ &-\sum_{i=1}^{N}\sum_{j=1}^{m}\mu_{ij}\sum_{l=1}^{k}(W_{ik}H_{kj} - V_{ij}) \\ &+\frac{\sigma_1}{2}\sum_{i=1}^{N}\sum_{\forall j\in I_i}(A_{ij}\bullet WW^T - d_{ij})^2 \\ &+\frac{\sigma_2}{2}\sum_{i=1}^{N}\sum_{j=1}^{m}\sum_{l=1}^{k}(W_{il}H_{lj} - V_{ij})^2\end{aligned} \tag{4.87}$$

The non-negativity constraints are missing from the Lagrangian. This is because we can enforce them through the limited bound BFGS also known as L-BFGS-B. The derivative of the augmented Lagrangian is:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial W} &= -2\sum_{i=1}^{N} B_i W \\ &-2\sum_{i=1}^{N}\sum_{\forall j\in I_i}\lambda_{ij}A_{ij}W \\ &-\sum_{i=1}^{N}\sum_{j=1}^{m}\mu_{ij}W \\ &+2\sigma_1\sum_{i=1}^{N}\sum_{\forall j\in I_i}(A_{ij}\bullet WW^T - d_{ij})A_{ij}W \\ &+2\sigma_2\sum_{i=1}^{N}\sum_{j=1}^{m}\sum_{l=1}^{k}(W_{il}H_{lj} - V_{ij})W\end{aligned} \tag{4.88}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial H} &= -\sum_{i=1}^{N}\sum_{j=1}^{m}\mu_{ij}H \\ &+2\sigma_2\sum_{i=1}^{N}\sum_{j=1}^{m}\sum_{l=1}^{k}(W_{il}H_{lj} - V_{ij})H\end{aligned} \tag{4.89}$$

4.5 Experimental Results

In order to evaluate and compare the performance of isoNMF with traditional NMF we picked 3 benchmark datasets that have been tested in the literature:

1. The CBCL faces database Figure 34(a,b) [39], used in the experiments of the original paper on NMF [55]. It consists of 2429 grayscale 19×19 images that they are hand aligned. The dataset was normalized as in [55].

2. The isomap statue dataset Figure 34(c) [40] consists of 698 64×64 synthetic face photographed from different angles. The data was downsampled to 32×32 with the Matlab *imresize* function (bicubic interpolation).

3. The ORL faces [42] Figure 34(d) presented in [37]. The set consists of 472 19×19 gray scale images that are not aligned. For visualization of the results we used the *nmfpack* code available on the web [43].

The results for classic NMF and isoNMF with k-neighborhood equal to 3 are presented in Figure 35, 36 and tables 2, 3. We observe that classic NMF gives always lower reconstruction error rates that are not that far away from the isoNMF. Classic NMF fails to preserve distances contrary to isoNMF that always does a good job in preserving distances. Another observation is that isoNMF gives more sparse solution than classic NMF. The only case where NMF has a big difference in reconstruction error is in the CBCL-face database when it is being preprocessed. This is mainly because the preprocessing distorts the images and spoils the manifold structure. If we don't do the preprocessing Figure 35(d), the reconstruction error of NMF and isoNMF are almost the same. We would also like to point that isoNMF scales equally well with the classic NMF.

In Figure 38 we see a comparison of the energy spectrums of classic NMF and isoNMF. We define the spectrum as

$$s_i = \frac{\sum_{l=1}^{N} W_{li}^2}{\sqrt{\sum_{l=1}^{M} H_{il}^2}}$$

This represents the energy of the component normalized by the energy of the prototype image generated by NMF/isoNMF. Although the results show that isoNMF is much more compact than NMF, it is not a reasonable metric. This is because the prototypes (rows of the H matrix are not orthogonal to each other. So in reality $\sum_{i=1}^{k} s_i < \sum_{i=1}^{N} \sum_{j=1}^{m} (WH)_{ij}^2$ and actually much smaller. This is because the dot product between the rows is not zero.

Table 2: Classic NMF, the relative root mean square error, sparsity and distance error for the four different datasets (cbcl normalized and plain, statue and orl)

classic NMF	cbcl norm.	cbcl	statue	orl
rec. error	22.01%	9.20%	13.62%	8.46%
sparsity	63.23%	29.06%	48.36%	46.80%
dist. error	92.10%	98.61%	97.30%	90.79%

Table 3: isoNMF, the relative root mean square error, sparsity and distance error for the four different datasets (cbcl normalized and plain, statue and orl)

isoNMF	cbcl norm.	cbcl	statue	orl
rec error	33.34%	10.16%	16.81%	11.77%
sparsity	77.69%	43.98%	53.84%	54.86%
dist. error	4.19%	3.07%	0.03%	0.01%

4.6 Summary

In this chapter we presented a deep study of the optimization problem of NMF, showing some fundamental existence theorems for the first time as well as various advanced optimization approaches – convex and non-convex, global and local.

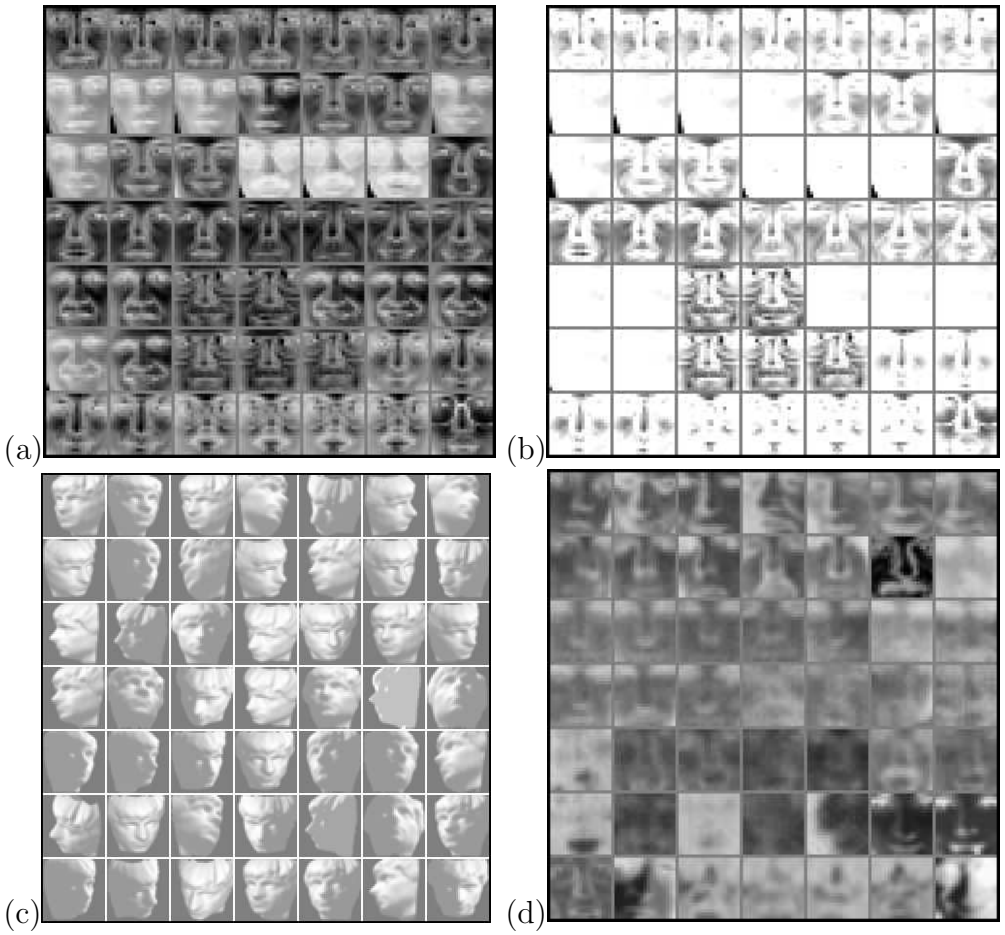


Figure 34: (a)Some images from the cbcl face database (b)The same images after variance normalization, mean set to 0.25 and thresholding in the interval [0,1] (c)The synthetic statue dataset from the isomap website [40] (d)472 images from the orl faces database [42]

We presented an algorithm that finds a local optimum of NMF by solving a sequence of semidefinite programs. We also used the same algorithm to solve the problem of Completely Positive Factorization. We also developed and experimentally demonstrated a new method, isoNMF, which preserves both non-negativity and isometry, simultaneously providing two types of interpretability. With the added reliability and scalability stemming from an effective optimization algorithm, we believe that this method represents a potentially valuable practical new tool for the exploratory analysis of common data such as images, text, and spectra.

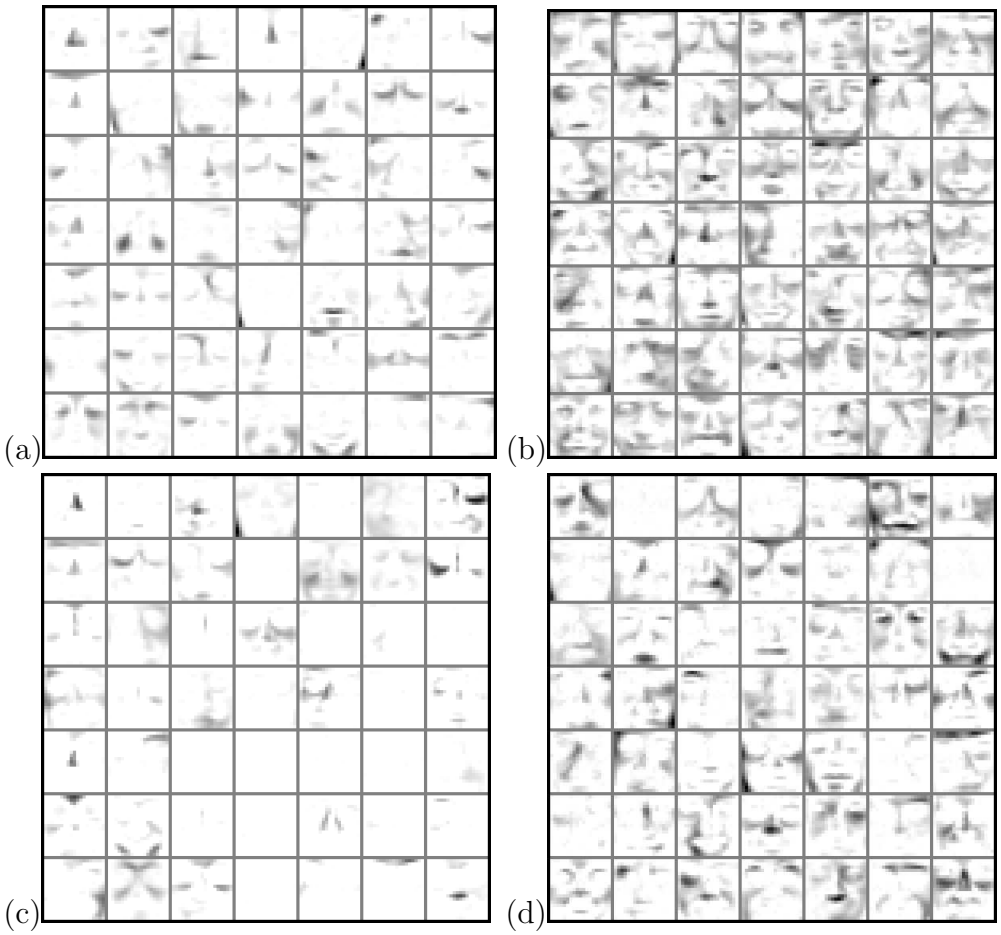


Figure 35: Top row: 49 Classic NMF prototype images. Bottom row: 49 isoNMF prototype images (a, c) CBCL-face database with mean variance normalization and thresholding, (b, d) CBCL face database without preprocessing.

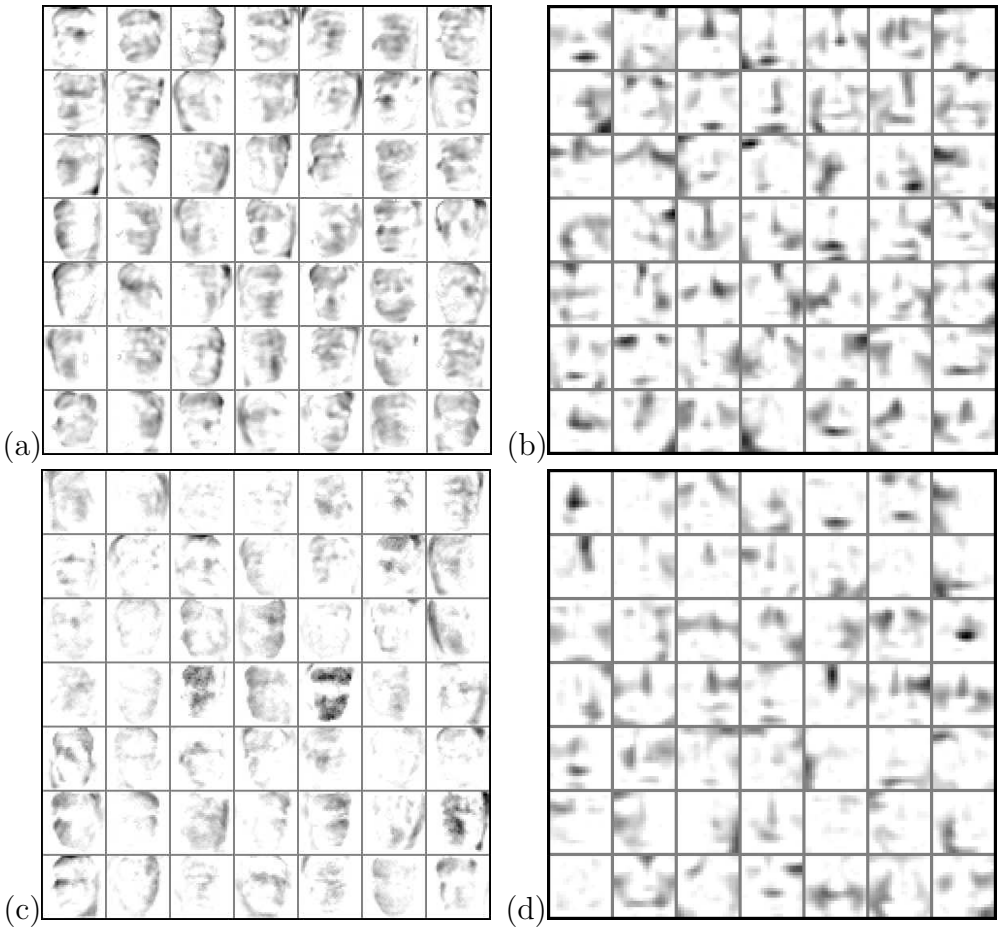
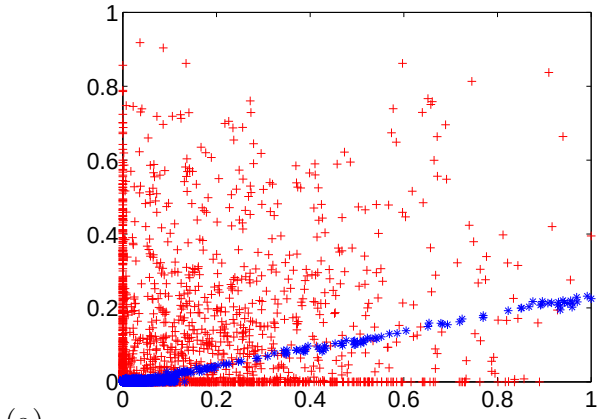
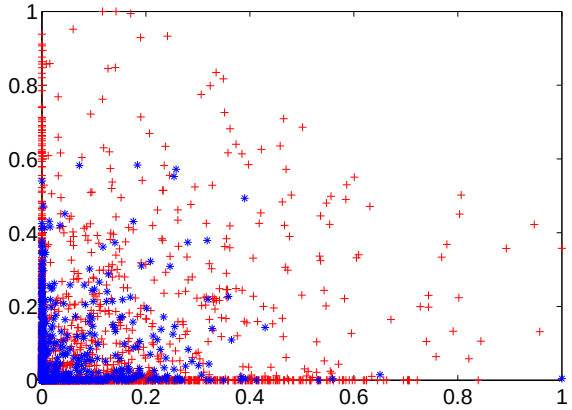


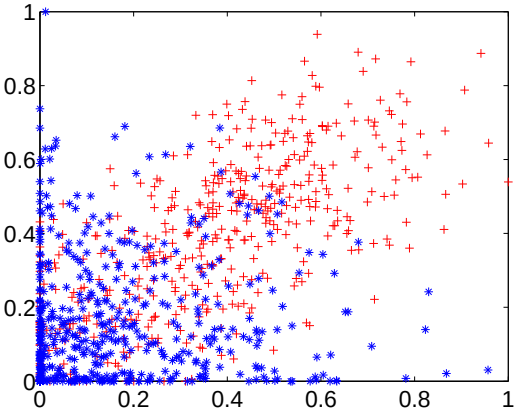
Figure 36: Top row: 49 Classic NMF prototype images. Bottom row: 49 isoNMF prototype images (a, c) statue database , (b, d) orl-faces database



(a)

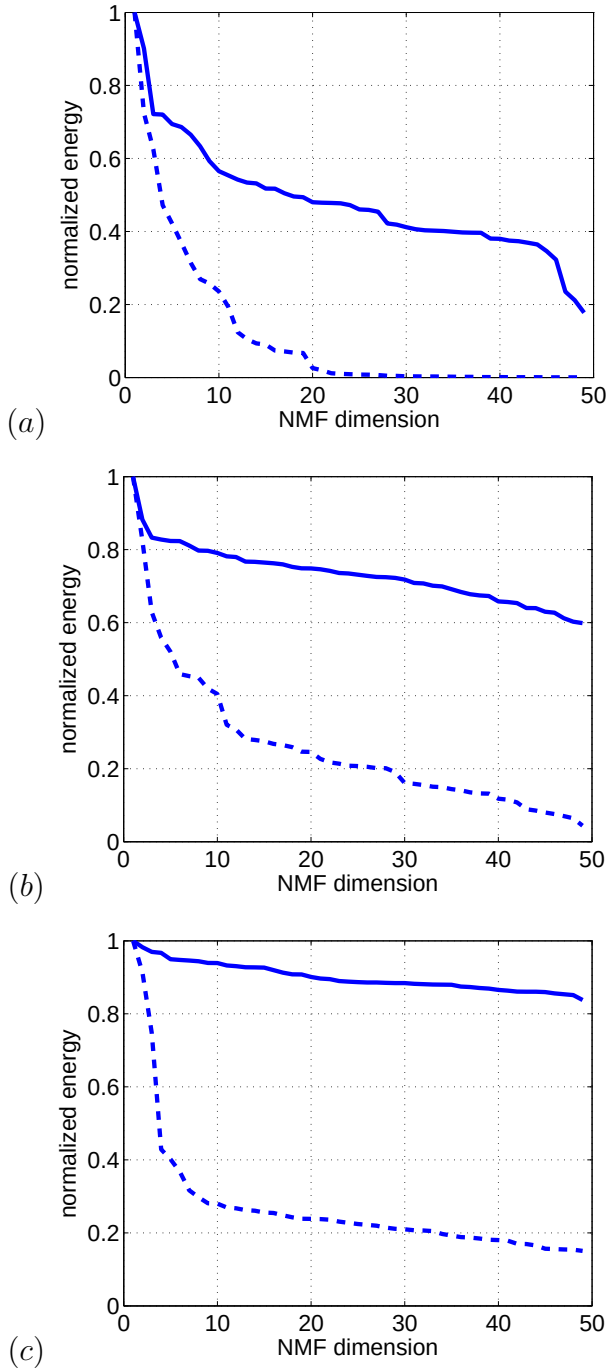


(b)



(c)

Figure 37: Scatter plots of two largest components of classic NMF(in blue) and Isometric NMF(in red) for (*a*)cbcl faces (*b*)isomap faces (*c*)orl faces



(a)

(b)

(c)

Figure 38: In this set of figures we show the spectrum of classic NMF (solid line) and Isometric NMF (dashed line) for the three datasets (a)cbcl face (b)isomap statue (c)orl faces. Although isoNMF gives much more compact spectrum we have to point that the basis functions are not orthogonal, so this Figure is not comparable to SVD type spectrums

CHAPTER V

LEARNING ISOMETRIC SEPARATION MAPS

Support Vector Machines (SVMs)have been quite successful in separating classes of data that are not linearly separable. The kernel trick lifts the data in a high dimensional Hilbert space usually of infinite dimension [79]. Embedding datasets in infinite dimensional spaces gives the advantage of separating data with linear hyperplanes in the lifted space, that otherwise would not be separable in the original space. So far it is not clear how the dimensionality of the kernel affects the performance of SVMs. It is not known yet how many dimensions are sufficient for separating the classes. It is very likely that the minimum dimension required for linear separability is much smaller than the original dimension of the data. This is because the data might already be embedded in a manifold with redundant dimensions.

Maximum Variance Unfolding (MVU) [95] along with other manifold learning methods has addressed the problem of reducing the dimensionality of the data by preserving local distances. Most of the time data end up living in a lower dimensional space. MVU explicitly finds the optimal kernel matrix for the data, by solving a semidefinite program as we saw in chapter 3. Although MVU usually gives the most compact spectrum [97, 57], it has very poor performance when it comes to using the kernel matrix for SVM classification [97] as it does not include any information about the linear separability of the classes. For example, in Figure 39b we show two classes on a Swiss roll manifold. After unfolding with MVU 39d, the classes remain non-linearly separable.

In [83] the authors introduced colored MVU as an algorithm that can find low dimensional embedding on data that are colored according to their class. It is cast as an optimization problem for minimizing the "user defined" class distance regularized by the local distance preservation. Depending on the weight of the regularizer the embedding dimension can change dramatically and the initial manifold geometry can be heavily distorted.

In this chapter we introduce a variation of MVU that takes into consideration the linear separability of the classes. The result is a new algorithm, Isometric Separation Mapping (ISM), that gives an unfolding that preserves the class structure of the manifold. The algorithm can be seen as a transductive (semi-supervised) SVM, since it requires the test data during training. Previous work on transductive SVMs has also been studied by several researchers. When the choice of the kernel is ad-hoc, the problem becomes very difficult as it boils down to mixed integer programming [7]. In [53] and [48] the authors train the kernel matrix over a set of predefined kernels. Although this gives higher flexibility in forming the kernel, it might still require a large number of predefined kernels. For example if one of the choices was the Gaussian, it would be necessary to keep a large number of them with different sigmas (bandwidths). It is widely known that if the bandwidth of the Gaussian is too wide or too narrow, kernel methods perform poorly. This technique usually leads to full rank semidefinite programs that are computationally hard. Finally, in [6] the Laplacian Eigenmap (LE) framework is used for training SVMs. Laplacian Eigenmaps are another dimensionality reduction method based on the Gaussian kernel. It also tries to capture the local geometry and take advantage of it in SVM training. Our technique does not make any assumption on the kernel function. The only requirement is to preserve isometry on the data.

The chapter is organized in the following way. In section 5.1 we present the ISM algorithm. Some examples on embedding manifolds with ISM are presented in 5.2. In section 5.3 we present a transductive SVM based on the ISM.

5.1 Isometric Separation Maps (ISM)

Although MVU and its variant MFNU give low-rank kernel matrices, experiments [97] show that they perform poorly when it comes to SVM classification. In this section we show that MVU/MFNU can be modified so that the kernel matrix can be

used for classification too.

In traditional SVMs the kernel is chosen ad-hoc and the goal is to find a hyperplane that can linearly separate the classes. The kernel is chosen in such a way that it lifts the data in a high-dimensional space, hoping that data would be linearly separable. In our approach we have the hyperplane given and we are trying to find the kernel matrix that separates the data along the hyperplane. Finding a kernel matrix to satisfy that condition is trivial, as it suffices to add one extra dimension on the data that will be either -1 or 1. What is interesting though is to find a mapping to a (higher or lower) dimensional space that keeps data points linearly separable and preserves the local isometry. As we will see later, depending on the structure of the classes, it is likely to end up in a higher dimensional space. We are interested in the minimum dimension of that space.

The solution of the problem is the following. We pick one of the data points x_A to be normal to the separating hyperplane. The choice of the point does not matter since it will just change the orientation of the points in space. The manifold consists of two classes C_1 and C_2. Let $x_i \in C_1$ be the points that belong in the same class with x_A, then $k(x_A, x_i) \geq 0$, where $k(x_A, x_i)$ is the generalized dot product between x_A and x_i. For points that belong to the opposite class $x_i \in C_2$, $k(x_A, x_i) \leq 0$. Now the problem of MVU/MFNU with linear separability constraints can be cast as the following Semidefinite Program:

$$\max_K \sum_{i=1}^{N} B_i \bullet K \tag{1.90}$$

subject to

$$\begin{aligned} A_{ij} \bullet K &= d_{ij} \quad \forall j \in I_i \\ K_{A,i} &\geq 0, \quad \forall i \in C_1 \\ K_{A,i} &\leq 0, \quad \forall i \in C_2 \\ K &\succeq 0 \end{aligned}$$

Using the same formulation as in chapter 3 (also published in [92]) we can solve the above problem in a non-convex framework that scales better. Extending the problem for more classes is pretty straightforward. The only modification is to use more anchor points that will serve as normal vectors to the separating hyperplanes. The problem is always feasible provided that $k \ll N$. As long as the k neighbors belong to the tangent space and the manifold is smooth, a folding (locally isometric transform) of the manifold along a hyperplane always exists [58]. If all pair distances are given then the Gram matrix is uniquely defined and the problem might be infeasible. In the trivial case where $k = 1$, meaning that each point has exactly one neighbor then the problem is always feasible. In general there is a maximum k where the problem might become infeasible. That means there is always a k where the training error is zero which means we can always find a dimensional space where the manifold can be embedded isometrically.

If some of the data points are labeled (training data) and some are not (test data), then the above method can be used as an SVM-like classifier that always achieves zero training error in contrast to other algorithms proposed for learning the kernel in SVM (mentioned in section 5 , where the kernel is learnt as a convex combination of preselected kernels). This might sound like over-fitting on the training data. In reality though this is not true since the test data participate during training glued on the training data with the distance constraints. Another remark on the ISM is that it is not a max margin classifier because it does not regularize the norm of the normal vector. It is not possible to do it since we need to also preserve the local distances.

5.2 Dimensionality Minimization with ISM

In order to verify ISM on dimensionality adjustment we tested it on the Swiss roll dataset (1500 points). Two classes were defined on the Swiss roll that were not linearly separable. ISM was performed on the dataset. Embedding in 2 dimensions

was not possible, as the isometry cannot be preserved (the algorithm terminated with 2% error on local distances). Embedding was possible in 3 dimensions where the algorithm terminated with 0.01% error in the local distance constraints. In both cases the classification error was zero. As we see in Figure 39, MVU unfolds the dataset in a strip where the classes are not linearly separable. The ISM on the other hand transforms the manifold in a set that preserves the local distances (k neighborhood=5) and divides the two classes in a linearly separable way. In order to further demonstrate the power of ISM, we test it in two even more complex cases. In Figure 40 we generated 3 classes on a Swiss roll. Clearly MVU/MFNU unfolds the manifold in a non-separable way. ISM was able to map the swiss roll in a 12-dimensional space where the 3 classes are completely linearly separable.

In Figure 41, the Principal Component Analysis (PCA) spectrum of the 12-dimensional Swiss roll is shown. The spectrum is quite rich. ISM can handle even more complicated cases. In Figure 42 we show 3 classes lying randomly on a Swiss roll. ISM was able to map the manifold in a 12-dimensional space, keeping the 3 classes linearly separable. In Figure 43 the PCA spectrum is depicted. The algorithm terminated with a very low feasibility error of 0.4% for distance preservation and 0.16% for linear separability. Further improvement of the feasibility error was possible, but L-BFGS becomes slow as it goes close to the optimum. In general the algorithm converges very quickly to 1% feasibility error. Further improvement is possible but takes time.

5.3 Transductive SVMs

The method described above can also be used as a transductive SVM in a semi-supervised setting. Transductive SVMs are in general difficult problems. If the kernel is preselected, then a mixed integer problem has to be solved. If the kernel is learned from the data as we mentioned earlier no guarantee can be given that the training

data will be linearly separable. In ISM the kernel is trained over all data, using all neighborhood information. After solving the optimization problem, the classification information for the test data will be on the sign of $K_{A,i}$ $\forall i \in T$, where T is the test set. In other words, the SVM will try to find a hyperplane that separates the data in the best way. ISM will pick a hyperplane and it will try to find a homeomorphic (unfolding or folding) transformation of the data such that all the positive classes are on the same part of the hyperplane and the negatives on the other one, Figure **??**.

In order to evaluate ISM as an SVM classifier, we chose a publicly available dataset and compared it versus traditional SVMs in two different modes. We used the publicly available *SVM-light* [41] software for traditional SVM classification. In the first experiment we picked 1000 points from the magic gamma telescope dataset, publicly available at the UCI repository [38]. We chose 50 points as training points and used the other 950 as test points. For traditional SVM classification we tested the linear, Gaussian and polynomial kernel, with different parameters for the bandwidth and the polynomial order. We also tuned the regularization factor so that the test error was minimized. In other words we pushed traditional SVMs to their best performance. The critical parameter for ISM SVM is the k-neighborhood. Usually, small values of k allow embedding in lower dimensional spaces, while large k allow embedding in higher dimensional ones. In tables 4 and 5 the results are summarized. We tested several k-neighborhoods for ISM and different kernels for traditional SVMs. From the results we observe that ISM behaves slightly better that SVM (73.68% versus 70.32%). This is mainly because the training set is small and SVM cannot capture the geometry very well.

In the second experiment we use the whole dataset. The training set contains 12080 datapoints and the test set 6340. Although the dataset is 10 dimensional, it is possible to reduce its dimension with MVU/MFNU down to 5. In order to make it linearly separable though with ISM, it was necessary to use more than 10. In

tables 6 and 7 the results are summarized. As we can see, SVM performs slightly better than ISM (83.28% versus 81.00%). Another remark in both cases is that ISM always behaves better than the linear kernel. Gaussian SVM performance has the best performance. This is expected since Gaussian matrices are usually full rank. ISM uses kernel matrices of much smaller rank and they achieve equivalent performance.

The results don't necessarily demonstrate a big difference between SVMs and ISM. We also experimented with some toy datasets, such as the half moon dataset presented in [6] and a Swiss roll, where one data point is given per class. ISM obviously behaves better than SVM, but this is trivial and not a fair comparison. In general the differences between ISM and traditional SVMs are on the same levels with the results reported in other transductive SVM papers [6, 53]. In practice ISMs are slower than SVMs since they are semidefinite problems contrary to SVMs that solve quadratic problems. It is interesting that they provide a tool to associate the dimensionality of the dataset with the classification score and linear separability. The more we increase the dimensionality of the dataset with ISM, the better the classification score. In fact, k acts as a regularizer. Large values of k correspond to better generalization of the SVM as the test error drops.

5.4 Summary

In this chapter we presented a new Manifold Learning method the Isometric Separation Maps. This method is ideal for reducing the dimension of the manifold with the class information associated with them. We also showed how ISM can be used as semi-supervised (transductive) classifiers. Although they don't have superior performance compared to traditional max margin SVMs, they are a useful tool for determining the dimensionality of the kernel space that is necessary for achieving linear separability. We believe that some improvement of the objective function is necessary so that generalization is improved. Probably a term minimizing the norm

Table 4: ISM SVM classification score versus k-neighborhood for the First Experiment

k-NEIGHBORS	DIMENSION	SCORE
5	50	70.10%
8	10	68.73%
10	12	70.63%
15	8	70.42%
15	12	69.05%
20	8	71.68%
20	12	71.68%
20	40	73.37
25	8	72.63%
25	12	71.89%
30	**40**	**73.68%**

Table 5: Traditional SVM Classification Score versus k-neighborhood

KERNEL	PARAMETER	SCORE
Gaussian	0.1	69.89%
Gaussian	0.5	70.00%
Gaussian	1.0	70.11%
Gaussian	1.5	70.00%
Gaussian	2.0	70.11%
Gaussian	4.0	70.21%
Gaussian	**5.0**	**70.32%**
Gaussian	6.0	70.21%
Gaussian	8.0	70.11%
linear	-	69.89%
polynomial	1	69.89%
polynomial	2	69.68%
polynomial	3	69.58%
polynomial	4	69.84%
polynomial	5	68.84%
polynomial	6	68.84%
polynomial	8	68.95%

Table 6: ISM SVM Classification Score versus k-neighborhood For the Whole Dataset

k-NEIGHBORS	DIMENSION	SCORE
12	30	80.22%
12	35	79.97%
12	40	80.47%
12	45	79.76%
12	50	79.81%
12	**55**	**81.00**%
15	40	80.39%
15	45	79.40%
15	50	79.07%
15	55	79.82%
20	40	78.96%
20	45	79.68%
20	50	80.13%
20	55	78.42%

Table 7: ISM SVM Classification Score versus k-neighborhood For the Whole Dataset

KERNEL	PARAMETER	SCORE
Gaussian	**8**	**83.28%**
Gaussian	6	82.77%
linear	-	78.64%
polynomial	2	81.62%
polynomial	3	82.07%
polynomial	5	81.26%

of the vector normal to the hyperplane (as in SVMs) can be used.

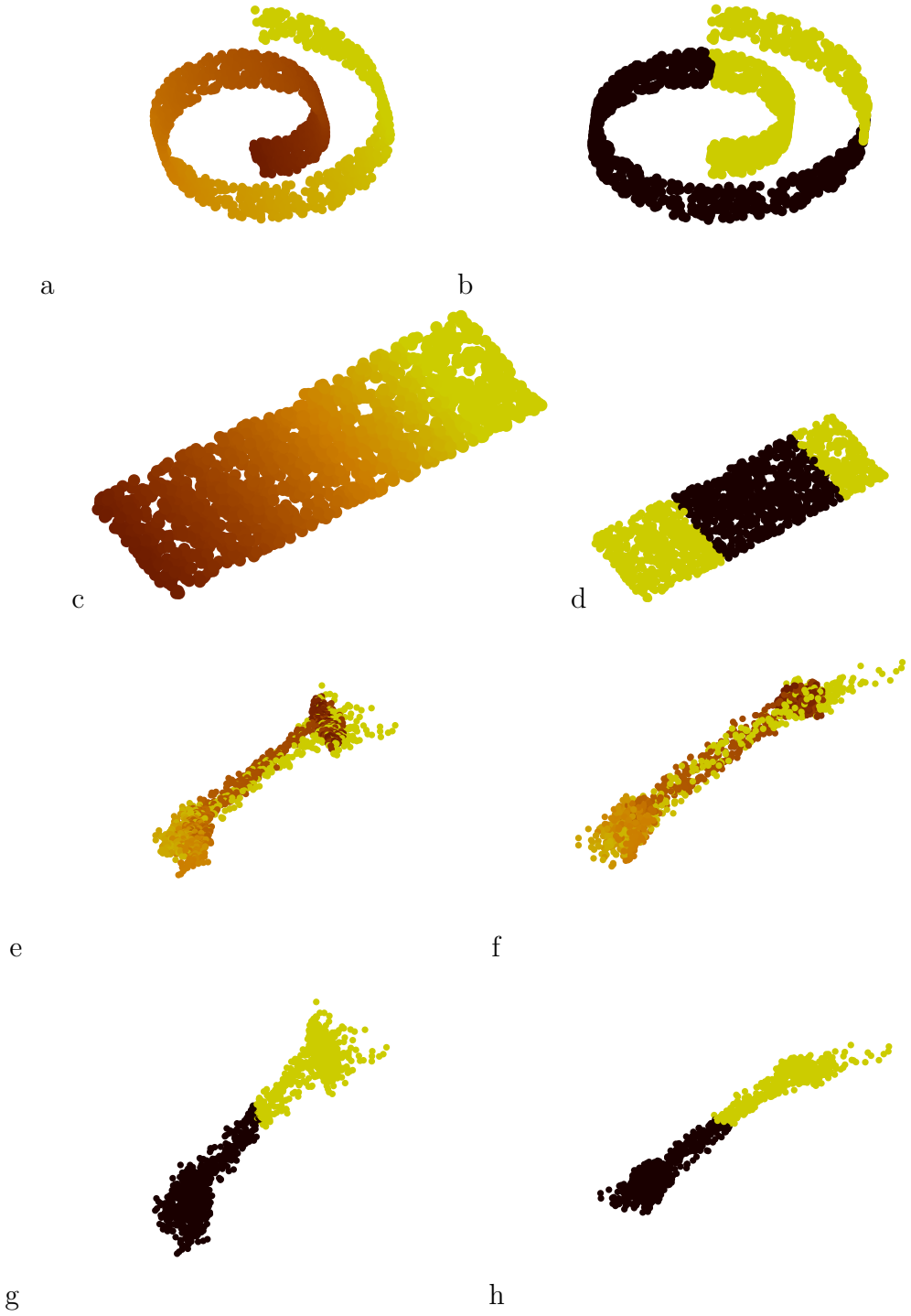


Figure 39: a)A three dimensional swiss roll painted with color gradient. b)The same swiss roll with two classes on it, black and green c)Unfolded swiss roll (a) with MVU/MFNU (no class information). The color gradient shows that local distances has been preserved. d) Unfolded swiss roll (b) with MVU/MFNU. The two classes are not linearly separable. e,f) Views of the swiss roll (a) with ISM. The class structure was taken from (b). The intension of this Figure is to show how the points are mapped so that the local neighborhoods are preserved. g,h)Views of the (b) manifold after ISM. Now points are painted with the class colors to show that they are linearly separable

Figure 40: Top: Three classes laying on a swiss roll. Bottom: After unfolding them with MVU the classes are not linearly separable. Isometric Separation Maps managed to map this manifold in a 12-dimensional space such that the classes were linearly separable by 3 hyperplanes 100% of the time and the 5-neighborhood distances were preserved with 0.1% relative root mean square error

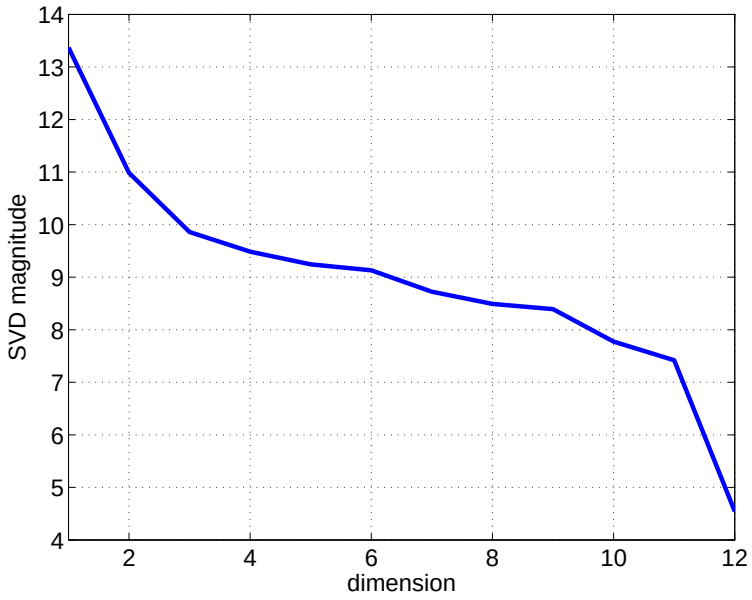


Figure 41: In this Figure we illustrate the PCA (SVD) spectrum of the unfolded swiss roll of Figure 40. As we can see it is pretty rich.

Figure 42: Top: Three classes laying randomly on a swiss roll. Bottom: After unfolding them with MVU the classes are not linearly separable. Isometric Separation Maps managed to map this manifold in a 12-dimensional space such that the classes were linearly separable by 3 hyperplanes. The optimization algorithm terminated with feasibility error 0.4% for 5-neighborhood distance preservation, while 99.83% of the points were correctly classified. The goal of this experiment was to verify experimentally that ISM can lift any strange dataset to a high dimensional space, such that classes are linearly separable

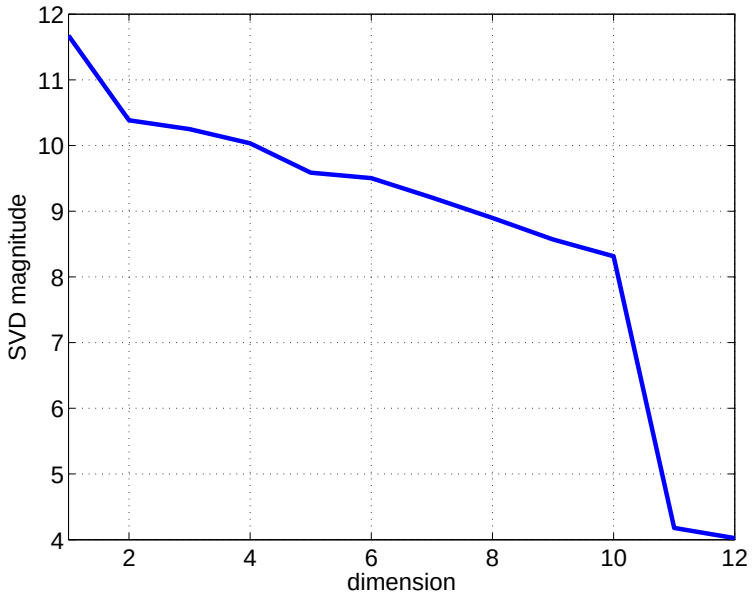


Figure 43: In this Figure we illustrate the PCA (SVD) spectrum of the unfolded swiss roll of Figure 42. As we can see it is pretty rich. Despite the bad structure of the classes, the ISM algorithm was able to map it on a 12 dimensional space.

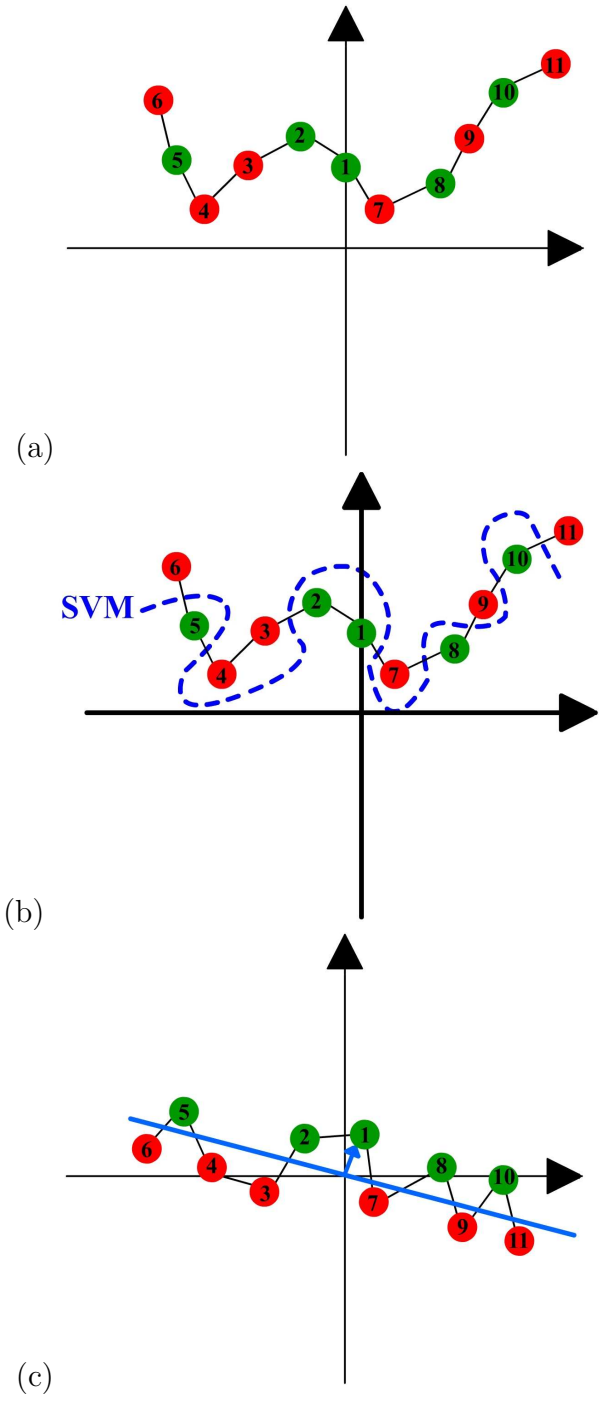


Figure 44: (a) A trivial one dimensional manifold (b) The separation hyperplane from an SVM with a gaussian kernel (c) ISM will do a homeomorphic transformation on the manifold so that a linear hyperplane can do perfect separation

CHAPTER VI

NONLINEAR MATRIX FACTORIZATIONS, A GENERAL FRAMEWORK FOR DIMENSIONALITY REDUCTION

In the previous chapters we examined two dimensionality reduction techniques MVU and NMF along with different extensions. In this chapter we will show how it is possible to put all of them under a general framework of nonlinear matrix factorizations with the use of sequential convex optimization.

6.1 Matrix factorizations and dimensionality reduction

A matrix $V \in \Re^{N \times m}, N > m$ represents a collection of N $m-$dimensional points on a Euclidean space, where the basis vectors are orthogonal. We can always express V as a product of two matrices $V = WH$ where $W = V$ and $H = I$ (I is the identity matrix). Linear dimensionality reduction methods either try to find an exact factorization of the form

$$V = WH \tag{1.91}$$

or an approximate one

$$V \simeq WH \tag{1.92}$$

so that $W \in \Re^{N \times k}, H \in \Re^{k \times m}$ where $k < m$. The H matrix carries the information about the prototypes (also known as basis vectors) and the W matrix is the new coordinate vectors.

Nonlinear dimensionality reduction methods apply a nonlinear transformation $f : \Re^{N \times m} \rightarrow \Re^{M \times l}$ on the initial matrix V and then apply a nonlinear operator $g : \Re^{N \times m} \rightarrow \Re^{M \times l}$ on the exact factorization:

$$f(V) = g(W, H) \tag{1.93}$$

or in the case of approximation:

$$||f(V) - g(W, H)||_L \leq \epsilon \tag{1.94}$$

where $|| \cdot ||_L$ is a Euclidean norm and ϵ is user defined.

For example in Collaborative Filtering [61, 85, 84] not all entries of V are known, so the operator f selects the given ones. In this case $g \equiv f$. Moreover in this case there

is not a unique solution but a set of matrices (W, H) that satisfy either equation 1.93 or 1.94. In practice (W, H) are found through an optimization program that tries to optimize other properties of (W, H) in order to get an improved and unique solution.

Another example is MVU, where operator f constructs a distance matrix out of V and selects only the local distances. Operator g solves a Semidefinite program that tries to find a Gram matrix that matches the selected distances and then by using SVD it finds W and assumes H to be the identity matrix I. All manifold methods use the same model with only difference that the g operator is a linear one. MVU and its variants are an exception because they use a sophisticated optimization program.

6.2 Casting MVU, NMF and more as a general rank-constrained semidefinite program

In this section we will show how MVU, NMF and a big family of nonlinear factorizations can be cast as the same rank-constrained semidefinite program.

Consider the following positive semidefinite matrix:

$$Z = \begin{bmatrix} I_k & W^T & H^T \\ W & \mathcal{W} & \mathcal{V}^T \\ H & \mathcal{V} & \mathcal{H} \end{bmatrix} \succeq 0 \tag{2.95}$$

if we constraint Z to have $\text{rank}(Z) = k$ then

$$\mathcal{W} = WW^T, \quad \mathcal{H} = HH^T, \quad \mathcal{V} = WH^T \tag{2.96}$$

If the nonlinear transformations f, g can be cast as linear constraints or convex inequalities on the Z matrix then the factorization

$$f(V) = g(W, H) \tag{2.97}$$

can be expressed and found through the following algorithm.

As we show in chapter 4 given any convex constraints on Z such that $\text{rank}(Z) \leq k$, Z can be found with the algorithm presented by Dattorro [19]. The general form of a $k-$rank factorization is:

$$\min_{Z} h(Z) \tag{2.98}$$

such that:

$$l_i(Z) \leq 0$$

$$A \bullet Z = 0$$

$$Z \succeq 0$$

$$\text{rank}(Z) \leq k$$

where $h(Z), l_i(Z)$ are convex functions that admit a conic representation. This problem is not convex due to the rank constraint but a local solution can be found if the problem is feasible with the algorithm shown in 45.

1. set $G^* = I$, $\epsilon = \epsilon_0$ and $\alpha = 1e-4$

2. Solve the following optimization program:

$$\min_{Z} Z \bullet G^* \tag{2.99}$$
$$\text{such that:}$$
$$h(Z) \leq \epsilon$$
$$l_i(Z) \leq 0$$
$$A \bullet Z = b_i$$
$$Z \succeq 0$$
$$\text{rank}(Z) \leq k$$

3. Let Z^* be the optimal Z, then solve the following program:

$$\min_{G} Z^* \bullet G \tag{2.100}$$
$$\text{such that:}$$
$$I \succeq G \succeq 0$$
$$\text{Trace}(G) = N + m$$

4. Let G^* be the optimal G, then if:

$$|G^* \bullet Z^*| \leq \alpha \tag{2.101}$$

terminate, Z^* is the solution of the problem else if:

$$|G^* \bullet Z^*| \geq \alpha \tag{2.102}$$

and no significant improvement has been done in reducing the $G^* \bullet Z^*$ from the previous problem has been done, then the problem is infeasible, go to step 1 and increase ϵ. Otherwise go to step 2.

Figure 45: Nonlinear Convex Factorization

6.2.1 MVU as a special case

It is straight forward to see that the MVU can now be expressed in the following way

1. set $G^* = I$, $\epsilon = \epsilon_0$ and $\alpha = 1e-4$

2. Solve the following optimization program:

$$\min_{Z} Z \bullet G^* \tag{2.103}$$

such that:

$$-\text{Trace}(Z(k+1:k+N, k+1:k+N)) \leq -\epsilon$$

$$\sum Z(k+1:k+N, k+1:k+N) = 0$$

$$A_{ij} \bullet Z(k+1:k+N, k+1:k+N) = d_{ij}{}^{1}$$

$$Z \succeq 0$$

$$\text{rank}(Z) \leq k$$

3. Let Z^* be the optimal Z, then solve the following program:

$$\min_{G} Z^* \bullet G \tag{2.104}$$

such that:

$$I \succeq G \succeq 0$$

$$\text{Trace}(G) = N + m$$

4. Let G^* be the optimal G, then if:

$$|G^* \bullet Z^*| \leq \alpha \tag{2.105}$$

terminat, Z^* is the solution of the problem else if:

$$|G^* \bullet Z^*| \geq \alpha \tag{2.106}$$

and no significant improvement has been done in reducing the $G^* \bullet Z^*$ from the previous problem has been done, then the problem is probably infeasible, go to step 1 and decrease ϵ. Otherwise go to step 2.

[1] d_ij are the distances between nearest neighbors

The above problem might be infeasible if k is too small. Typically one should start with large k, no more than m and reduce it until the above problem becomes infeasible. This is the minimum k. Of course this formulation is more complicated than the original MVU [97] one since is requires the solution of more than one semidefinite programs. On the other hand we should not forget that maximization of variance is just a heuristic to reduce the rank. As Dattorro [19] suggested isometric embedding can be done straight forward with the above algorithm without the maximization of the variance. A similar approach for rank reduction of the dot product matrix WW^T was suggsted in [78] where a heuristic for reducing the rank was introduced.

6.2.2 LOOCV unfolding as a special case

In section 3.6.1 we presented an algorithm for maximizing the cross-validation for maximizing the Leave One Out Cross Validation (LOOCV). Here we present how this algorithm can be a special form of the general factorization:

1. set $G^* = I$, $\epsilon = \epsilon_0$ and $\alpha = 1e-4$

2. Solve the following optimization program:

$$\min_{Z} Z \bullet G^* \tag{2.107}$$

such that:

$$-\sqrt[N]{\prod_{i=1}^{N}\sum_{j\neq i} Z(k+i,k+i)} \leq -\epsilon \quad ^2$$

$$Z(k+1:k+N,k+1:k+N) \geq 0$$

$$A_{ij} \bullet Z(k+1:k+N,k+1:k+N) = d_{ij}{}^3$$

$$Z \succeq 0$$

$$\text{rank}(Z) \leq k$$

[3] In order to represent this constraint with positive semidefinite constraints we need to introduce approximately $\log_2 N$ auxiliary variables t.

[3] d_ij are the distances between nearest neighbors

3. Let Z^* be the optimal Z, then solve the following program:

$$\min_G Z^* \bullet G \tag{2.108}$$

such that:

$$I \succeq G \succeq 0$$

$$\text{Trace}(G) = N + m$$

4. Let G^* be the optimal G, then if:

$$|G^* \bullet Z^*| \leq \alpha \tag{2.109}$$

terminat, Z^* is the solution of the problem else if:

$$|G^* \bullet Z^*| \geq \alpha \tag{2.110}$$

and no significant improvement has been done in reducing the $G^* \bullet Z^*$ from the previous problem has been done, then the problem is probably infeasible, go to step 1 and decrease ϵ. Otherwise go to step 2.

6.2.3 NMF as a special case

We have already shown in chapter 4 how NMF can fit in that framework. For the shake of completeness we repeat the algorithm here:

1. set $G^* = I$, $\epsilon = \epsilon_0$ and $\alpha = 1e - 4$

2. Solve the following optimization program:

$$\min_{Z} Z \bullet G^* \tag{2.111}$$

such that:

$$||Z(k+1:k+N, k+N+1:k+N+m) - V||_2^2 \leq \epsilon$$

$$Z(k+1:k+N, 1:k) \geq 0$$

$$Z(k+N+1:k+N+m, 1:k) \geq 0$$

$$Z \succeq 0$$

$$\text{rank}(Z) \leq k$$

3. Let Z^* be the optimal Z, then solve the following program:

$$\min_{G} Z^* \bullet G \tag{2.112}$$

such that:

$$I \succeq G \succeq 0$$

$$\text{Trace}(G) = N + m$$

4. Let G^* be the optimal G, then if:

$$|G^* \bullet Z^*| \leq \alpha \tag{2.113}$$

terminat, Z^* is the solution of the problem else if:

$$|G^* \bullet Z^*| \geq \alpha \tag{2.114}$$

and no significant improvement has been done in reducing the $G^* \bullet Z^*$ from the previous problem has been done, then the problem is probably infeasible, go to step 1 and increase ϵ. Otherwise go to step 2

6.3 Extending to factorizations with any nonlinear dot product or even divergence

In this section we show how it is possible to solve factorization problems of the form 1.93 or 1.94 when g admits a polynomial approximation:

$$g(W_i, H_j) = \sum_{i=1}^{n} \sum_{\sum a_k + b_k = i} \prod_{k=1}^{m} w_{ik}^{a_k} \prod_{k=1}^{m} h_{kj}^{b_k} \tag{3.115}$$

where W_i, H_j are rows of the columns of the corresponding matrices and w_{ik}, h_{kj} are matrix elements.

In chapter 4 section 4.2 we show a trick for constructing a semidefinite matrix that has all possible terms up to second order. Consider the following example positive semidefinite matrix:

$$Z = \begin{bmatrix}
1 & w_{11} & w_{12} & w_{13} & w_{21} & w_{22} & w_{23} & w_{31} & w_{32} & w_{33} & h_{11} & h_{12} & h_{21} & h_{22} \\
w_{11} & w_{11}^2 & w_{11}\,w_{12} & w_{11}\,w_{13} & w_{11}\,w_{21} & w_{11}\,w_{22} & w_{11}\,w_{23} & w_{11}\,w_{31} & w_{11}\,w_{32} & w_{11}\,w_{33} & w_{11}\,h_{11} & w_{11}\,h_{12} & w_{11}\,h_{21} & w_{11}\,h_{22} \\
w_{12} & w_{11}\,w_{12} & w_{12}^2 & w_{12}\,w_{13} & w_{12}\,w_{21} & w_{12}\,w_{22} & w_{12}\,w_{23} & w_{12}\,w_{31} & w_{12}\,w_{32} & w_{12}\,w_{33} & w_{12}\,h_{11} & w_{12}\,h_{12} & w_{12}\,h_{21} & w_{12}\,h_{22} \\
w_{13} & w_{11}\,w_{13} & w_{12}\,w_{13} & w_{13}^2 & w_{13}\,w_{21} & w_{13}\,w_{22} & w_{13}\,w_{23} & w_{13}\,w_{31} & w_{13}\,w_{32} & w_{13}\,w_{33} & w_{13}\,h_{11} & w_{13}\,h_{12} & w_{13}\,h_{21} & w_{13}\,h_{22} \\
w_{21} & w_{11}\,w_{21} & w_{12}\,w_{21} & w_{13}\,w_{21} & w_{21}^2 & w_{21}\,w_{22} & w_{21}\,w_{23} & w_{21}\,w_{31} & w_{21}\,w_{32} & w_{21}\,w_{33} & w_{21}\,h_{11} & w_{21}\,h_{12} & w_{21}\,h_{21} & w_{21}\,h_{22} \\
w_{22} & w_{11}\,w_{22} & w_{12}\,w_{22} & w_{13}\,w_{22} & w_{21}\,w_{22} & w_{22}^2 & w_{22}\,w_{23} & w_{22}\,w_{31} & w_{22}\,w_{32} & w_{22}\,w_{33} & w_{22}\,h_{11} & w_{22}\,h_{12} & w_{22}\,h_{21} & w_{22}\,h_{22} \\
w_{23} & w_{11}\,w_{23} & w_{12}\,w_{23} & w_{13}\,w_{23} & w_{21}\,w_{23} & w_{22}\,w_{23} & w_{23}^2 & w_{23}\,w_{31} & w_{23}\,w_{32} & w_{23}\,w_{33} & w_{23}\,h_{11} & w_{23}\,h_{12} & w_{23}\,h_{21} & w_{23}\,h_{22} \\
w_{31} & w_{11}\,w_{31} & w_{12}\,w_{31} & w_{13}\,w_{31} & w_{21}\,w_{31} & w_{22}\,w_{31} & w_{23}\,w_{31} & w_{31}^2 & w_{31}\,w_{32} & w_{31}\,w_{33} & w_{31}\,h_{11} & w_{31}\,h_{12} & w_{31}\,h_{21} & w_{31}\,h_{22} \\
w_{32} & w_{11}\,w_{32} & w_{12}\,w_{32} & w_{13}\,w_{32} & w_{21}\,w_{32} & w_{22}\,w_{32} & w_{23}\,w_{32} & w_{31}\,w_{32} & w_{32}^2 & w_{32}\,w_{33} & w_{32}\,h_{11} & w_{32}\,h_{12} & w_{32}\,h_{21} & w_{32}\,h_{22} \\
w_{33} & w_{11}\,w_{33} & w_{12}\,w_{33} & w_{13}\,w_{33} & w_{21}\,w_{33} & w_{22}\,w_{33} & w_{23}\,w_{33} & w_{31}\,w_{33} & w_{32}\,w_{33} & w_{33}^2 & w_{33}\,h_{11} & w_{33}\,h_{12} & w_{33}\,h_{21} & w_{33}\,h_{22} \\
h_{11} & w_{11}\,h_{11} & w_{12}\,h_{11} & w_{13}\,h_{11} & w_{21}\,h_{11} & w_{22}\,h_{11} & w_{23}\,h_{11} & w_{31}\,h_{11} & w_{32}\,h_{11} & w_{33}\,h_{11} & h_{11}^2 & h_{11}\,h_{12} & h_{11}\,h_{21} & h_{11}\,h_{22} \\
h_{12} & w_{11}\,h_{12} & w_{12}\,h_{12} & w_{13}\,h_{12} & w_{21}\,h_{12} & w_{22}\,h_{12} & w_{23}\,h_{12} & w_{31}\,h_{12} & w_{32}\,h_{12} & w_{33}\,h_{12} & h_{11}\,h_{12} & h_{12}^2 & h_{12}\,h_{21} & h_{12}\,h_{22} \\
h_{21} & w_{11}\,h_{21} & w_{12}\,h_{21} & w_{13}\,h_{21} & w_{21}\,h_{21} & w_{22}\,h_{21} & w_{23}\,h_{21} & w_{31}\,h_{21} & w_{32}\,h_{21} & w_{33}\,h_{21} & h_{11}\,h_{21} & h_{12}\,h_{21} & h_{21}^2 & h_{21}\,h_{22} \\
h_{22} & w_{11}\,h_{22} & w_{12}\,h_{22} & w_{13}\,h_{22} & w_{21}\,h_{22} & w_{22}\,h_{22} & w_{23}\,h_{22} & w_{31}\,h_{22} & w_{32}\,h_{22} & w_{33}\,h_{22} & h_{11}\,h_{22} & h_{12}\,h_{22} & h_{21}\,h_{22} & h_{22}^2
\end{bmatrix} \tag{3.116}$$

if the rank of Z is one then:

$$Z = zz^T \tag{3.117}$$

where

$$z = \begin{bmatrix} 1 \\ w_{11} \\ w_{12} \\ w_{13} \\ w_{21} \\ w_{22} \\ w_{23} \\ w_{31} \\ w_{32} \\ w_{33} \\ h_{11} \\ h_{12} \\ h_{21} \\ h_{22} \end{bmatrix} \tag{3.118}$$

The iterative algorithm described in 45 can be applied for this matrix Z and the rank constraint is equal to 1. The matrix contains all the monomials up to order 2 and with the appropriate linear constraints we construct all the values of $g(W, H)$. In general if V is $N \times m$ and the factorization is of order k, then Z is $(N+m)k+1 \times (N+m)k+1$ matrix. If we want to add terms of higher order we need to construct an auxiliary positive semidefinite matrix Z' $((N+m)^2k^2+1 \times (N+m)^2k^2+1)$ that is rank one, the extra constraints for Z' are that

$$Z'(1, i(N+m)+j) = Z(i,j), \quad 1 \leq i,j \leq (N+m)k+1 \tag{3.119}$$

In other words the first row/column of Z' is Z unfolded. With this trick we can construct super large matrices that contain any monomial orders we want. Of course the size of the matrix grows very quickly.

6.4 Moving even further, automatic construction of the nonlinear operator g

Up to now we have shown how a factorization of the form:

$$||f(V) - g(W,H)||_L \leq \epsilon$$

where g is known and can be approximated with a polynomial expansion. The polynomial expansion can either be Taylor or a different one. What is more interesting though is to find g along with W, H on the same time. The requirement is that g has a polynomial expansion approximation.

This can actually be done using the algorithm described in 45. All we need is to define the constraint on the optimization matrices. We show in the previous section how to construct matrices that contain polynomial terms of W, H. For simplicity we will consider order 2 polynomials defined. The PSD matrix Z has been defined in equation 3.116. We know define the PSD matrix Z' $((N+m)^2k^2+p \times (N+m)^2k^2+p)$, where p is the number of the coefficients required for the polynomial expansion. The necessary constraint is that:

$$Z'(1, i*(N+m)+j) = Z(i,j), \quad 1 \leq i,j \leq (N+m)k+1 \tag{4.120}$$

In other words the first line of Z' contains Z unfolded into a vector, plus the coefficients of the polynomial. As with Z, Z' has to be constrained to have rank one. Inside Z' products between the coefficients and the monomials can be found and they can be selected with linear constraints so that they sum up to the desired value.

6.5 Summary

In this chapter we gave a universal framework for non-linear factorizations, based on sequential convex programming. The theoretical approach shows that it is possible to do matrix factorizations with any nonlinear function that admits polynomial expansion. In fact it is possible to find the nonlinear mapping and the factors at the same time. Although only local convergence can be guaranteed the heuristic for rank reduction seems to work well in practice. The algorithms are based on convex programs which have polynomial complexity in terms of computation and in terms of memory, they are impractical. More research has to be done towards approximations that can make the solution of the optimization programs computationally feasible in

practice.

CHAPTER VII

DIMENSIONALITY REDUCTION OF LARGE SPEECH CORPORA

Much of research has been done in speech recognition over the last three decades and several schemes have been proposed. Most of the speech recognition engines share a common representation for speech, the Mel-Frequency Cepstrum Coefficients (MFCC). It is the most popular feature for speech recognition for over 20 years. In practice 13 MFCCs are extracted on a time window of 10-20msec and their first and second delta coefficients with smoothing are computed [71]. In total the dimensionality of the features used in speech recognition is around the value of 39. Despite its success, there is not much work done on its statistical properties, such as its dimensionality over large speech datasets, such as TIMIT [77]. Only recently some researchers have presented theoretical [45] and experimental [5, 81] work on embedding speech in a lower dimensional space. Unfortunately most of the experiments were performed on small speech datasets (few hundred data points). It is expected that speech data points coming from a few sentences from the same speaker will be highly correlated and they would lie on a lower dimensional manifold. Building a graph from these data points results in a graph with at least two major clusters, due to the voiced and unvoiced structure of speech. Moreover the methods that they used [5] (Laplacian Eigenmaps) are not suitable for discovering the intrinsic dimension of the data as indicated in a comparison of the methods in [97, 57]. It is shown in [81] that it is possible to improve speech recognition results by nonlinearly projecting speech features on a lower dimensional space. Motivated by these results we want to investigate the performance of the state-of-the-art dimensionality reduction technique on MFCC features. Our goal is to nonlinearly project the whole TIMIT dataset on a lower dimensional space. The challenge is to estimate the intrinsic dimensionality of the lower dimensional space and also recover the intrinsic dimensions. It is very interesting to find out what kind of information about speech these dimensions contain. The biggest challenge, though, is the computational part, since the dataset a prohibitive n(i.e., 1.5 million 39-dimensional vectors).

In this chapter we apply the MFNU to a data set with 1 million datapoints from the TIMIT speech recognition database. TIMIT contains 1.5M data points (train and test set) and it is computationally heavy to apply any Manifold Learning method on the whole dataset. The computation of local neighborhoods is the major bottleneck. Applying the state-of-the-art all k-nearest neighbor method for 1M points took 3.5 days, while naive computation would take more than 100 days. Our results show that the 300K and 1M point subsets of TIMIT can be embedded in 15 dimensions by preserving 10-point neighborhoods. As a comparison to PCA we mention that the first 15 components correspond to less than 50% percent of the spectral energy.

The chapter is organized in the following order. In section 7.1 we give information about MFCC feature generation. In section 7.2 we discuss the dimensionality analysis with Principal Component Analysis (PCA). Finally in Section 7.3 we discuss the results of our experiments.

7.1 MFCC Features

In our implementation for MFCC computation the freely available code from *Voicebox* was used [13]. The speech signal was divided into frames of length 25 msec, with 50% overlap and 13 MFCC coefficients (without the zeroth coefficient) where extracted from each frame. Twenty-seven filters were used to group the frequency bins into critical bands. The delta and delta-delta components are also computed increasing the dimensionality to 39. We examine both cases a)variance normalization on the MFCC components on every sentence [59] b)no variance normalization at all.

7.2 Principal Component Analysis

PCA is a scalable technique that can detect components that are linearly dependent. Given a set of N $d-$dimensional points $X \in \Re^{N \times d}$ centered around zero, the covariance matrix $C \in \Re^{d \times d}$ is:

$$C = X^T X \tag{2.121}$$

The eigenvalue decomposition $C = U^T \Lambda U$ defines the minimum number of dimensions to represent the dataset. PCA can be applied to any dataset regardless of its size since its complexity is linear to the data size and cubic to the dimension. Unfortunately very few datasets can be processed with PCA effectively because the dimensions of real datasets are usually nonlinearly correlated. In Figure 47 we can see the PCA spectrum of the TIMIT database (without variance normalization). It shows that 10-components are sufficient to describe the data since almost 96% of the total energy is contained in them. The same results are also true for a smaller subset of TIMIT. This suggests that 10 is an upper bound for the intrinsic dimension of dataset. However the answer is different when the dataset is normalized. Mean-Variance normalization[1] is a very common technique in speech recognition that improves the error rate [59]. The PCA spectrum becomes more rich Figure 48 and all 39 components are important.

7.3 Experiments

The TIMIT dataset when transformed to MFCC on 25msec window and 50% overlap consists of 1.5 million (train and test set), 39-dimensional vectors. Our goal is to perform MFNU on the whole dataset.

7.3.1 Preliminary experiments

In our preliminary results we sampled randomly 20,000 and 100,000 points from the dataset. Application of MFNU showed that the whole dataset (without variance normalization) can be described with 3 dimensions only.[2] The whole experiment took 5 hours (for 100K points) where 95% of the time was spent in optimization and only 5% in computing the neighborhoods. Experiments on subsets of TIMIT without mean variance normalization showed that the embedding dimension never exceeded 3. Running MVU for more than 100K points didn't increase the embedding dimension.

[1] Every dimension of the MFCC vector is zero mean and unit variance over a sentence time period

[2] For a k-neighborhood equal to 7 and embedding dimension 3 the MFNU is feasible which means that the local distances can be preserved

For mean variance normalized data things change as most of the time is spent on the neighborhood computation rather than in optimization. For a 20,000 random subset of TIMIT (with variance normalization) 3 dimensions where not sufficient to represent the set. Repeated experiments with MFNU revealed that the set can be represented with 5 which is a significant improvement to the original 39-dimension. The results (without variance normalization) are depicted in Figure 49a. In Figure 49b the dataset is projected on the first 3 principal components from PCA.

7.3.2 300K points experiments

TIMIT features without mean variance normalization are not particularly interesting since their statistical properties didn't change from 20K to 100K points. For the mean variance normalized MFCCs things change. Computing all 10-nearest neighborhood took 9 hours while running the optimization took about 2 hours. Repeated experiments showed that the problem becomes feasible (all the distance constraints can be satisfied) after 15 dimensions[3]. The feasibility error becomes less than 1% so it makes sense to claim that the true dimension should be around that value. Principal Component Analysis on the 15-dimensional projection showed that all components have equal energy, indicating that the spectrum is flat. If we compare our results with PCA shown on Figure 7.4, we see that the first 15 components of PCA correspond to 53% of the energy of the total spectrum.

It is also interesting to see how the Probability Density Function (PDF) changes with the non-linear projection. Our experiments show that each of the 39 dimensions follows a gaussian distribution. There are only 2 dimensions that deviate from that rule. The projected dimensions follow the Gaussian distribution too. In Figure 51 the PDFs of the 39 dimensions and the projected 15 are shown for the whole dataset. In Figure 52 and 53 we show show the PDFs for the *uh* and *sh* phonemes. Again

[3]By setting the dimension less than 15 it was not possible to get feasibility error close to 0. That means the data cannot be embedded in less than 15 dimensions

some of the 39 dimensions are not Gaussians, while all of the 15 are. In general it is safe to conclude that the MVU preserved the original PDFs.

One critical question is how we chose the k-neighborhood for the points. Using a k-nearest neighbor classifier, we found out that for k=10 the points have 98.5% of the time a neighbor that belongs to the same class. A 10-nearest neighbor classification gives 70% accuracy. The fact that almost every point is connected to phonemes of the same class might lead to the conclusion that a k-nearest neighbor classifier is powerful method for doing phoneme classification. If we examine the nearest neighbors we will see that most of the time they come from the same sentence. This has to do with the way the data are generated, since the MFCCs are computed on a slowly sliding window, adjacent windows are strongly correlated. In fact we run the following experiment. We found the 10-nearest neighbors of a test set of 400K points in a training set containing points 1.1M points. This experiment took 1.5 days. In this experiment, the 10-nearest neighborhood of every test point contained a neighbor belonging in the same class 80% of the time. It is still high, but now the 10-nearest neighbor classification score is 46.7%.

7.3.3 1 Million points experiments

Our ultimate goal is to apply MVU/MFNU on the whole dataset 1.5M points. From our experience in dealing with large datasets it is critical to scale the experiments gradually. After 300K points the next milestone was the 1M points. Computing all nearest neighbors was the biggest challenge as it took 3.5 days while optimization terminated with error 2% after 1.5 days. Considering the scale of the problem (15 million variables and approximately 10 million constraints) the time is considered reasonable. All k-nearest neighborhood computation seems to dominate and this is due to the bad dimensionality properties of MFCCs. The goal of scaling up to 1M points was not to study further the statistical properties of TIMIT since

300K is a sufficient sample. It is considered as an intermediate step for full throttle MVU/MFNU on the whole dataset. Once this is done, then the unfolded features can be used for Speech recognition in a state-of-the-art speech recognizer.

7.3.4 Weaknesses, limitations of MVU/MFNU

Although MVU has shown some promising results on medium/large scale datasets as seen in chapter 3 there are some limitations as the number of data points increases.

The first problem is the choice of the k-neighborhood. If k is very small then the intrinsic dimension found will be small too and far from the reality. Up to now there is no theory for the choice of k in MVU. In [82] the authors gave an estimate of $k \sim N^{\frac{1}{d+1}}$ where d is the intrinsic dimension, that is unknown. But still the result is significant, since it states that as the collection of the data increases, k should increase too.

Another problem is that after building the k-neighborhood graph, some nodes tend to have high degree and some not. In other words some points tend to connect to everything. It is very common when points are images for example, that a blurry image tends to be nearest neighbor with almost every point. In general high concentration of neighbors spoils the manifold assumption. The graph can still be embedded in a low dimensional space but the maximization of variance doesn't work [78]. Dattorro's algorithm for rank reduction works but it is not scalable. Minimum Volume Embedding is a similar algorithm using a heuristic, but again it is not scalable. In [44] an algorithm is presented for evening the distribution of the links in the graph.

Finally, the major problem of wrong choice of k or poor graph construction is the preservation of ranks between neighbors. In other words MVU guarantees that it will preserve the distances of the k-nearest neighbors for every point. What it doesn't guarantee though is that these points will still remain the nearest neighbors. We tested the results of our experiments on speech and we show that although the

algorithm preserves the distances of the k-nearest neighbors (computed on the 39-dimensional points), but when we run nearest neighbors again on the 15-dimensional points we got different points as nearest neighbors Figure 7.3.4. In theory this could be solved by adding rank constraints. Instead of just preserving distances, also preserve ranks between distances. This is impractical though, since it would require $O(N^2)$ constraints. Instead a more practical solution would be to run all k-nearest neighbors every 10 or twenty iterations of the optimization process and add penalty constraints on points that appear to be nearest neighbors and they shouldn't be. The extra cost is relatively small since all k-nn is relatively cheap in lower dimensions.

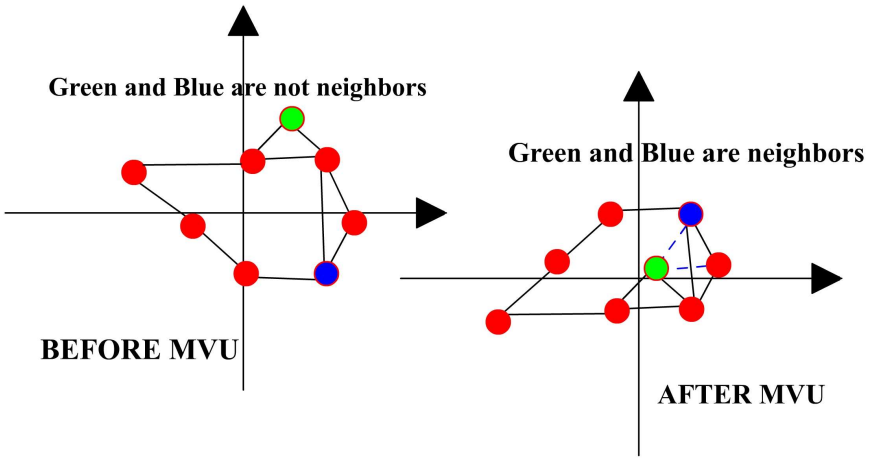


Figure 46:

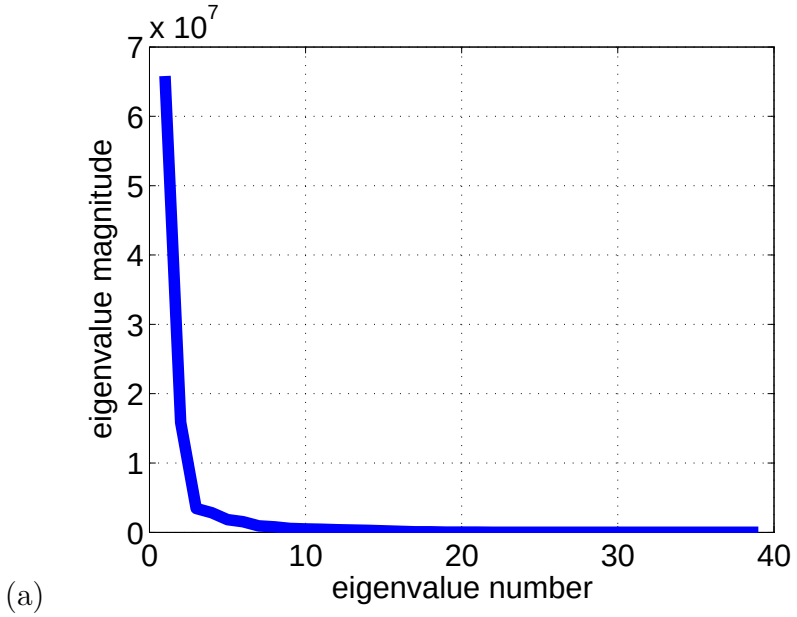
7.3.5 Future work

Running MFNU on such a large dataset is a big challenge. Up to now in our preliminary examples we used Fastlib [25], a C++ framework for implementing large scale Machine Learning Algorithms in C++. Our implementation is single threaded. Parallel implementation is one of our future plans.

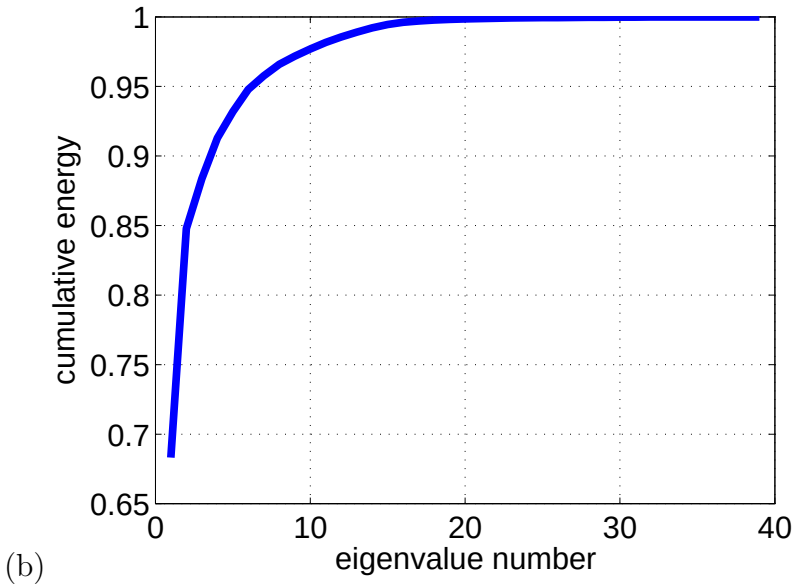
7.4 Summary

In this chapter a modern manifold learning method MVU/MFNU was for the first time applied on a large sample (1 million points[4]) of TIMIT, dealing with several computational challenges in all nearest neighbor computations and optimization. The results show significant improvement of the dimensionality from 39 to 15. The scale of the problem revealed weaknesses and limitations of MVU and gave insight for further improvement. This work stands as an intermediate step for applying non-linear projections for speech recognition

[4]This corresponds to a 15 million dimensional problem with approximately 10 million equality constraints

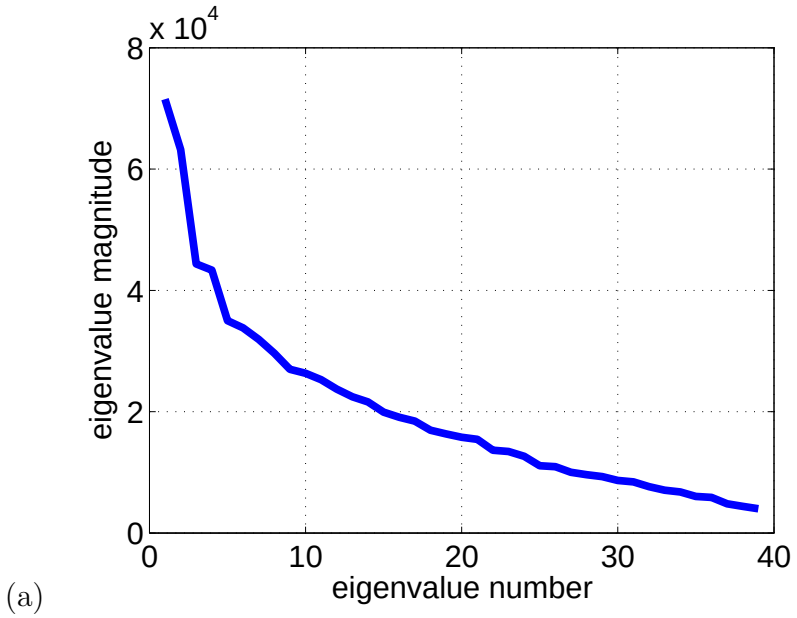


(a)

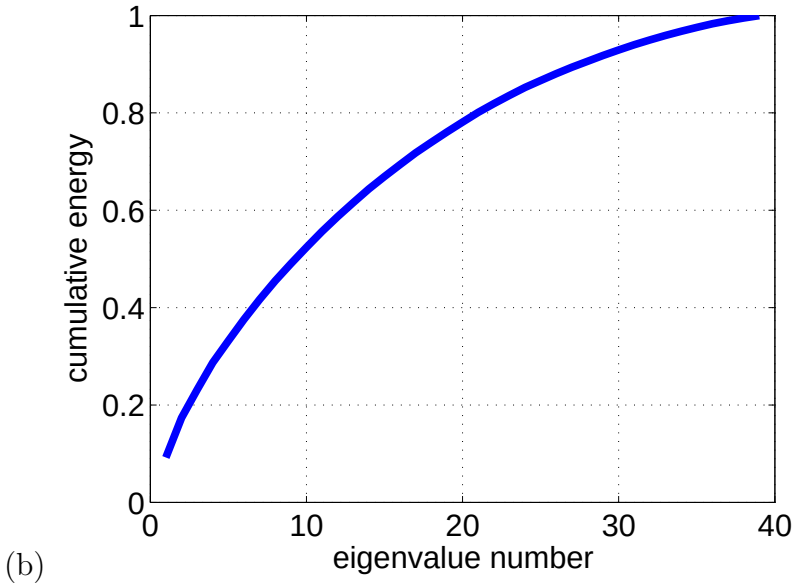


(b)

Figure 47: (a) The magnitude of the principal components for the whole TIMIT dataset (b) Cumulative percentage energy of the eigenvalues.

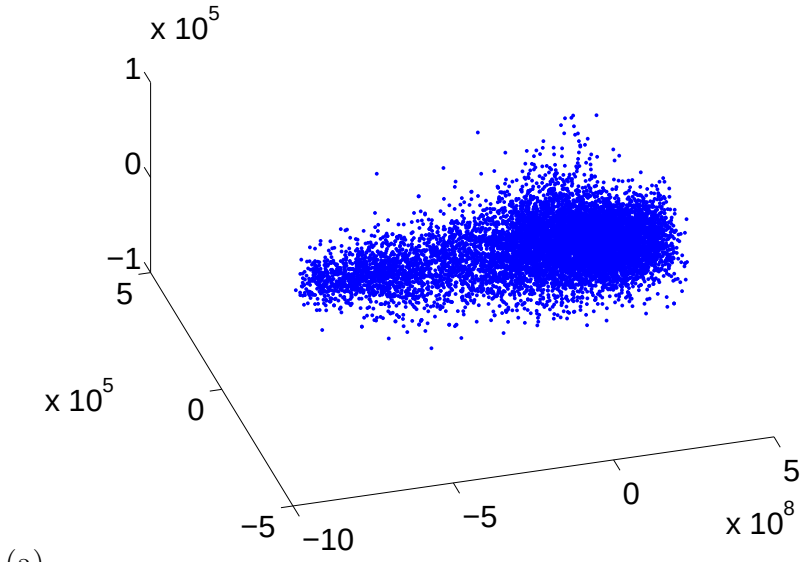


(a)

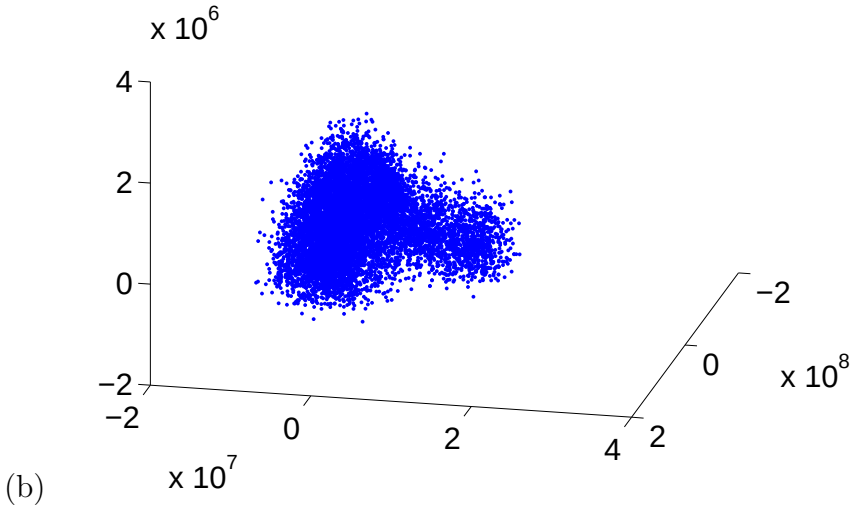


(b)

Figure 48: (a) The magnitude of the principal components for the 20,000 TIMIT dataset (with variance normalization) (b) Cumulative percentage energy of the eigenvalues.



(a)



(b)

Figure 49: (a) Embedding of the 100,000 TIMIT dataset (without variance normalization) with the MFNU method. (b) Embedding of the same dataset with PCA

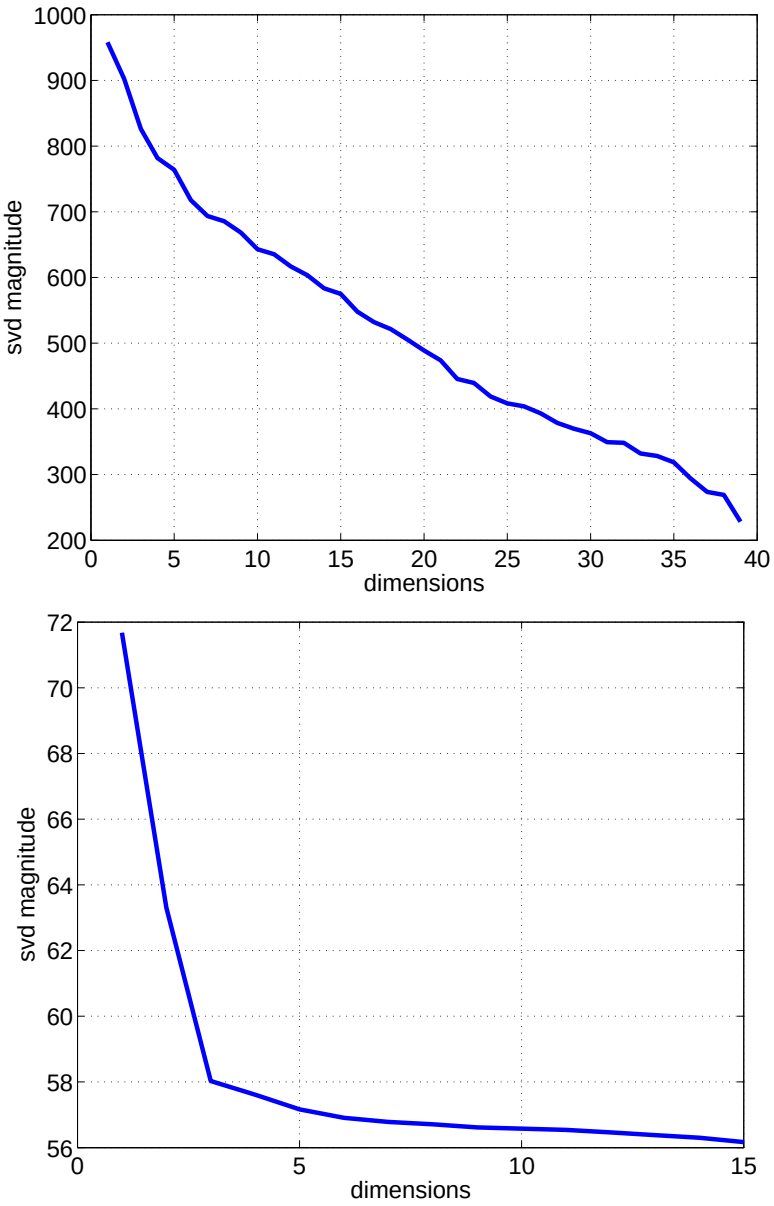


Figure 50: Left: PCA of the 39 dimensional TIMIT datapoints, Right: PCA spectrum of the 15 dimensional unfolded

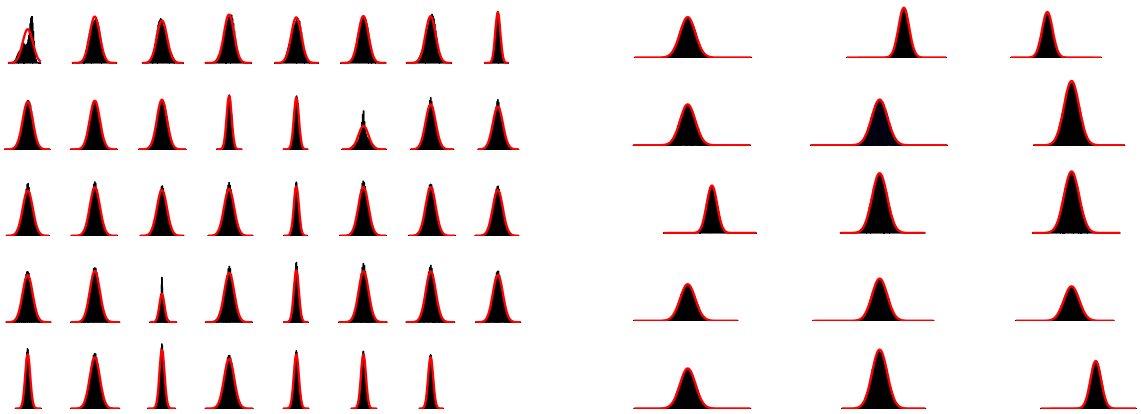


Figure 51: Left:PDFs of the 39 dimensional TIMIT data points, Right: PDFs of the of the 15 dimensional Unfolded

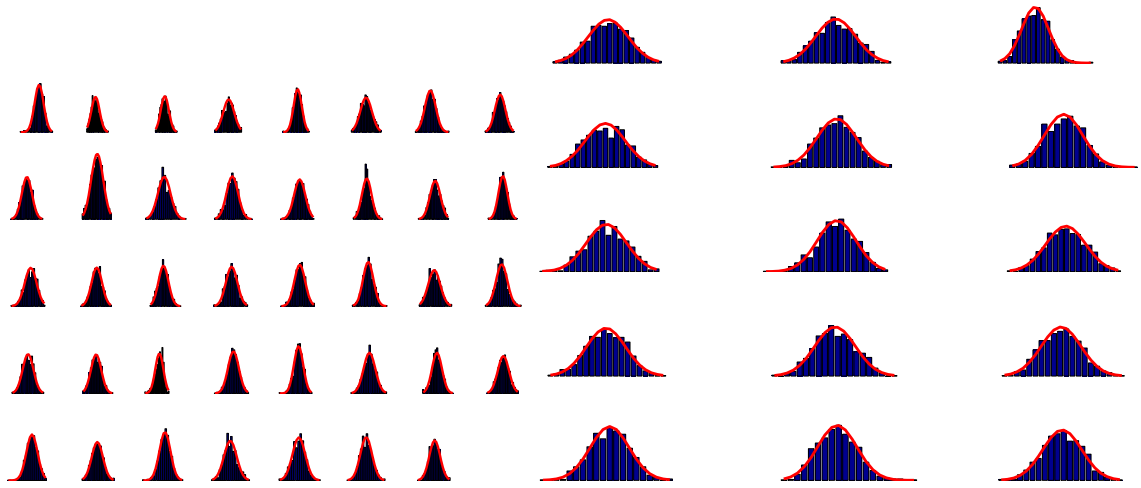


Figure 52: Left:PDFs of the 39 dimensional TIMIT data points for *uh*, Right: PDFs of the of the 15 dimensional Unfolded *uh*

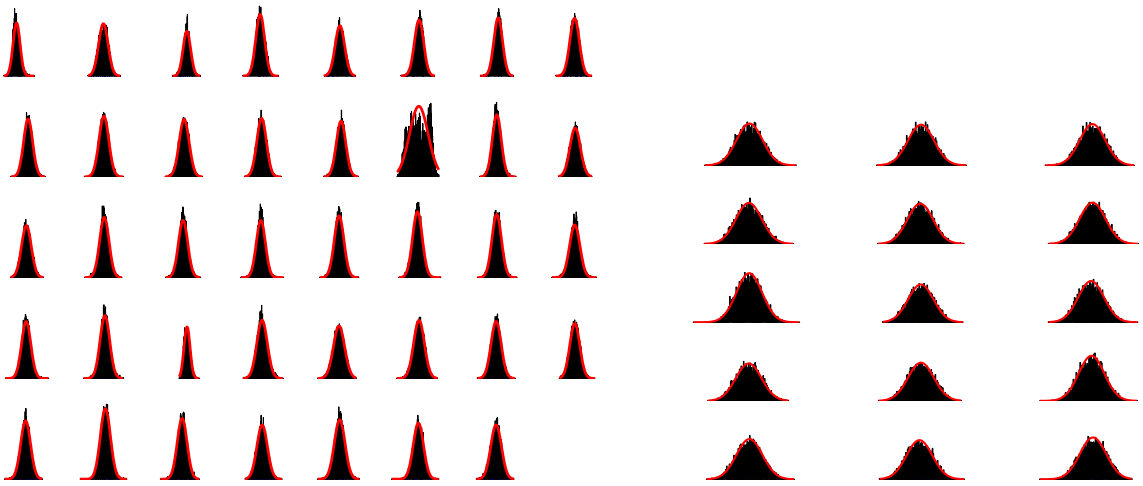


Figure 53: Right:PDFs of the 39 dimensional TIMIT data points for *sh*, Left: PDFs of the of the 15 dimensional Unfolded *sh*

CHAPTER VIII

CONCLUSION

In this dissertation we presented a series of algorithms for dimensionality reduction that can be cast as a sequence of convex optimization programs. Emphasis was given on the scalability of the algorithms. Semidefinite Programming remains the most important bottleneck, because of the $O(N^2)$ semidefinite matrix it requires. In order to scale MVU, the basic algorithm of this dissertation, it was necessary to sacrifice the convexity of the original formulation. Our experiments indicated that with the appropriate heuristics it is possible to achieve equivalent performance with the convex solver. Sacrificing non-convexity for scalability is one direction [18]. We conjecture that speedup of semidefinite programming can be achieved with approximations of the matrix with a ball tree. The results from [35] indicate that there is room for improving the complexity of representing a semidefinite matrix with the cosine tree.

Another important outcome of this thesis is that Nonnegative Matrix Factorization can be cast as a convex problem that guarantees global solution. We showed several convex relaxations along with a rank constrained formulation based on sequential convex programming. A global optimization technique was also presented that was not scalable. In the case of NMF it was necessary to sacrifice the convexity for scalability. A byproduct of our research on NMF was an algorithm for solving the problem of Completely Positive Factorization with sequential Convex programming.

The theoretical analysis of chapter 6 gives a unified framework for designing new more powerful dimensionality reduction algorithms that use more complex dot products than the the linear dot product. Despite its polynomial complexity that prevents scalability, appropriate approximations can make it scalable. We believe

that the theoretical results create more motivation for computational approximations on semidefinite programming as mentioned in the first paragraph.

An attempt was made to apply MVU on real speech datasets. The preliminary results didn't show any interesting property of the given intrinsic dimensions. Due to the data volume it was not practical to tune the parameters of MVU so as to get the best possible performance. The experiments gave us the opportunity to locate the problems and limitations of MVU and its variants.

8.1 Directions for future research

Although a lot of effort was put on scaling MVU on large datasets there is still a lot of work that has to be done to reach the demand of the big industries that generate terabytes of data. If I would start my PhD tomorrow I would spend time in parallel implementation of the algorithms presented here. Another promising direction is giving up determinism in the computations, by sampling the data. All the computations in this dissertation were exact (nearest neighbors, gradient, etc.). Recent work [35], has shown that approximations with Monte Carlo Sampling give tremendous speedups. There is no doubt that Semidefinite Programming (SDP) is a great tool for many machine learning applications. The main bottleneck of (SDP) is the representation of the positive semidefinite matrix that is of $O(N^2)$. We believe that trees can be used to approximate the matrix and compress it to $O(N)$.

CHAPTER IX

AUTHOR'S PUBLICATIONS

1. "Hyperkernel Based Density Estimation", Ravi S. Ganti, Nikolaos Vasiloglou and Alexander Gray, NIPS 2009 workshop on kernel learning

2. "Learning the Intrinsic Dimensions of the Timit Speech Database with Maximum Variance Unfolding", N. Vasiloglou, D V. Anderson, Alexander G. Gray. to appear in the 13th DSP workshop.

3. "Learning Isometric Separation Maps", N. Vasiloglou, A. Gray, D. Anderson. NIPS workshop on kernel learning, available at arxiv.org

4. "Non-Negative Matrix Factorization, Convexity and Isometry", N. Vasiloglou, A. Gray, D. Anderson, to appear in SIAM data mining 2009, available at arxiv.org

5. "Scalable Semidefnite Manifold Learning", N. Vasiloglou, A. Gray, D. Anderson, The 2008 IEEE Machine Learning in Signal Processing, Cancun, Mexico

6. "Parameter Estimation for Manifold Learning, Through Density Estimation" N. Vasiloglou, A. Gray, D. Anderson, The 2006 IEEE Machine Learning in Signal Processing, Maynooth Ireland

7. "Towards high quality region-of interest medical video over wireless networks using motion compensated temporal filtering" S. Rao, N. Vasiloglou, The 5th IEEE International Symposium on Signal Processing and Information Technology December 2005, Athens, Greece

8. "Isolated word, speaker dependent recognition under the presence of noise, based on an audio retrieval algorithm" N. Vasiloglou, R.W. Schafer, M.C. Hans, Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference.

9. "Design and optimization of 3D RF modules, microsystems and packages using electromagnetic, statistical and genetic tools [mm-wave interdigitated passband filter application]" N. Bushyager, D. Staiculescu, L. Martin, J.-H. Lee, N. Vasiloglou, M.M Tentzeris. Electronic Components and Technology, 2004. ECTC 04. Proceedings

10. "Fast hybrid electromagnetic/statistical approach for design and optimization of RF systems and packages" N. Bushyager, D. Staiculescu, L. Martin, J.-H. Lee, N. Vasiloglou, M.M Tentzeris. M.M. Advanced Packaging Materials: Processes, Properties and Interfaces, 2004.

11. "Investigation of the Effect of Fractal Shapes on the Broadband Behavior of One-Dimensional Optimized Antennas",N. Vasiloglou, D.Staiculescu and M.M.Tentzeris, Proc. of the 2004 URSI Symposium, p.69, Monterey, CA, June 2004.

12. "Lossless audio coding with MPEG-4 structured audio" N. Vasiloglou, R.W. Schafer, M.C. Hans, Web Delivering of Music, 2002 Proceedings.

REFERENCES

[1] Agarwal, S., Lanckriet, G., Wills, J., Imageworks, S., Cayton, L., and Belongie, S., "Generalized Non-metric Multidimensional Scaling," *To Appear, AISTATS*, 2007.

[2] Agrafiotis, D., "Stochastic proximity embedding," *Journal of Computational Chemistry*, vol. 24, no. 10, pp. 1215–1221, 2003.

[3] An, L. and Tao, P., "DC Programming Approach and Solution Algorithm to the Multidimensional Scaling Problem," *From Local to Global Optimization*, pp. 231–276, 2001.

[4] Anstreicher, K. and Burer, S., "Computable representations for convex hulls of low-dimensional quadratic forms," *Manuscript, University of Iowa, February*, 2007.

[5] Belkin, M. and Niyogi, P., "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation," 2003.

[6] Belkin, M., Niyogi, P., and Sindhwani, V., "On manifold regularization," in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, 2005.

[7] Bennett, K. and Demiriz, A., "Semi-Supervised Support Vector Machines," *NIPS*, 1999.

[8] Bentley, J., "Binary Search Trees Used for Associative Searching," *Communications*, 1975.

[9] Berman, A. and Shaked-Monderer, N., *Completely Positive Matrices*. World Scientific, 2003.

[10] Bishop, C. and service), S. O., *Pattern recognition and machine learning*. Springer, 2006.

[11] Boutsidis, C. and Gallopoulos, E., "SVD based initialization: A head start for nonnegative matrix factorization," *Pattern Recognition*, 2007.

[12] Boyd, S. and Vandenberghe, L., *Convex Optimization*. Cambridge University Press, 2004.

[13] Brookes, M., "VOICEBOX: Speech Processing Toolbox for MATLAB," *World Wide Web, http://www. ee. ic. ac. uk/hp/staff/dmb/voicebox/voicebox. html*, 2000.

[14] Burer, S., "On the copositive representation of binary and continuous nonconvex quadratic programs," *Mathematical Programming, Series A*, vol. 115, 2008.

[15] Burer, S. and Letchford, A., "On Non-Convex Quadratic Programming with Box Constraints,"

[16] Burer, S. and Monteiro, R., "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization," *Mathematical Programming*, vol. 95, no. 2, pp. 329–357, 2003.

[17] Coifman, R. and Lafon, S., "Diffusion maps," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 5–30, 2006.

[18] Collobert, R., Sinz, F., Weston, J., and Bottou, L., "1 Trading Convexity for Scalability,"

[19] Dattorro, J., *Convex Optimization & Euclidean Distance Geometry*. Lulu. Com, 2006.

[20] Donoho, D. and Grimes, C., "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[21] Dür, "Interior Points of the Completely Positive Cone,"

[22] Fazel, M., Hindi, H., and Boyd, S., "A rank minimization heuristic with application to minimum order system approximation," *American Control Conference, 2001. Proceedings of the 2001*, vol. 6, 2001.

[23] Floudas, C., *Deterministic Global Optimization: Theory, Methods and Applications*. Kluwer Academic Pub, 2000.

[24] Friedman, J., Bentley, J., and Finkel, R., "An Algorithm for Finding Best Matches in Logarithmic Expected Time," *ACM Transactions on Mathematical Software*, vol. 3, no. 3, pp. 209–226, 1977.

[25] Garry, B., Ryan, R., Nikolaos, V., and Alexander, G., "FASTlib Design and Development Manual, Version 0.1," tech. rep., Georgia Institute of Technology, 2008.

[26] Gonzalez Teofilo, F., "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293 – 306, 1985.

[27] Gray, A. and Moore, A., "N-Body problems in statistical learning," *Advances in Neural Information Processing Systems*, vol. 13, 2001.

[28] Györfi, L., *A Distribution-Free Theory of Nonparametric Regression*. Springer, 2002.

[29] Hair Jr, J., Anderson, R., Tatham, R., and Black, W., *Multivariate data analysis: with readings*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1995.

[30] Ham, J., Lee, D., Mika, S., and Schölkopf, B., "A kernel view of the dimensionality reduction of manifolds," in *Proceedings of the twenty-first international conference on Machine learning*, ACM New York, NY, USA, 2004.

[31] Hastie, T., Tibshirani, R., and Friedman, J., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[32] Heiler, M. and Schnorr, C., "Learning Non-Negative Sparse Image Codes by Convex Programming," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, 2005.

[33] Heiler, M. and Schnorr, C., "Reverse-Convex Programming for Sparse Image Codes," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 3757, p. 600, 2005.

[34] Heiler, M. and Schnorr, C., "Controlling Sparseness in Non-negative Tensor Factorization," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 3951, p. 56, 2006.

[35] Holmes, M., Gray, A., and Charles, I., "QUIC-SVD: Fast SVD Using Cosine Trees," *NIPS*, 2008.

[36] Horst, R. and Tuy, H., *Global Optimization: Deterministic Approaches*. Springer, 1996.

[37] Hoyer, P., "Non-negative Matrix Factorization with Sparseness Constraints," *The Journal of Machine Learning Research*, vol. 5, pp. 1457–1469, 2004.

[38] http://archive.ics.uci.edu/ml/datasets.html.

[39] http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html.

[40] http://isomap.stanford.edu/face_data.mat.Z.

[41] http://svmlight.joachims.org/.

[42] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

[43] http://www.cs.helsinki.fi/u/phoyer/software.html.

[44] Huang, B. and Jebara, T., "Maximum likelihood graph structure estimation with degree distributions," in *Analyzing Graphs: Theory and Applications, NIPS Workshop*, 2008.

[45] Jansen, A. and Niyogi, P., "Intrinsic Fourier analysis on the manifold of speech sounds," *ICASSP 2006 Proceedings.*

[46] Kaykobad, M., "On nonnegative factorization of matrices," *Linear Algebra and its Applications*, vol. 96, pp. 27–33, 1987.

[47] Kim, H. and Park, H., "Non-Negative Matrix Factorization Based on Alternating Non-Negativity Constrained Least Squares and Active Set Method,"

[48] Kim, S., Magnani, A., Koh, A., and Boyd, S., "Learning the kernel via convex optimization," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pp. 1997–2000, 2008.

[49] Kruskal, J., "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.

[50] Kruskal, J. and Wish, M., *Multidimensional Scaling.* Sage Publications, 1978.

[51] Kulis, B., Surendran, A., and Platt, J., "Fast Low-Rank Semidefinite Programming for Embedding and Clustering," *Proc. 11th Intl. AISTATS Conference*, 2007.

[52] Kulis B., Sra S., J. e. S. D. I., "Scalable Semidefinite Programming using Convex Perturbation," *UTCS Technical Report, TR-07-47*, September, 2007.

[53] Lanckriet, G., Cristianini, N., Bartlett, P., El Ghaoui, L., and Jordan, M., "Learning the Kernel Matrix with Semidefinite Programming," *The Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[54] Langville, A., Meyer, C., and Albright, R., "Initializations for the nonnegative matrix factorization," *Proc. of the 12 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

[55] Lee, D. and Seung, H., "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[56] Lee, D. and Seung, H., "Algorithms for Non-negative Matrix Factorization," *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pp. 556–562, 2001.

[57] Lee, J. and Verleysen, M., *Nonlinear dimensionality reduction.* Springer.

[58] Lee, J., *Introduction to Smooth Manifolds.* Springer, 2003.

[59] Liu, D. and Nocedal, J., "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1, pp. 503–528, 1989.

[60] Liu, F., Stern, R., Huang, X., Acero, A., and SCIENCE, C.-M. U. P. P. S. O. C., *Efficient CEPSTRAL Normalization for Robust Speech Recognition*. Defense Technical Information Center, 1993.

[61] Marlin, B., *Collaborative Filtering: A Machine Learning Perspective*. PhD thesis, University of Toronto, 2004.

[62] Minc, H., *Nonnegative matrices*. Wiley New York, 1988.

[63] Moore, A., *The Anchors Hierarchy: Using the Triangle Inequality to Survive High Dimensional Data*. Carnegie Mellon University, the Robotics Institute, 2000.

[64] Nemirovski A, *Lectures on Modern Convex Optimization*.

[65] Nocedal, J. and Wright, S., *Numerical Optimization*. Springer, 1999.

[66] Ouyang, H. and Gray, A., "Learning dissimilarities by ranking: from SDP to QP," in *Proceedings of the 25th international conference on Machine learning*, pp. 728–735, ACM New York, NY, USA, 2008.

[67] Ozakin, A. and Gray, A., "Density preserving maps," *AI & Statistics (submitted)*, 2009.

[68] Paatero, P. and Tapper, U., "Positive Matrix Factorization: A Non-negative Factor Model with Optimal Utilization of Error Estimates of Data Values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[69] Pinkowski, B., "Principal component analysis of speech spectrogram images," *Pattern Recognition*, vol. 30, no. 5, pp. 777–787, 1997.

[70] Quatieri, T. and Quatieri, T., *Discrete-time speech signal processing: principles and practice*. Prentice Hall PTR, 2001.

[71] Rabiner, L. and Juang, B., *Fundamentals of speech recognition*. Prentice-Hall, 1993.

[72] Recht, B., Fazel, M., and Parrilo, P., "Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization," *Arxiv preprint arXiv:0706.4138*, 2007.

[73] Rosales, R. and Fung, G., "Learning sparse metrics via linear programming," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 367–373, ACM New York, NY, USA, 2006.

[74] Roweis, S. and Saul, L., "Nonlinear Dimensionality Reduction by Locally Linear Embedding," 2000.

[75] Schoelkopf, B., Smola, A., and Mueller, K., "Kernel Principal Component Analysis," *LECTURE NOTES IN COMPUTER SCIENCE*, pp. 583–588, 1997.

[76] Schölkopf, B. and Smola, A., *Learning with kernels*. The MIT Press, 2002.

[77] Seneff, S. and Zue, V., "Transcription and alignment of the TIMIT database," *Symposium on Advanced Man-Machine Interface through Spoken Language*, 1988.

[78] Shaw, B. and Jebara, T., "Minimum volume embedding," in *Artificial Intelligence and Statistics*, 2007.

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

[80] Silverman, B., *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, 1986.

[81] Singh-Miller, N., Collins, M., and Hazen, T., "Dimensionality Reduction for Speech Recognition Using Neighborhood Components Analysis," *Interspeech 2007*.

[82] Smith, A., Huo, X., and Zha, H., "Convergence and rate of convergence of a manifold-based dimension reduction algorithm," in *Advances in Neural Information Processing Systems 21* (Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., eds.), 2009.

[83] Song, L., Smola, A., Borgwardt, K., and Gretton, A., "Colored Maximum Variance Unfolding," NIPS, 2008.

[84] Srebro, N., "Learning with Matrix Factorizations," 2004.

[85] Srebro, N., Rennie, J., and Jaakkola, T., "Maximum-margin matrix factorization," *Advances in Neural Information Processing Systems*, vol. 17, pp. 1329–1336, 2005.

[86] Sturm, J., "Using SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11, no. 1, pp. 625–653, 1999.

[87] Talwalkar A., K. S. and H., R., "Large-Scale Manifold Learning," *IEEE International Conference on Vision and Pattern Recognition*, 2008.

[88] Tao, P. and An, L., "Difference of convex functions optimization algorithms (DCA) for globally minimizing nonconvex quadratic forms on Euclidean balls and spheres," *Operations Research Letters*, vol. 19, no. 5, pp. 207–216, 1996.

[89] Tenenbaum, J., Silva, V., and Langford, J., "A Global Geometric Framework for Nonlinear Dimensionality Reduction," 2000.

[90] Vandenberghe, L. and Boyd, S., "Semidefinite Programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996.

[91] Vasiloglou, N., Gray, A., and Anderson, D., "Parameter Estimation for Manifold Learning, Through Density Estimation," *Machine Learning for Signal Processing, 2006. Proceedings of the 2006 16th IEEE Signal Processing Society Workshop on*, pp. 211–216, 2006.

[92] Vasiloglou, N., Gray, A., and Anderson, D., "Scalable Semidefinite Manifold Learning," *MLSP*, 2008.

[93] Weinberger, K., Blitzer, J., and Saul, L., "Distance Metric Learning for Large Margin Nearest Neighbor Classification," *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, vol. 18, p. 1473, 2006.

[94] Weinberger, K., Packer, B., and Saul, L., "Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization," in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pp. 381–388, 2005.

[95] Weinberger, K. and Saul, L., "An Introduction to Nonlinear Dimensionality Reduction by Maximum Variance Unfolding," in *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, vol. 21, p. 1683, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[96] Weinberger, K. and Saul, L., "Unsupervised Learning of Image Manifolds by Semidefinite Programming," *International Journal of Computer Vision*, vol. 70, no. 1, pp. 77–90, 2006.

[97] Weinberger, K., Sha, F., and Saul, L., "Learning a kernel matrix for nonlinear dimensionality reduction," in *Proceedings of the twenty-first international conference on Machine learning*, ACM New York, NY, USA, 2004.

[98] Weinberger, K., Sha, F., Zhu, Q., and Saul, L., "Graph Laplacian Regularization for Large-Scale Semidefinite Programming," *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, vol. 19, p. 1489, 2007.

[99] Wolkowicz, H., Saigal, R., and Vandenberghe, L., *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*. Kluwer Academic Publishers, 2000.

[100] Wright, S., *Primal-Dual Interior-Point Methods*. Society for Industrial Mathematics, 1997.

[101] Zhang, Z. and Zha, H., "Principal Manifolds and Nonlinear Dimension Reduction via Local Tangent Space Alignment," *Arxiv preprint cs.LG/0212008*, 2002.

VITA

Nikolaos Vasiloglou also known as Nick the Greek in the scientific community was born in Lesbos Greece in 1976. He graduated from the National Technical University of Athens in 2000 with a bachelor's in Electrical and Computer Engineering. He received his MsE from Georgia Institute of Technology under the supervision of the DSP grandfather Ronald Schafer. He received his PhD from the same institute in 2009 under the special care of David Anderson and the supervision of Alexander the Gray(t).