

UNIVERSITY OF CALIFORNIA, MERCED

**Mean-shift Algorithms for
Manifold Denoising, Matrix Completion and Clustering**

A dissertation submitted in partial satisfaction of the
requirements for the degree

Doctor of Philosophy

in

Electrical Engineering & Computer Science

by

Weiran Wang

Committee in charge:

Professor Miguel Á. Carreira-Perpiñán, Chair
Professor Stefano Carpin
Professor Marcelo Kallmann

2013

Copyright

Weiran Wang, 2013

All rights reserved.

The dissertation of Weiran Wang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, Merced

2013

DEDICATION

To Lexin and Isaac...

TABLE OF CONTENTS

	Signature Page	iii
	Dedication	iv
	Table of Contents	v
	List of Figures	viii
	List of Tables	x
	Acknowledgements	xi
	Vita and Publications	xii
	Abstract of the Dissertation	xiii
Chapter 1	Introduction	1
	1.1 Manifold learning	1
	1.2 Mean-shift algorithms	5
	1.3 Contributions	6
Chapter 2	Manifold Blurring Mean-shift algorithms for manifold denoising	9
	2.1 Introduction	9
	2.2 Review of related work	11
	2.2.1 Machine Learning algorithms	11
	2.2.2 Computer Graphics algorithms	14
	2.2.3 Computational Geometry algorithms	19
	2.3 The Manifold Blurring Mean-shift (MBMS) Algorithm	19
	2.3.1 Practicalities	23
	2.4 Experimental results	24
	2.4.1 Noisy spiral with outliers	25
	2.4.2 More complex shapes	26
	2.4.3 Dimensionality reduction	26
	2.4.4 Classification of MNIST digits	29
	2.5 Discussion	32
	2.5.1 Different operators	32
	2.5.2 Local scaling	36
	2.5.3 Estimation of intrinsic dimensionality	39
	2.6 Conclusion	43
Chapter 3	A denoising view of matrix completion	47
	3.1 Introduction	47

	3.2	A brief review of related work	49
	3.3	Blurring mean-shift denoising algorithms for matrix completion	53
	3.3.1	GBMS/MBMS revisited	53
	3.3.2	GBMS/MBMS for matrix completion	55
	3.4	Experimental results	61
	3.5	Conclusion	68
Chapter 4		The K -modes algorithm for clustering	69
	4.1	Introduction	69
	4.2	A K -modes Objective Function	73
	4.3	Two K -modes Algorithms	74
	4.3.1	For Fixed σ	74
	4.3.2	Homotopy Algorithm	75
	4.3.3	User Parameters	76
	4.4	Relation with Other Algorithms	76
	4.5	Experimental results	79
	4.5.1	Toy Examples	79
	4.5.2	Degree Distribution of a Graph	81
	4.5.3	Handwritten Digit Images	83
	4.5.4	Summary	85
	4.6	Discussion	86
	4.7	Conclusion and Future Work	87
Chapter 5		The Laplacian K -modes algorithm for clustering	88
	5.1	Introduction	89
	5.2	Algorithm	90
	5.2.1	The Laplacian K -Modes Algorithm	90
	5.2.2	Optimization Procedure for Laplacian K -modes	91
	5.2.3	Out-of-sample Problem	95
	5.3	A brief review of related work	97
	5.4	Experimental results	100
	5.4.1	Illustrative Synthetic Examples	100
	5.4.2	Clustering Analysis	104
	5.5	Conclusion	108
Chapter 6		Concluding remarks	109
Appendix A		Projection onto the probability simplex	112
	A.1	Problem	112
	A.2	Algorithm	113
	A.3	A simple proof	114
	A.4	Matlab code	117

Appendix B	Convergence rates of the gradient projection algorithms	118
B.1	Basic algorithm	118
B.2	Weakly convex functions	119
B.3	Strongly convex functions	125
B.4	Acceleration scheme	130
B.4.1	Weakly convex function	130
B.4.2	Strongly convex functions	133

LIST OF FIGURES

Figure 1.1:	Sample images from the MNIST benchmark	2
Figure 1.2:	Motion-capture data of a running sequence	3
Figure 1.3:	Paths followed by the Gaussian mean shift algorithm	4
Figure 2.1:	Manifold blurring mean-shift algorithm (MBMS) and its particular cases	22
Figure 2.2:	Denoising a spiral with outliers over iterations	25
Figure 2.3:	Denoising a complex shape with nonuniform density and noise . . .	27
Figure 2.4:	Dimensionality reduction with Isomap and LTSA after MBMS denoising	28
Figure 2.5:	Classification performance of Nearest Neighbor on denoised MNIST	29
Figure 2.6:	Sample pairs of (original,denoised) images from MNIST	30
Figure 2.7:	Some misclassified images of MNIST	31
Figure 2.8:	Denoising a spiral with outliers using different operators and full graph	34
Figure 2.9:	Denoising a spiral with outliers using different operators and k -nn graph	35
Figure 2.10:	Estimation of intrinsic dimensionality of the 100D swissroll dataset.	41
Figure 2.11:	Estimation of intrinsic dimensionality of the swissroll dataset over different input dimensionalities.	42
Figure 2.12:	Denoising and dimensionality reduction with Isomap and LTSA after MBMS denoising using MLE estimated intrinsic dimensionality	44
Figure 2.13:	Denoising and dimensionality reduction with Isomap and LTSA after MBMS denoising using EIG estimated intrinsic dimensionality .	45
Figure 2.14:	Boundary effect of MBMS denoising using MLE and EIG estimated intrinsic dimensionality	46
Figure 3.1:	Denoising matrix completion algorithms	58
Figure 3.2:	Reconstruction errors over iterations on 100D swissroll	62
Figure 3.3:	Denoising effect of the different algorithms on 100D swissroll . . .	62
Figure 3.4:	Results on Mocap dataset: error curves	63
Figure 3.5:	Results on Mocap dataset: sample reconstructions	63
Figure 3.6:	Selected reconstructions of MNIST-7	65
Figure 3.7:	Performance of MBMS using random initializations.	66
Figure 3.8:	Reconstruction of MBMS using random initializations.	67
Figure 4.1:	A cluster of 7 rotated-1 USPS digit images and the centroids found by K -means, K -modes, and mean-shift	71
Figure 4.2:	K -modes results on 3-Gaussian clusters	79
Figure 4.3:	K -modes results on 2-moons	80
Figure 4.4:	K -modes results on degree distribution of a graph	82

Figure 4.5:	Clustering results on USPS data with K -modes	84
Figure 5.1:	Learning curve of Laplacian K -modes on a synthetic problem.	94
Figure 5.2:	Synthetic dataset of 5-spirals	100
Figure 5.3:	Synthetic dataset of noisy 2-moons	101
Figure 5.4:	Occluder segmentation result	103
Figure 5.5:	Clustering results on USPS data with Laplacian K -modes	104
Figure 5.6:	Centroids found by different algorithms on MNIST	106
Figure 5.7:	Centroids found by different algorithms on COIL-20	107

LIST OF TABLES

Table 2.1:	Summary of different operators	33
Table 3.1:	Reconstruction errors obtained by different algorithms on 100D swis- sroll	61
Table 3.2:	Reconstruction errors obtained by different algorithms on MNIST-7 .	64
Table 5.1:	Comparison of properties of several well-known clustering algorithms	89
Table 5.2:	Statistics of three real world datasets for clustering	105
Table 5.3:	Clustering accuracy on three datasets	105
Table 5.4:	Normalized Mutual Information on three datasets	106

ACKNOWLEDGEMENTS

First of all, I am grateful to my advisor Miguel, for his insightful guidance throughout the past few years, for his meticulousness in helping me with problems arising from my research, and for his tough and honest advice for personality and life. I also would like to thank my thesis committee members, Stefano and Marcelo, who have provided meaningful comments and challenging questions from different perspectives.

I am extremely fortunate to study from Mayya Tokmann, Harish Bhat, Arnold Kim, and Boaz Ilan, with whom I took or audited almost all PDE courses offered by the Applied Mathematics Department of UC Merced. They are very generous to allow me to sit in their classes and answer my questions, even though I am not officially their student. What they have offered me is not just knowledge of PDE and beauty of math, but more importantly healthy distractions when I was extremely depressed by vitreous detachment. I am especially indebted to Mayya, for her kind and sincere suggestions about health and life.

Over the past 5 years, the Chinese students of the EECS department often had lunch together, as well as poker nights and basketball games. Though such activities, we had fun and shared a lot of information about how to settle down and get used to the life in the US. I especially appreciate the the help I received from Chao Qin, Tao Liu, Lun Jiang, and Jianwu Zeng, when I was still new to Merced.

I thank all the undergraduate students I have taught during my teaching assistantships. They are so young and brave, and they always remind me of my own happy college life in China and that I should hold on to my dreams.

I am most grateful to my wife Lexin, who has always been a great cooker, listener, and healer to me. Without her, I can not imagine how I would go though this long journey. We just had our son Isaac who, up to now, seems to be a miracle and gift of life. It is with their company that I now have a home and shall no longer feel lonely. And thanks to my parents, whom I can not possibly thank in words.

VITA

- 2005 Bachelor of Engineering in Computer Science and Engineering, Huazhong University of Science and Technology, Wuhan, China
- 2008 Master of Science in Computer Science, Chengdu Institute of Computer Applications, Chinese Academy of Sciences, Chengdu, China
- 2013 Ph. D. in Electrical Engineering and Computer Science, University of California, Merced

PUBLICATIONS

Miguel Á. Carreira-Perpiñán and Weiran Wang. “The K-modes Algorithm for Clustering”. *arXiv:1304.6478 [cs.LG]*, Apr. 23, 2013.

Miguel Á. Carreira-Perpiñán and Weiran Wang. “Distributed Optimization of Deeply Nested Systems”. *arXiv:1212.5921 [cs.LG]*, Dec. 24, 2012.

Weiran Wang and Miguel Á. Carreira-Perpiñán. “Nonlinear Low-Dimensional Regression using Auxiliary Coordinates”. *The Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2012)*, pages 1295-1304, 2012.

Weiran Wang and Miguel Á. Carreira-Perpiñán and Zhengdong Lu. “A Denoising View of Matrix Completion”. *Advances in Neural Information Processing Systems 24 (NIPS 2011)*, pages 334–342, 2011.

Weiran Wang and Miguel Á. Carreira-Perpiñán. “Manifold Blurring Mean Shift Algorithms for Manifold Denoising”. *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, pages 1759–1766, 2010.

ABSTRACT OF THE DISSERTATION

**Mean-shift Algorithms for
Manifold Denoising, Matrix Completion and Clustering**

by

Weiran Wang

Doctor of Philosophy in Electrical Engineering & Computer Science

University of California, Merced, 2013

Professor Miguel Á. Carreira-Perpiñán, Chair

Modern high dimensional data poses serious difficulties for various learning tasks. However, most high dimensional problems exhibit manifold structure, i.e., there exist only a few degrees of freedom that matter for the task at hand. Exploring such intrinsic structure is the key to designing accurate and efficient learning algorithms. In this thesis, we demonstrate the use of mean-shift, a popular mode-finding and clustering algorithm, for learning problems involving manifold structure. In particular, we propose several new

algorithms based on the mean-shift update for the tasks of manifold denoising, matrix completion, and centroid-based clustering.

The first algorithm, manifold blurring mean-shift (MBMS), is an algorithm of the predictor-corrector type. It alternates a predictor, blurring mean-shift step that acts as an isotropic low-pass filter and a corrector, projection step that removes the shrinkage of data along the manifold, where the manifold structure is estimated by local PCA. The algorithm achieves anisotropic denoising, can be used as a pre-processing step for dimension reduction and classification, and significantly improves over previous manifold denoising algorithms. Furthermore, we apply MBMS to matrix completion/missing value problems by iteratively denoising the whole dataset and filling in the observed entries. In contrast to the popular low-rank approaches based on a globally linear assumption, our algorithm preserves locally linear structure instead when the data is globally nonlinear. This simple approach provides a fresh view of the matrix completion problem, and greatly improves over several previous approaches.

We also propose two new, mode-based algorithms for clustering. The first one, which we call the K-modes algorithm, partitions a dataset into a pre-specified number of clusters, and provides a representative centroid of each cluster. Each centroid is the mode of the kernel density estimate defined by each cluster and is thus located in a high-density area. The algorithm is computationally inexpensive and more robust than K-means and mean-shift. We then provide a continuous relaxation for the hard partition rule of K-modes and impose a Laplacian smoothing penalty so that similar input samples receive similar assignments. The new algorithm, which we call the Laplacian K-modes algorithm, is able to handle non-convex, complex-shaped clusters, has an efficient optimization procedure, and shares nice properties with many well-known clustering algorithms.

The proposed mean-shift algorithms are simple and very easy to implement, yet they have superior performance on the tasks of consideration compared to previous approaches. We demonstrate them on various high dimensional datasets from different domains.

Chapter 1

Introduction

1.1 Manifold learning

In the field of machine learning, one often come across datasets with manifold structure. In this thesis, we refer to “manifold structure” the following intuitive properties of the dataset:

- Although the dataset may live in very high dimensional input space, there is intrinsically only a few degrees of freedom that control the generation of data points or matter for the task at hand. These degrees of freedom expand a lower dimensional *latent space*, whose dimensionality is called the *intrinsic dimensionality* of the dataset.
- Small changes in the latent space corresponds to small changes of data in the input space.
- In the input space, there exists a so-called *tangent space* at each data point, which is a linear subspace and has the same dimensionality as the intrinsic dimensionality, and a local neighborhood of the point lies approximately on this space. As a result, each point can be approximately reconstructed linearly by its local neighborhood.



Figure 1.1: Sample images from the MNIST benchmark.

In real-world applications, datasets with manifold structure abound. We give two examples below.

Figure 1.1 shows sample images of the well-known MNIST dataset (LeCun et al., 1998), which is a widely used benchmark for handwritten digit recognition. The dataset contains 28x28 grayscale images of digits 0-9 from multiple users. It is obvious that small changes in translation, rotation, scaling, and different writing styles (e.g., self-intersection of digit 2 and short bar for digit 7) will change the appearance of image (pixel values) slightly, and should not change the identity of the image. On the other hand, a good digit recognition system shall take into account this structure and be invariant to these variations.

Another good example of manifold structure is the sensor readings from the motion-capture problem: there might be multiple sensors attached to the joints of the human subject, each recording the position of a joint, but the sensor readings are highly redundant as the degrees of variations in a given motion sequence is typically quite lower. Thus the motion sequence approximately lies in a nonlinear, low dimensional latent space. Figure 1.2 shows the motion-capture data recorded from several cycles of a running sequence (217 points, each corresponds to 34 3D markers) and its 2D latent repre-

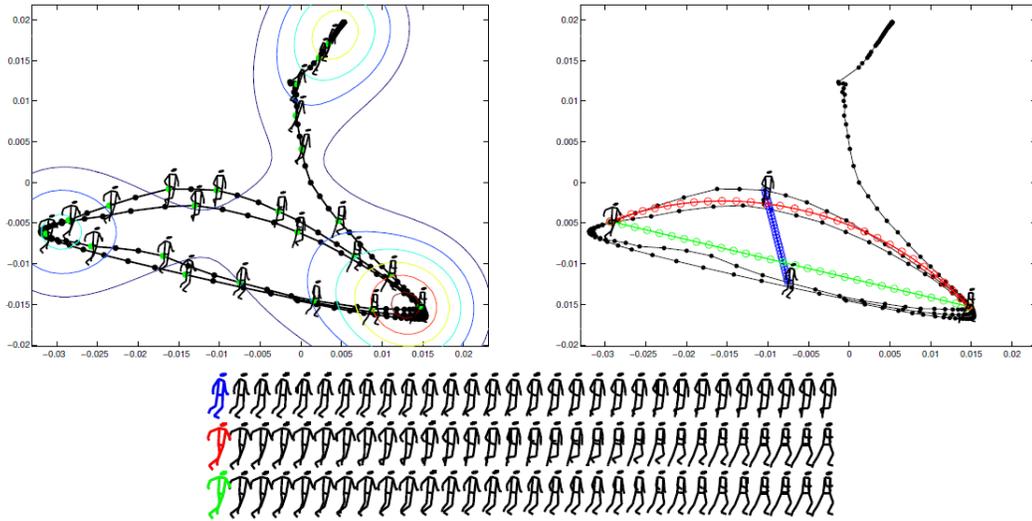


Figure 1.2: Motion-capture data of a running sequence and its latent representation. Left: the latent space, data points are connected in the sequential order of their corresponding observations, some of which are plotted as a stick man; the loop is travelled clockwise. Right: 3 trajectories in latent space (containing 30 equispaced samples) and reconstruction of the corresponding trajectory in observed space (lower plot). This figure is taken from Carreira-Perpiñán and Lu (2007).

sensation obtained by Carreira-Perpiñán and Lu (2007). The latent representation of this sequence is a loop, which characterizes the motions sequence very well, and sampling in the latent space produces realistic motions. The redundancy in the high dimensional observed space also gives a good chance for reconstructing the readings even when a large proportion of them are missing.

As we can see, exploring the manifold structure help us better understand the generation of data, and should be incorporated for particular learning tasks at hand. Manifold learning, also called dimension reduction, has been a very active research area (Belkin and Niyogi, 2003; Carreira-Perpiñán and Lu, 2007, 2008; Coifman et al., 2005; Donoho and Grimes, 2003; Hinton and Salakhutdinov, 2006; Roweis and Saul, 2000; Tenenbaum et al., 2000), and the abovementioned properties are the most important intuitions in this field.

From a computational point of view, learning a full-fledged parametric model for high dimensional data often leads to severe over-fitting (*curse of dimensionality*), thus di-

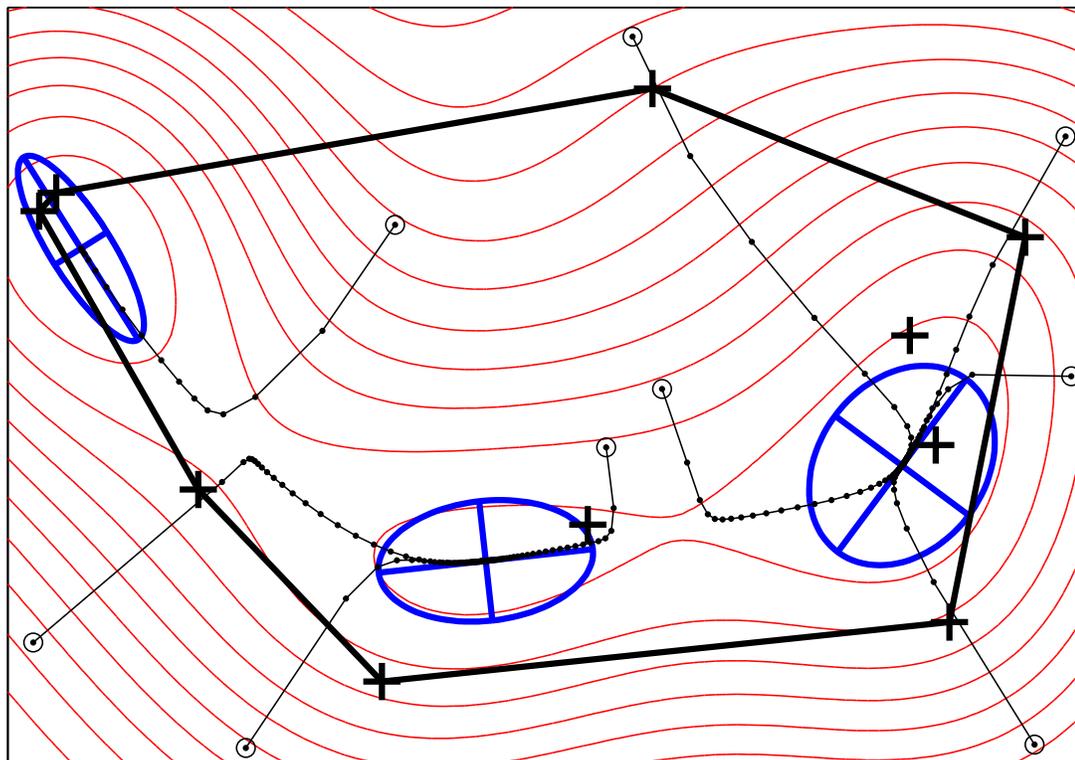


Figure 1.3: Paths followed by the Gaussian mean shift algorithm for various starting points, overlaid on a contour plot of the Gaussian kernel density estimate $p(x)$ (in the homoscedastic, isotropic case but with non-uniform π_i). The data points x_i are marked “+”. A mode is located at the centre of each ellipse; the ellipse indicates the eigenvectors (rescaled to improve visibility) of the Jacobian $J(x)$ at that mode. The thick-line polygon is the convex hull of the data points. This figure is taken from Carreira-Perpiñán (2007).

dimension reduction techniques are also useful for model regularization. This accounts for the success of the currently very popular lasso (Tibshirani, 1996), group-lasso (Yuan and Lin, 2006), low-rank (Candès and Tao, 2010) regularization techniques, which can also be considered as dimension reduction by introducing sparsity. Dimension reduction may also significantly reduce the training and test time of a model.

1.2 Mean-shift algorithms

The mean-shift algorithm originates in an idea of Fukunaga and Hostetler (1975) and has been developed by Cheng (1995), Carreira-Perpiñán (2000), Comaniciu and Meer (2002) and others. Given the N data points $\{\mathbf{x}_1, \dots, \mathbf{x}_N\} \subset \mathbf{R}^D$, consider a kernel density estimate (kde) with kernel $K(t)$ for $t \geq 0$:

$$p(\mathbf{x}) = \sum_{i=1}^N \pi_i \frac{1}{Z_i} K(d(\mathbf{x}, \mathbf{x}_i; \Sigma_i)), \quad (1.1)$$

where $\pi_i \in (0, 1)$ is the mixing proportion of point i (satisfying $\sum_{i=1}^N \pi_i = 1$), Σ_i is its covariance matrix (positive definite), Z_i is a normalization constant that only depends on Σ_i (e.g., $Z_i = |2\pi\Sigma_i|^{1/2}$ for the Gaussian Kernel), and $d(\mathbf{x}, \mathbf{x}_i; \Sigma_i) = (\mathbf{x} - \mathbf{x}_i)^T \Sigma_i^{-1} (\mathbf{x} - \mathbf{x}_i)$ is the Mahalanobis distance. Most widely used are the Gaussian kernel $G(t) = e^{-t/2}$ and the Epanechnikov kernel

$$K(t) = \begin{cases} 1 - t, & \text{if } t \in [0, 1), \\ 0, & \text{otherwise.} \end{cases}$$

We can find modes (local maxima) of $p(\mathbf{x})$ by seek stationary points $\frac{\partial p(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{0}$, and solving for \mathbf{x} suggests a fixed point iteration scheme called mean-shift update. We mainly focus on the simple setting where the data points have constant weights ($\pi_i = \frac{1}{N}$) and the kde uses Gaussian kernel with isotropic covariance ($\Sigma_i = \sigma^2 \mathbf{I}$). This setting brings into bear an elegant mean-shift update rule

$$p(n|\mathbf{x}) = \frac{\exp\left(-\frac{1}{2} \|\mathbf{x} - \mathbf{x}_n\|^2 / \sigma^2\right)}{\sum_{n'=1}^N \exp\left(-\frac{1}{2} \|\mathbf{x} - \mathbf{x}_{n'}\|^2 / \sigma^2\right)}, \quad \mathbf{x} \leftarrow \mathbf{f}(\mathbf{x}) = \sum_{n=1}^N p(n|\mathbf{x}) \mathbf{x}_n, \quad (1.2)$$

mapping any point $\mathbf{x} \in \mathbf{R}^D$ to a weighted mean of points in the dataset. The difference $\mathbf{f}(\mathbf{x}) - \mathbf{x}$ is called the *mean-shift* vector and hence the name of the update. Since the weights $p(n|\mathbf{x})$ are non-negative, each update lies in the convex hull of the dataset, see Figure 1.3 for an illustration of mean-shift path. We refer to this setting as *Gaussian Mean-shift (GMS)*.

The mean-shift algorithm can then be applied to clustering by running mean-shift updates from each data point, declaring each mode as representative of one cluster, and assign data point to the mode it converges to. The algorithm is nonparametric and the clustering is deterministic given the bandwidth σ . Mean-shift has proven particularly successful in computer vision applications such as image segmentation (Comaniciu and Meer, 2002) and tracking (Comaniciu et al., 2003). However, it is also well-known that mean-shift suffers greatly from high computational complexity ($\mathcal{O}(N^2D)$ per iteration) and slow convergence speed. The Gaussian Mean-shift (GMS) algorithm is equivalent to the EM algorithm and has in general linear convergence rate (Carreira-Perpiñán, 2007). In fact, accelerating mean-shift has been a topic of active research (Carreira-Perpiñán, 2006a; Yuan et al., 2010).

The *Blurring Mean-shift* algorithm is a different version of the usual mean-shift algorithm. In blurring mean-shift, each point of the dataset actually moves to the weighted mean of the previous dataset after each iteration, and thus the whole dataset get updated. Focusing on the Epanechnikov kernel for computational efficiency, Fukunaga and Hostetler (1975) already observe the potential of blurring mean-shift for clustering and dimensionality reduction (denoising). Cheng (1995) later proves the convergence of blurring mean-shift, an unusual one where the whole dataset converge to a point. Carreira-Perpiñán (2006b) further proves the convergence rate to be cubic for Gaussian kernel (much faster than the GMS algorithm) and suggests robust stopping criteria to obtain a partition of dataset in the *Gaussian Blurring Mean-shift (GBMS)* clustering algorithm.

1.3 Contributions

In this thesis, we apply the mean-shift algorithm to a range of learning tasks involving manifold structure. The contributions and organization of this thesis are summarized as below.

- In Chapter 2, we apply the mean-shift algorithms to the problem of manifold de-

noising. We propose a predictor-corrector type of algorithm that alternates two steps: a first (blurring) mean-shift step that acts as a low-pass filter and shrink the noise in all directions in input space, and a second corrector step that removes the shrinkage of data along manifold, with manifold structure estimated by local PCAs. As a result, the noise orthogonal to the manifold has been removed while signal along the manifold has been kept. We name this new algorithm *Manifold Blurring Mean-shift (MBMS)*. This simple algorithm has close relationship to the research of anisotropic denoising in computer graphics and image denoising communities, and significantly improves previous manifold denoising algorithms in the field of machine learning.

- In Chapter 3, we apply the MBMS algorithm to the matrix completion/missing value problem. One basic idea of missing value reconstruction is to assume that the data matrix has low-rank. While there exists nice optimization problems and certain theoretical guarantees to this approach, the *globally* low-rank assumption is somewhat restrictive. When the data is globally nonlinear, we shall instead make use of its *locally* linear property. So we take the reconstruction of a low-rank matrix completion algorithm as starting point, consider this reconstruction to be noisy at missing entries, and apply our manifold denoising algorithm, with values at known entries fixed. This simple approach provides a fresh view of the matrix completion problem, and greatly improves the starting point.
- In Chapter 4, we propose a new *K-modes* algorithm for clustering. As a popular clustering algorithm, mean-shift model the data with non-parametric kernel density estimate (kde) and iteratively moves each data points towards the modes (local maximum of kde). But it is difficult to find a pre-specified number of clusters because the number of modes are implicitly determined by the kernel width parameter. We combine density and hard clustering assignment and proposed a new objective function called *K-modes*, which can partition data into a pre-specified number of cluster, while the centroid of each cluster is the mode of kde defined by each cluster and can be found by mean-shift updates. The *K* modes returned by the algorithm live in high density area of the data space and are prototypical

representatives of the dataset, and thus provides better understanding of the data and clustering result.

- In Chapter 5, we improve the K -modes algorithm for datasets with manifold structure. The above K -modes algorithm has the drawback that it can only find convex clusters. We combine K -modes with the Laplacian smoothing techniques (widely used in spectral clustering, manifold regularization, semi-supervised learning), and provide a new model that can handle non-convex, complicated shaped clusters. Part of the model requires solving a convex quadratic program with simplex constraints. We applied to this problem the gradient projection algorithm and its Nesterov's acceleration techniques, which leads to a very simple procedure that facilitates efficient projections onto the probability simplex, has nice convergence rate, and scales very well. We name this new algorithm *Laplacian K -modes*. It shares nice properties with many well-known clustering algorithms.
- In Chapter 6, we give concluding remarks and discuss future research for manifold learning with mean-shift.
- In Appendix A and Appendix B, we give self-contained proofs for an efficient algorithm for computing the projection onto the probability simplex and the convergence rate of gradient projection algorithms, respectively.

Chapter 2

Manifold Blurring Mean-shift algorithms for manifold denoising

In this chapter, we propose a new family of algorithms for denoising data assumed to lie on a low-dimensional manifold. The algorithms are based on the blurring mean-shift update, which moves each data point towards its neighbors, but constrain the motion to be orthogonal to the manifold. The resulting algorithms are nonparametric, simple to implement and very effective at removing noise while preserving the curvature of the manifold and limiting shrinkage. They deal well with extreme outliers and with variations of density along the manifold. We apply them as preprocessing for dimensionality reduction; and for nearest-neighbor classification of MNIST digits, with consistent improvements up to 36% over the original data (Wang and Carreira-Perpiñán, 2010).

2.1 Introduction

Machine learning algorithms often take as starting point a high-dimensional dataset of N points $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N) \in \mathbb{R}^{D \times N}$, and then learn a model that is useful to infer information from this data, or from unseen data. Most algorithms, however, are more or less sensitive to the amount of noise and outliers in the data. For example, spectral di-

dimensionality reduction methods such as Isomap (Tenenbaum et al., 2000) first estimate a neighborhood graph on the dataset \mathbf{X} and then set up an eigenvalue problem to determine low-dimensional coordinates for each data point. Both steps are sensitive to noise and outliers, in particular building the neighborhood graph: it may be hard to find a good value (if it exists at all) for the number of neighbors k or the ball radius ϵ that will avoid disconnections or shortcuts. Other dimensionality reduction algorithms, such as latent variable models (e.g. mixtures of probabilistic PCAs (Tipping and Bishop, 1999)), try to learn a parametric model of the manifold and noise by maximum likelihood. However, these models are prone to bad local optima partly caused by noise and outliers. Although there are different ways of reducing the effects of noise and outliers, such as learning a graph in a more robust way (Carreira-Perpiñán and Zemel, 2005) or using robust error functions, in this paper we concern ourselves with a different approach: to denoise the dataset \mathbf{X} as a preprocessing step.

Data preprocessing is commonplace in machine learning. Consider, for example, the many simple but useful operations of subtracting the mean (possibly as a running average), low-pass filtering, standardizing the covariance, or removing outliers by trimming. Other operations are specific to certain types of data: deskewing or blurring for images, energy removal or cepstral normalization for speech. These operations help to achieve some invariance to unwanted transformations or to reduce noise and improve robustness. Here, we are interested in more sophisticated denoising techniques that adapt to the local manifold structure of high-dimensional data. We will assume that the dataset \mathbf{X} comes from a manifold of dimension $L < D$ to which noise has been added. We will not make any assumptions about the nature of this noise—the form of its distribution (e.g. whether long-tailed), or whether it varies along the manifold. Denoising a manifold is also useful by itself, for example 3D mesh smoothing in computer graphics (Taubin, 1995) or skeletonization of shapes such as digits. However, we will focus on denoising as a preprocessing step for supervised or unsupervised learning.

A good denoising algorithm should make as few assumptions about the data as possible, so nonparametric methods are preferable; and produce the same result for a given dataset, i.e., be deterministic. At the same time, it should have a small number of user

parameters to control the algorithm’s behavior (e.g. the amount of denoising). We propose an algorithm that fulfills these desiderata. It is based on two powerful ideas: the noise removal ability of locally averaging with a kernel of scale σ (implemented with the mean-shift algorithm); and the linear approximation of local manifold structure of dimension L (implemented with local PCA on the k nearest neighbors).

2.2 Review of related work

2.2.1 Machine Learning algorithms

In the machine learning field, one natural way of smoothing is to use dimensionality reduction techniques. Given the problem setting defined in Section 2.1, for some point $\mathbf{p} \in \mathbf{R}^D$, one could achieve a denoised version $\mathbf{q} \in \mathbf{R}^D$ in at least two possible manners:

1. First project \mathbf{p} to the latent space using a *dimension reduction/projection mapping* \mathbf{F} , and project its latent representation back to data space using a *reconstruction mapping* \mathbf{f} , i.e., $\mathbf{q} = \mathbf{f}(\mathbf{F}(\mathbf{p}))$.
2. Project \mathbf{p} onto the manifold by minimizing $\min_{\mathbf{y} \in \mathcal{Y}} \|\mathbf{p} - \mathbf{f}(\mathbf{y})\|$. By finding the optimizing parameter \mathbf{y}_0 , \mathbf{q} is set to be $\mathbf{f}(\mathbf{y}_0)$.

Some dimensionality reduction techniques provide both mappings, such as autoencoder (DeMers, 1993; Hinton and Salakhutdinov, 2006), latent variable models (Brand, 2003; Tipping and Bishop, 1999), and some unsupervised regression algorithm (Carreira-Perpiñán and Lu, 2008). While some other techniques provide only the reconstruction mapping, for instance most of the unsupervised regression algorithms (Lawrence, 2004; Smola et al., 2001). Most spectral methods (Belkin and Niyogi, 2002; Coifman et al., 2005; Roweis and Saul, 2000; Tenenbaum et al., 2000) provide only the latent representation of input data without giving any of the mappings. However, denoising can still be achieved in this case, as by Etyngier et al. (2007).

In spite of the fact that dimensionality reduction techniques can be used to achieve denoising, we argue that these methods (often sensitive to noise and/or local optima) may learn a better manifold if the training set is preprocessed to remove noise and outliers. Thus the aim of this research work is to propose a nonparametric denoising method that imposes minimal model assumptions and acts as preprocessing steps for other purposes (dimensionality reduction, supervised learning, etc).

Unnikrishnan and Hebert (2007) propose an algorithm to denoise point set that has manifold structure in 2-D or 3-D space, if the noise model of the sampling sensor is known. The algorithm models the neighborhood of each point with multiple linear subspaces which may intersect with each other. And as a result, the neighborhood is described as zero level set of some polynomial, the coefficients of which must satisfy certain constraints for it to degenerate as product of linear equations. A constrained weighted least squares minimization is then formed, and solved with initialization from the unconstrained version of the problem. The problem with this approach is that it can not deal with very high dimensional data, since the number of coefficients in the polynomial will be $\binom{n+d}{n}$ where n is the dimension of input sampled data and d is the degree of the polynomial, and the problem then becomes intractable for large n and d . Also, the authors note that choosing the correct support region of the loss function is crucial to the performance of the algorithm and it is still an open problem.

Hein and Maier (2007) extend the main idea of surface smoothing in the computer graphics literature to arbitrary dimensions. The paper establishes a diffusion process using graph Laplacian for the input point set, and solves it with the implicit Euler scheme. It also builds a noise model for the sampled manifold data, and derives large sample limit and mathematical analysis based on theory of graph Laplacian. Experiments show that the denoising algorithm could act as a preprocessing step for some semi-supervised learning problem to get better performance. The problem with this approach is that it produces obvious shrinking phenomenon and the dataset then becomes disconnected and concentrates in local clusters. This effect is inherent due to the usage of isotropic graph Laplacian.

Park et al. (2004) propose an algorithm for outlier handling and noise reduction of non-linear manifold data, mainly by projecting each data point onto an affine subspace computed from a local neighborhood by weighted PCA. A number of neighbors around each point are chosen to fit a local linear space. The weights of the neighbors and a weighted center of the neighborhood are computed iteratively to reduce the effect of outliers. A Minimum Spanning Tree can be further computed within the neighborhood to identify the outlier, according to the criteria that distance between an outlier and the nearest region of data is much larger than the average distance within that region. After projecting data points onto each local linear space, a bias between the projection and the true non-noisy data is estimated and corrected. This projection step of this algorithm is similar to one special case of the algorithm presented in Section 2.3. However, we argue that the heuristics for choosing weights and detecting outliers are complicated (parameters are hard to set in practice). In contrast, our algorithm performs well even with the existence of heavy outliers.

Gong et al. (2010) propose an algorithm for manifold denoising based on locally linear reconstructions. The algorithm first runs local PCA within the neighborhood of each data point and obtains a denoised version—the PCA reconstruction—for each point in that neighborhood. As a result, each point has multiple locally denoised versions (it lies in the K -nearest neighborhood for multiple points). Then the algorithm optimizes over a globally denoised version for the entire dataset by minimizing the sum of squared error between the global version and the local versions for each point, together with a smooth penalty term. This process may be repeated by replacing the original dataset with the globally denoised version. The authors show with experiments that their algorithm could achieve smaller reconstruction error for images with certain types of noise for image manifolds such as digits and faces, and the denoised images could improve classification rate. However, in the experiment, the algorithm needs to take as input a training set that contains both clean (noise free) images and noisy images to achieve good performance. The other disadvantage is that the algorithm needs to solve a $N \times N$ linear systems in each iteration to compute the globally denoised version, the dimension of which equals the number of points in the dataset.

2.2.2 Computer Graphics algorithms

There is a branch of denoising algorithms from the computer graphics field, which aims at smoothing mesh surfaces obtained by range sensing techniques. Taubin et al. (1996) consider the 3-D surface coordinates as signals, and thus smoothing of polyhedral surfaces is equivalent to designing low-pass filters on them. The authors define the *Laplacian* of a discrete graph signal by the formula (umbrella operator)

$$\mathcal{L}(\mathbf{x}_i) = \sum_{j \in \mathcal{N}_1(i)} w_{ij}(\mathbf{x}_j - \mathbf{x}_i), \quad (2.1)$$

where $\mathcal{N}_1(i)$ denotes the set of first order neighborhood of vertex i , i.e., the set of vertices that share edges (or faces) with vertex i , and weights w_{ij} are positive numbers adding up to 1 for each vertex. Let \mathbf{W} be the matrix of weights. Defining $\mathbf{K} = \mathbf{I} - \mathbf{W}$, the Laplacian operator has a simple matrix form $\mathcal{L}(\mathbf{x}) = -\mathbf{K}\mathbf{x}$, and \mathbf{K} has eigenvalues (frequencies) $0 \leq k_1 \leq \dots \leq k_n \leq 2$. Then $f(\mathbf{K})$, an analytic function of \mathbf{K} , is applied to the original signal to get a filtered version. Since $f(\mathbf{K})$ has the same eigenvectors as \mathbf{K} and has eigenvalues $f(k_1), \dots, f(k_n)$, it changes the frequency distribution of the signal and could be designed to approximate an ideal low-pass filter. Repeatedly using the linear operator $f(\mathbf{K})$ can therefore tailor the frequency content of the original mesh. The authors also point out the transfer function of Gaussian smoothing $f_N(k) = (1 - \lambda k)^N$ with $0 < \lambda < 1$ produces shrinkage. Hence, they suggest using $\lambda|\mu$ algorithm where $f_N(k) = ((1 - \lambda k)(1 - \mu k))^{N/2}$ with $0 < \lambda < -\mu$ to solve this problem. In this algorithm, the pass-band frequency is $k_{PB} = \frac{1}{\lambda} + \frac{1}{\mu}$, where λ and μ are set by the user.

Later on, Desbrun et al. (1999) further formalize the above approach in its *diffusion process* framework and propose a new class of denoising algorithms. Having defined a discrete Laplacian operator \mathbf{L} on the mesh, attenuating noise is obtained through a diffusion process:

$$\frac{\partial \mathbf{X}}{\partial t} = \lambda \mathbf{L}. \quad (2.2)$$

Higher power of \mathbf{L} , or combination of different powers of \mathbf{L} could also be used in the equation instead of \mathbf{L} . For example, $\lambda|\mu$ algorithm by Taubin et al. (1996) could be

regarded as using $(\lambda + \mu)\mathbf{L} - \lambda\mu\mathbf{L}^2$ within this framework where \mathbf{L} is just the umbrella operator in equation (2.1). By integrating the above equation, high frequency component of the mesh are smoothed, while the shape of the mesh are mostly kept. The authors show that Taubin et al. (1996) is actually solving the diffusion equation iteratively with an explicit Euler scheme

$$\mathbf{X}^{\tau+1} = (\mathbf{I} + \lambda dt \mathbf{L})\mathbf{X}^\tau, \quad \tau = 0, 1 \dots, \quad (2.3)$$

where \mathbf{X}^0 is the initial noisy mesh. Each integration here is linear in both time and memory. However, for this scheme to work, $\lambda dt < 1$ has to be met to assure numerical stability. As a result, the explicit method of Taubin et al. (1996) needs a large number of iterations to obtain a noticeable denoising effect and the overall cost makes it inefficient. To address this issue, the authors propose to use the implicit Euler scheme

$$(\mathbf{I} - \lambda dt \mathbf{L})\mathbf{X}^{\tau+1} = \mathbf{X}^\tau, \quad \tau = 0, 1 \dots, \quad (2.4)$$

where in each iteration the new mesh is obtained through solving a linear system. Since $(\mathbf{I} - \lambda dt \mathbf{L})$ is sparse, efficient numerical method (preconditioned bi-conjugate gradient) can be applied. With the implicit scheme providing unconditional stability, much fewer iterations are needed to get a desirable smoothing effect and the overall cost is much less than explicit method.

The authors also suggest a different way of preventing shrinkage which can be easily implemented for triangulated meshes. This is due to the fact that the volume of the triangulated mesh can be computed efficiently with a closed form formula. After each integration, volume of the new mesh is computed, and then all the vertex positions are scaled simultaneously to achieve the volume of original mesh.

More importantly, the authors propose a noise removal procedure called curvature flow that preserves the shape better than previous algorithms, by solving a different differential equation:

$$\frac{\partial \mathbf{x}_i}{\partial t} = -\bar{\kappa}_i \mathbf{n}_i, \quad (2.5)$$

where $\bar{\kappa}_i$ and \mathbf{n}_i denote the mean curvature and surface normal at point \mathbf{x}_i respectively.

Intuitively, by solving this equation, each vertex is only allowed to move along the surface normal while not allowed to drift tangentially, and the magnitude of the motion depends on how curvy the surface is at the vertex. The authors develop a closed formulation for estimating mean curvature from triangulated mesh data, and solve the diffusion equation within the implicit manner as well. Since the curvature flow here uses only the intrinsic property of the surface, it achieves the best smoothing effect while maintaining the overall shape.

As the use of points sets instead of meshes are becoming more popular, new algorithms are proposed for smoothing point set representations of 3-D objects. A big difference between point sets and meshes is that, connectivity information between vertices are no long available for point sets. As a result, algorithms that deal with point sets need to establish the neighborhood relation between vertices by themselves, usually using k-nearest neighbor or ϵ -ball criteria or a combination of both. This problem setting is closer to the ones we have in machine learning.

Lange and Polthier (2005) then denoise the point based surface with an anisotropic Laplacian defined on point sets which reflects more detailed curvature properties than the isotropic Laplacian used by previous papers. The advantage of this approach is that it can detect and enhance sharp geometric features of the surface such as edges and corners, while these features are not preserved in the isotropic fairing scheme. The algorithm first approximates a tangent space for each point with the local neighborhood, by solving a least squares minimization or equivalently an eigenvalue problem. With the normal vector at each point, the directional curvatures of each point in the direction of its neighbors and then a discrete version of Weingarten map at each vertex are estimated. The eigenvalues and eigenvectors of the Weingarten shape operator correspond to the principal curvatures and principal curvature directions at each vertex. So an typical edge can be detected if at some vertex one principal curvature is almost zero while the other one is larger than some threshold (provided by the user). Finally, the anisotropic Laplacian used in the diffusion process scheme is defined at each point \mathbf{x}_i as

$$\Delta_{|\mathbf{x}_i}^A = \text{div}_{|\mathbf{x}_i} \circ (A_i \cdot \nabla)_{|\mathbf{x}_i}, \quad (2.6)$$

where ∇ denotes the gradient operator and div denotes the divergence operator. A_i plays a crucial role in this setting because it provides the cut-off effect: a sample \mathbf{x}_j in the neighborhood of \mathbf{x}_i is not considered for $\Delta_{|\mathbf{x}_i}^A$ if the curvature at \mathbf{x}_i in direction \mathbf{x}_j is larger than some threshold. This is the key to preserving sharp features. If $A \equiv 1$, then the above definition reduces to isotropic Laplacian. It is interesting to see that, the idea of anisotropic diffusion is readily used in image processing to improve the performance of Gaussian filtering at the singular parts of a image and goes back to Perona and Malik (1990). Furthermore, Clarenz et al. (2004) combine anisotropic curvature evolution on noisy triangulated meshes and anisotropic diffusion of noisy texture on fixed surface, and solve the coupled problem for textured surfaces, enhancing both geometric features and texture features at the same time.

Other than the Laplacian operator, an alternative smoothing operator is suggested by Pauly et al. (2006) for point-based surfaces based on the implicit surface definition. Given a set of unstructured sample points $(\mathbf{x}_1, \dots, \mathbf{x}_N)$, a smooth surface is defined as the zero level set of a function

$$\mathcal{I}(\mathbf{x}) = \mathbf{n}(\mathbf{x}) \cdot (\mathbf{x} - \mathbf{a}(\mathbf{x})), \quad (2.7)$$

where $\mathbf{n}(\mathbf{x})$ is the surface normal at given coordinate \mathbf{x} and $\mathbf{a}(\mathbf{x})$ is the weighted average of sample points

$$\mathbf{a}(\mathbf{x}) = \frac{\sum_{i=1}^N \mathbf{x}_i \phi_{\mathbf{x}}(\|\mathbf{x}_i - \mathbf{x}\|)}{\sum_{i=1}^N \phi_{\mathbf{x}}(\|\mathbf{x}_i - \mathbf{x}\|)}. \quad (2.8)$$

The weighting function $\phi_{\mathbf{x}}$ is used to restrict the influence of faraway data points and is chosen to be the Gaussian kernel function, i.e., $\phi_{\mathbf{x}}(r) = e^{(-r^2/h_{\mathbf{x}}^2)}$ where the kernel width $h_{\mathbf{x}}$ should be set according to local feature size. Then it is natural to estimate the surface normal (of unit length) at \mathbf{x} by solving the following minimization problem

$$\begin{aligned} \min_{\mathbf{n}(\mathbf{x}) \in \mathbb{R}^D} \quad & \sum_{i=1}^N (\mathbf{n}(\mathbf{x}) \cdot (\mathbf{x}_i - \mathbf{a}(\mathbf{x})))^2 \phi_{\mathbf{x}}(\|\mathbf{x}_i - \mathbf{x}\|), \\ \text{s.t.} \quad & \|\mathbf{n}(\mathbf{x})\| = 1, \end{aligned} \quad (2.9)$$

the solution of which is the eigenvector corresponding to the smallest eigenvalue of a

weighted covariance matrix. After that, each data point \mathbf{x} near the surface can be used as initial value to solve the implicit function iteratively, and a projection operator could thus be derived and applied repeatedly until convergence. The implicit surface here is called the *weighted least squares* approximation.

Note that, there is a similar yet different method called *moving least squares*, used by Levin (2003) for approximation and interpolation of scattered data. This method generally provides smooth approximation of $(d - 1)$ -dimensional manifold in \mathbf{R}^d , $d \geq 2$. A projection operator is obtained in two steps. For any point \mathbf{r} near the surface, first compute a local hyperplane $H : \mathbf{n} \cdot (\mathbf{x} - \mathbf{q}) = 0$ where \mathbf{q} is a point on the hyperplane and \mathbf{n} is the surface normal. H is used to approximate the tangent hyperplane of the underlying surface near \mathbf{r} . And it is designed to have several properties, one of which is that \mathbf{q} is the projection of \mathbf{r} onto H . Meanwhile, an orthonormal coordinate system originated at \mathbf{q} is established on H . The second step is to compute a polynomial $p(\mathbf{y})$ in \mathbf{R}^{d-1} with some total degree m , through minimizing the weighted sum of squared errors

$$\sum_{i=1}^N (p(\mathbf{y}_i) - f_i)^2 \phi_{\mathbf{x}}(\|\mathbf{x}_i - \mathbf{q}\|), \quad (2.10)$$

where \mathbf{y}_i 's are the coordinates of the projections of \mathbf{x}_i 's onto H under the predefined orthonormal coordinate system, and f_i 's are the heights of \mathbf{x}_i 's over H . Therefore, the projection of \mathbf{x} on the smooth manifold is $P_m(\mathbf{x}) = \mathbf{q} + p(\mathbf{0})\mathbf{n}$.

In conclusion, the computer graphics papers focus on smoothing closed surfaces (they may give special treatment for non-closed surfaces or surfaces with holes, for example by Desbrun et al. (1999)) in 3D space. Many of them approximate differential operators on the surface and implement smoothing by solving some partial differential equation. Their techniques may not necessarily extend to the machine learning field since datasets in machine learning usually dwell in very high dimensional space (for example, an image could have tens of thousands of pixels), and many notions in computer graphic may not have correspondence there (for example, volume preservation and edge). However, the use of differential operators and diffusion process, approximation of tangent space, and the idea of preventing tangential drift are useful and applicable to high dimensional

data.

2.2.3 Computational Geometry algorithms

There are also research work trying to smooth point-based surface from a computational geometry point of view. Dey (2007) describes a suit of algorithms for reconstructing closed 1-D curve or 2-D surface from dense samples, from inadequately dense sample and from noisy samples. And the author provides mathematical proofs and guarantees for his method. The algorithms are mainly based on Voronoi diagrams and their dual Delaunay triangulations. For example, in essence, the surface normal is estimated by the direction in which a Voronoi cell elongates. However, this kind of algorithms may not carry over well to machine learning for two reasons. First, the Voronoi diagram and its dual work well only in low dimensions, and become computationally infeasible for high dimensions. Second, certain sampling conditions need to be satisfied for these computational geometry algorithms to work well. For example, the typical ϵ -uniform sampling condition requires that, for each point on the surface, there is a sample point within a small factor ϵ of its local feature size which is defined as the distance from the point to the medial axis. Under such conditions, the error between the estimate from Voronoi diagram and the true surface normal is guaranteed to be smaller than some bound depending on ϵ . We note that the concepts of medial axis and local feature size are often not applicable to problem settings in machine learning.

2.3 The Manifold Blurring Mean-shift (MBMS) Algorithm

Our manifold denoising algorithm is based on the following ideas:

- *Local clustering with Gaussian blurring mean-shift (GBMS)* (Figure 2.1): the blurring mean-shift update (Fukunaga and Hostetler, 1975) with unit step size

moves data points to the kernel average of their neighbors:

$$\mathbf{x}_n \leftarrow \sum_{m \in \mathcal{N}_n} \frac{G(\|(\mathbf{x}_n - \mathbf{x}_m)/\sigma\|^2)}{\sum_{m' \in \mathcal{N}_n} G(\|(\mathbf{x}_n - \mathbf{x}_{m'})/\sigma\|^2)} \mathbf{x}_m. \quad (2.11)$$

The average is over $\mathcal{N}_n = \{1, \dots, N\}$ (full graph) or the k nearest neighbors of \mathbf{x}_n (k -nn graph), and $G(t) = e^{-t/2}$. A single mean-shift step locally climbs up the kde defined by the data points, and after one step all points are updated so the dataset shrinks over iterations. As discussed previously, the process eventually converges to a state where all points coincide (Cheng, 1995) when using full graph, but it can be reliably stopped to produce good clusterings that depend on σ (Carreira-Perpiñán, 2006b, 2008).

- *Local tangent space estimation with PCA*: local PCA gives the best linear L -dimensional manifold in terms of reconstruction error (i.e., orthogonal projection on the manifold):

$$\min_{\boldsymbol{\mu}, \mathbf{U}} \sum_{m \in \mathcal{N}'_n} \|\mathbf{x}_m - (\mathbf{U}\mathbf{U}^T(\mathbf{x}_m - \boldsymbol{\mu}) + \boldsymbol{\mu})\|^2 \quad (2.12)$$

s.t. $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ with $\mathbf{U}_{D \times L}$, $\boldsymbol{\mu}_{D \times 1}$, whose solution is $\boldsymbol{\mu} = \mathbb{E}_{\mathcal{N}'_n} \{\mathbf{x}\}$ and \mathbf{U} = the leading L eigenvectors of $\text{cov}_{\mathcal{N}'_n} \{\mathbf{x}\}$. In general, \mathcal{N}'_n need not equal \mathcal{N}_n .

Although GBMS by itself has strong denoising power (controlled by σ and the number of iterations), this denoising is directed not only orthogonally to the manifold but also tangentially. This causes motion along the manifold, which changes important properties of the data that are not noise (for example, for a handwritten digit, it may change its style). It also causes strong shrinkage, first at the manifold boundaries but also within the manifold (see the example of Figure 2.2). Thus, while very useful for clustering, its applicability to manifold denoising is limited.

Our **Manifold Blurring Mean-shift (MBMS) algorithm** combines these two steps. At each iteration and for every data point \mathbf{x}_n , a *predictor averaging step* is computed using one step of GBMS with width σ . We can use the full graph ($\mathcal{N}_n = \{1, \dots, N\}$), or the k -nn graph ($\mathcal{N}_n = k$ nearest neighbors of \mathbf{x}_n) which has a similar effect as using a finite

support kernel (e.g. Epanechnikov kernel). This step is responsible for local denoising. Then, a *corrector projective step* is computed using the local PCA of dimensionality L on the k nearest neighbors of \mathbf{x}_n . This is responsible for local manifold structure, and removes the tangential component of the motion. The two steps are iterated until sufficient denoising is achieved while avoiding shrinkage and distortions of the manifold (see later). The complete algorithm is in Figure 2.1. We will refer to the algorithm as MBMSf if using the full graph for the GBMS step, MBMS k if using the k -nn graph (same k for the GBMS and PCA steps), or simply as MBMS when the difference is not relevant.

Besides GBMS (MBMS for $L = 0$), another particular case of MBMS is of special interest, which we call **Local Tangent Projection (LTP) algorithm** (Figure 2.1): it is MBMS k with $\sigma = \infty$, or equivalently it replaces the GBMS step with the mean of the k nearest neighbors. Thus, each point projects onto its local tangent space, and the process is iterated. It is simpler (one parameter less) and almost as effective as MBMS. Finally, two other particular cases are PCA, for $\sigma = \infty$ and $k = N$, and no denoising (the dataset will not change), for $L = D$ or $\sigma = 0$.

Note the following remarks. First, for given L , all versions of MBMS move points along the same direction (orthogonally) and only differ in the length of the motion. This length decreases monotonically with L because it is an orthogonal projection of the full-length motion (GBMS). The length increases with σ initially (more denoising) but may decrease for larger σ (as farther neighbors weigh in). Second, the GBMS coefficients in (2.11) are updated at each iteration; not doing so is faster, but gives worse results. Third, all the algorithms admit online versions by moving points asynchronously, i.e., by placing the step “ $\mathbf{x}_n \leftarrow \mathbf{x}_n + \partial\mathbf{x}_n$ ” inside the **for** loop.

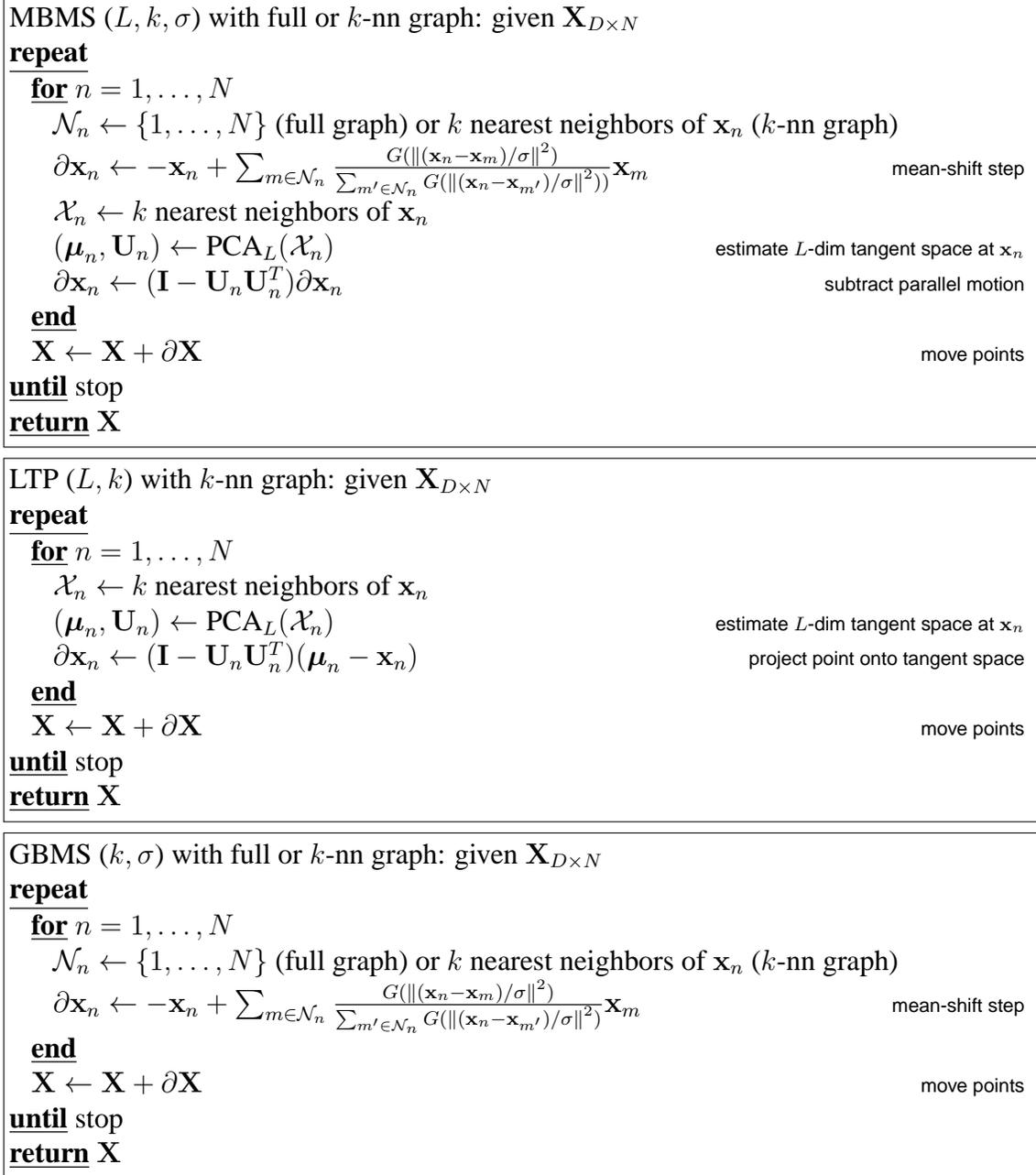


Figure 2.1: Manifold blurring mean-shift algorithm (MBMS) and its particular cases Local Tangent Projection (LTP, k -nn graph, $\sigma = \infty$) and Gaussian Blurring Mean-shift (GBMS, $L = 0$). \mathcal{N}_n contains all N points (full graph, MBMSf) or only \mathbf{x}_n 's nearest neighbors (k -nn graph, MBMSk).

2.3.1 Practicalities

How to set the parameters?

If MBMS is embedded into another algorithm (e.g. classification), the most effective way to set the parameters is to cross-validate them with a test set, although this does add significant computation if other classifier parameters need to be cross-validated too; we do this in our MNIST experiments. Otherwise, the parameters have an intuitive meaning, and based on our experience it seems easy to find good regions for them:

- σ is related to the level of local noise outside the manifold. The larger σ is, the stronger the denoising effect; but too large σ can distort the manifold shape over iterations because of the effect of curvature and of different branches of the manifold. Using a smaller σ is safer but will need more iterations. Using a k -nn graph is even safer, as the motion is limited to near the k nearest neighbors and allows larger σ , in fact $\sigma = \infty$ yields the LTP method.
- k is the number of nearest neighbors that estimates the local tangent space; this is the easiest to set and we find MBMS quite robust to it. It typically grows sublinearly with N .
- L is the local intrinsic dimension; it could be estimated (e.g. using the correlation dimension) but here we fix it. If L is too small, it produces more local clustering and can distort the manifold; still, it can achieve pretty good results for good σ ($L = 0$ is GBMS, which can achieve some reasonable denoising, after all). If L is too large, points will move little ($L = D$: no motion).
- Number of iterations: in our experience, a *few* (1–3) iterations (with suitable σ) achieve most of the denoising; more iterations can refine this and achieve a better result, but eventually shrinkage arises.

We find MBMS_f and MBMS_k/LTP with a few iterations give the best results in low and high dimensions, respectively, but using a k -nn graph (in particular LTP) is generally a safer and faster option that achieves very similar results to MBMS_f.

Stopping criterion

Because the denoising effect is strong, a practical indicator of whether we have achieved significant denoising while preventing shrinkage is the histogram over all data points of the orthogonal variance λ_{\perp} (the sum of the trailing $k - L$ eigenvalues of \mathbf{x}_n 's local covariance). Its mean decreases drastically in the first few iterations (and would converge cubically to zero in the Gaussian case), while the mean of the histogram of the tangential variance λ_{\parallel} decreases only slightly and stabilizes; see Figure 2.4. For curved manifolds, λ_{\perp} tends to a positive value dependent on the local curvature.

Computational complexity

Per iteration, this is $\mathcal{O}(N^2D + N(D + k) \min(D, k)^2)$, where the first term is for finding nearest neighbors and for the mean-shift step, and the second for the local PCAs. If one uses the k -nn graph and does not update the neighbors at each iteration (which affects the result little) then the first term is negligible and the cost per iteration is linear on N ; the one-off cost of computing the nearest neighbors is amortized if MBMS is followed by a spectral method for dimensionality reduction.

2.4 Experimental results

We demonstrate our MBMS algorithms on synthetic and real-world datasets in this section. We emphasize that although the algorithms proposed here can be applied to many areas as computer graphics and robotics, our focus here is to deal with general high dimensional manifold data of possibly unknown intrinsic dimensionality, and without ground truth neighborhood information.

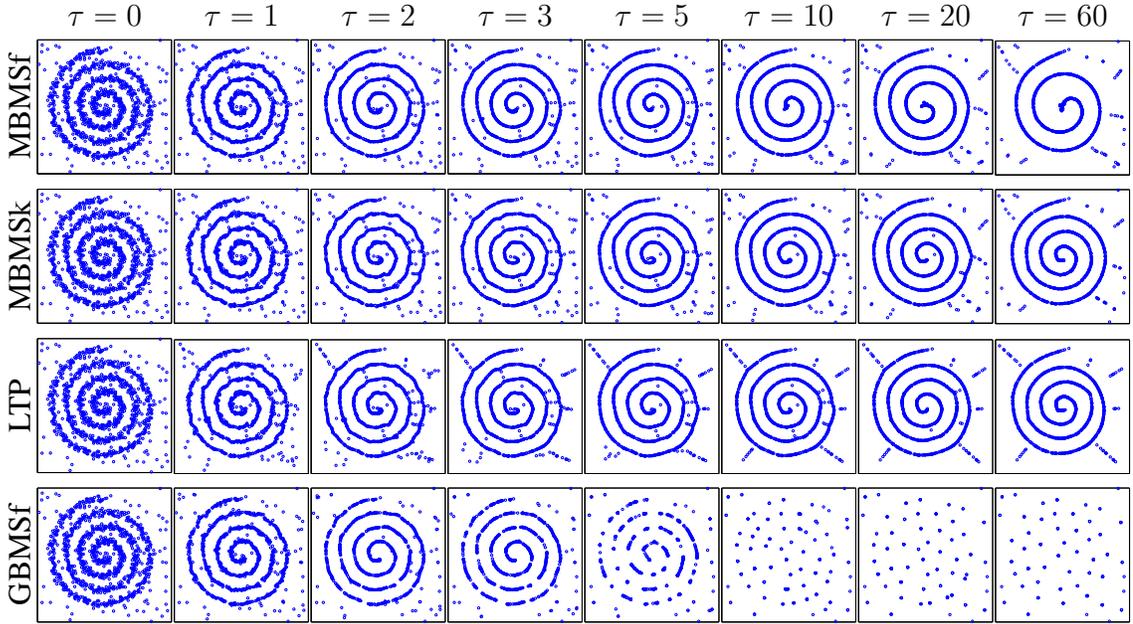


Figure 2.2: Denoising a spiral with outliers over iterations ($\tau = 0$ is the original dataset). Each box is the square $[-30, 30]^2$, where 100 outliers were uniformly added to an existing 1 000-point noisy spiral. Algorithms (L, k, σ) : (1, 10, 1.5) and full graph (MBMSf), (1, 10, 1.5) and k -nn graph (MBMSk), (1, 10, ∞) and k -nn graph (LTP), and $(0, \cdot, 1.5)$ and full graph (GBMSf).

2.4.1 Noisy spiral with outliers

Figure 2.2 shows four versions of MBMS with a noisy spiral dataset ($N = 1\,000$ points with Gaussian noise) with 10% outliers added uniformly. GBMS ($L = 0$) clusters points locally and, while it denoises the manifold, it also visibly shrinks it tangentially, so already from the first iterations the boundaries shrink and points form multiple clusters along the manifold. When using $L = 1$ in MBMS to account for a curve, in-manifold movement is prevented and so these undesirable effects are reduced. The three versions with $L = 1$ behave very similarly for the first 5–10 iterations, achieving excellent denoising while being remarkably unaffected by outliers. Visually, the full graph (MBMSf) looks best, although it begins to be affected by shrinking much earlier than the k -nn graph versions (MBMSk and LTP); the inside of the spiral slowly winds in, and also the whole spiral shrinks radially. MBMSk and LTP preserve the spiral shape and size for far longer: after 200 iterations only a small radial shrinkage occurs. The

reason is that the k -nn graph limits the influence on the mean-shift step of farther points (in regions with different curvature or even different branches); strong denoising (large σ) still occurs but is locally restricted. We have observed a similar behavior with other datasets.

After denoising for a few steps, outliers can be easily detected—the distance to their nearest neighbors is far larger than for non-outliers—and either removed, or projected on the tangent space of the k nearest neighbors on the manifold. The reason why they remain almost stationary and do not affect denoising of the mainstream points is simple. Points near the manifold (non-outliers) have no outliers as neighbors because the continuity of the manifold means all their neighbors will be near the manifold; neither the mean-shift step nor the tangent space estimation are affected, and these points move as if there were no outliers. Outliers have most neighbors somewhere near the manifold, and their tangent space is estimated as nearly orthogonal to the manifold at that point; they barely move, and remain isolated for many iterations (eventually they are denoised too, depending on how far they are from the manifold wrt k and σ). By this same reasoning, if MBMS is applied to disconnected manifolds, each will be denoised in isolation.

2.4.2 More complex shapes

Figure 2.3 shows a 1D manifold (two tangent ellipses) with a self-intersection, a gap, noise that varies depending on the manifold location, and a sharp density discontinuity. In spite of these varying conditions, MBMSf achieves very good denoising with a single (L, k, σ) value (row 1). Using the diffusion-map affinity normalization $\mathbf{D}^{-\alpha} \mathbf{W} \mathbf{D}^{-\alpha}$ of Coifman et al. (2005) with $\alpha = 1$ slightly improves the result (row 2), but with constant noise it has only negligible differences with our usual case ($\alpha = 0$).

2.4.3 Dimensionality reduction

Figure 2.4 shows the k -nn-graph version (MBMSk) with a noisy Swiss roll in 100 dimensions (97 of which are noise). Isomap (Tenenbaum et al., 2000) and particularly

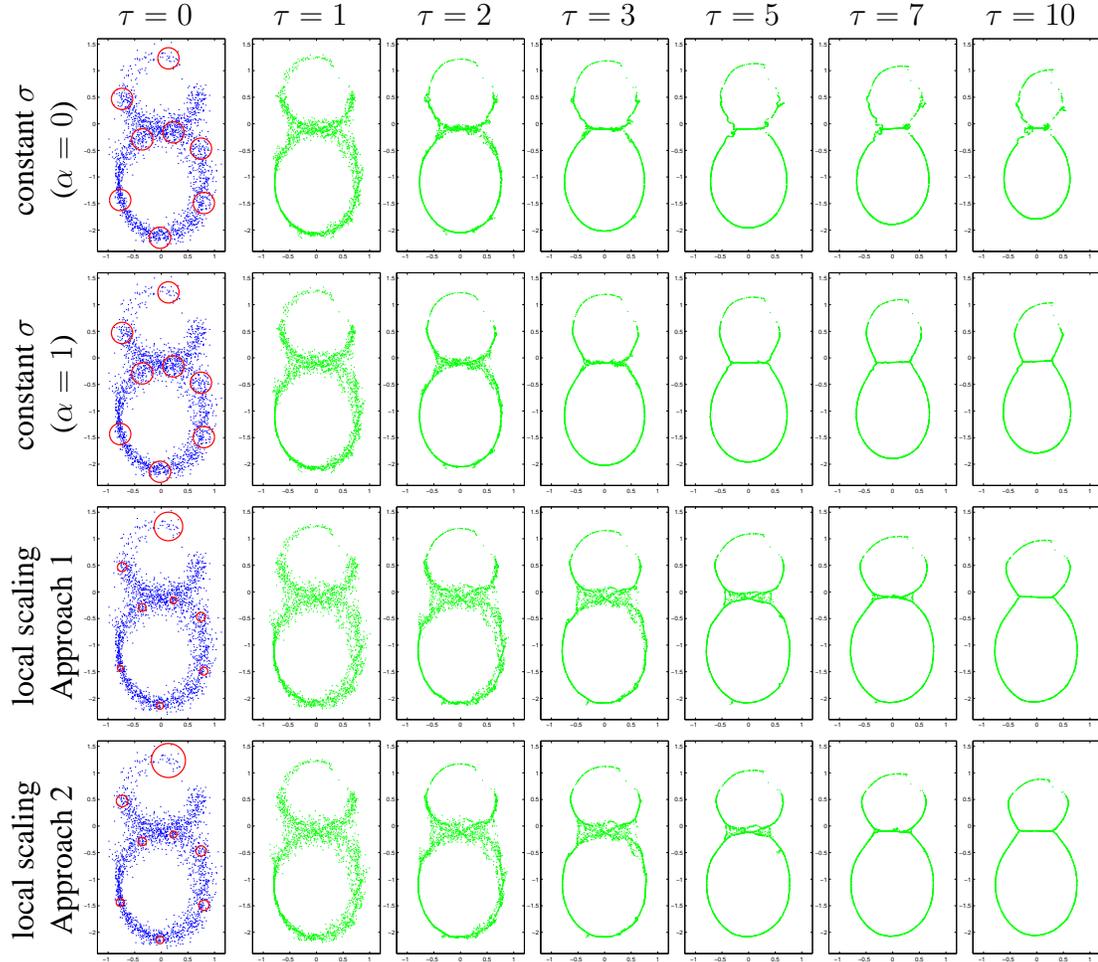


Figure 2.3: Denoising a complex shape with nonuniform density and noise with MBMSf ($L = 1, k = 35$) using the usual affinity ($\sigma = 0.2, \alpha = 0$, row 1), the diffusion-map affinity normalization ($\sigma = 0.2, \alpha = 1$, row 2), the local scaling approach 1 ($K = 35, \gamma = 1.4$, row 3), and local scaling approach 2 ($K = 70$, row 4). The upper partial ellipse has Gaussian noise of stdev 0.15 and the lower ellipse of stdev varying between 0 and to 0.2, with a sharp density discontinuity. The first column ($\tau = 0$) shows the original noisy dataset and the kernel widths (the radii of the red circles) used by different approaches at several points (centers of the red circles) in different density regions.

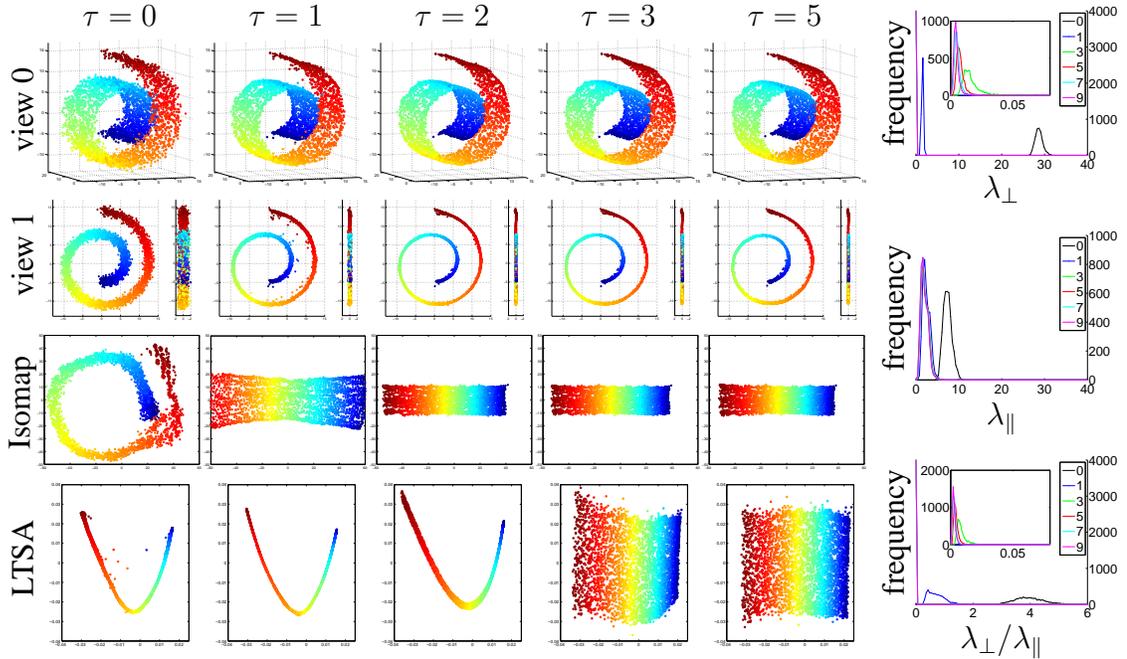


Figure 2.4: Dimensionality reduction with Isomap and LTSA for different iterations of MBMSk denoising (10-nearest-neighbor graph, $L = 2$, $k = 30$, $\sigma = 5$). $\tau = 0$ is the original Swiss roll dataset ($N = 4000$ points) lifted to 100 dimensions with additive Gaussian noise of stdev 0.6 in each dimension. Isomap/LTSA used a 10-nn graph. Isomap’s residual variances (Tenenbaum et al., 2000) ($\tau = 0, 1, 2, 3, 5$): 0.3128, 0.0030, 0.0002, 0.0002, 0.0003. View 0 shows dimensions 1–3; view 1 shows dimensions 1, 2 (left subplot) and 2, 4 (right subplot). Right column: histograms over all data points of the normal, tangential, and normal/tangential ratio of the variances; the curves correspond to the iterations $\tau = 0, 1, 3, 5, 7, 9$, and the insets for λ_{\perp} and $\lambda_{\perp}/\lambda_{\parallel}$ blow up the bins near zero (which contain all points for $\tau \geq 2$).

LTSA (Zhang and Zha, 2004) are sensitive to noise and to shortcuts in the neighborhood graph, but these are eliminated by MBMS. Excellent embeddings result for a wide range of iterations, and one can trade off a little more denoising with a little more shrinkage. In general, and depending on the level of noise, 2–3 iterations are often enough. The histograms show that the tangent space eigenvalues λ_{\parallel} change little over iterations, i.e., there is little in-manifold motion. However, the normal space eigenvalues λ_{\perp} drop drastically in the first 3 iterations (the histogram is a spike at almost zero) and then stay roughly constant (they do not become exactly zero because of the manifold curvature), indicating strong denoising orthogonal to the manifold, and signaling a good stopping point. We repeated the experiment by adding 10% outliers within a box bounding the

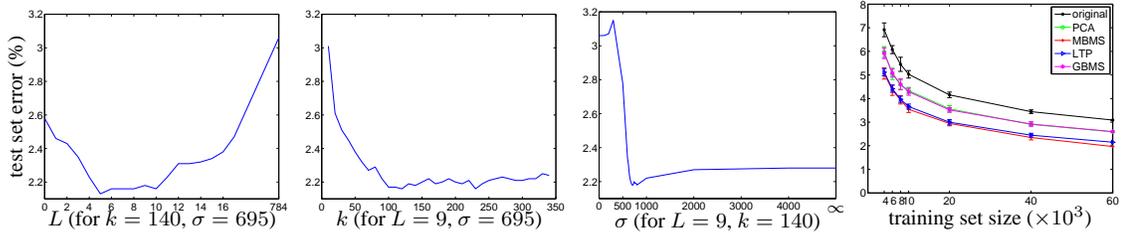


Figure 2.5: *Left 3 plots:* 5-fold cross-validation error (%) curves with a nearest-neighbor classifier on the entire MNIST training dataset (60k points, thus each fold trains on 48k and tests on 12k) using MBMSk; we selected $L = 9$, $k = 140$, $\sigma = 695$ as final values. *Right plot:* denoising and classification of the MNIST test set (10k points), by training on the entire training set (rightmost value) and also on smaller subsets of it (errorbars over 10 random subsets). Algorithms (L, k, σ) , all using a k -nn graph: MBMSk (9, 140, 695), LTP (9, 140, ∞), GBMS (0, 140, 600), and PCA ($L = 41$).

Swiss roll with essentially identical results (points near the manifold are denoised, outliers remain stationary), demonstrating the robustness of MBMS.

2.4.4 Classification of MNIST digits

It is reasonable to assume that much of the essential character (style, thickness, etc.) of a handwritten digit can be explained by a small number of degrees of freedom, so that MBMS denoising might preserve such a manifold while removing other types of noise; and that this may improve classification accuracy. Our setup is as follows. We use a nearest-neighbor classifier (like MBMS, a nonparametric method), which allows us to focus on the performance of MBMS without classifier-specific effects due to local optima, model selection, etc. As denoising methods we use PCA (i.e., projecting the data onto the L principal components’ manifold) and 3 versions of MBMS using the k -nn graph and a single iteration: MBMSk, LTP and GBMS. We estimate (approximately) optimal parameters by 5-fold cross-validation by searching over a grid, denoising separately each class of the training fold ($N = 48\,000$ grayscale images of dimension $D = 784$, or 28×28 pixels) and measuring the classification error on the test fold (12 000 digits). For classification, the test points are fed directly (without denoising) to the nearest-neighbor classifier. Figure 2.5 (left 3 plots) shows the MBMSk error curves over L , k and σ ; notice how MBMSk improves the baseline error (no denoising, also

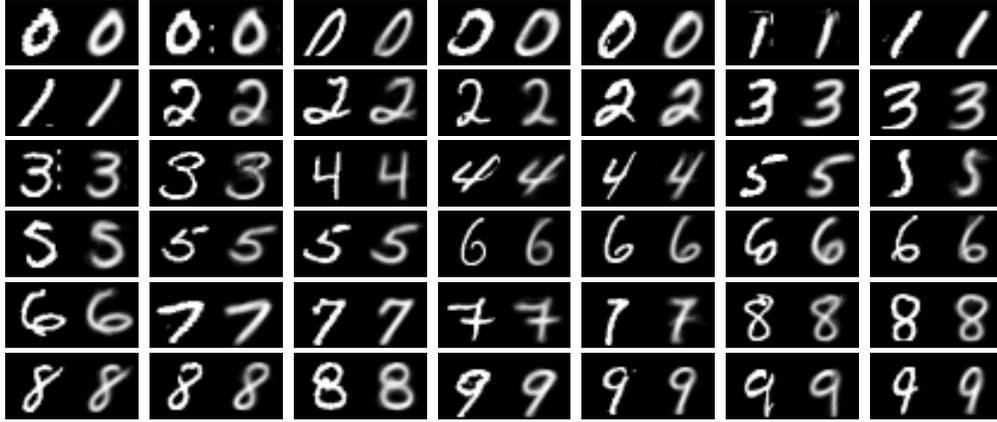


Figure 2.6: Sample pairs of (original,denoised) images from the training set. A few (2.62%) grayscale values outside the $[0, 255]$ training range have been clipped for visualization.

achieved by $L = D = 784$ or $\sigma = 0$) of 3.06% over a very wide range of (L, k, σ) . We chose $(9, 140, 695)$ and trained the models on the entire training set (60k points); Figure 2.5 (right plot) shows the test set classification error. MBMSk achieves 1.97% (a 36% relative decrease over the baseline of 3.09%); LTP $(9, 140, \infty)$ achieves a slightly larger error of 2.15% (30% relative decrease). GBMS and PCA also improve over the baseline but far less (2.59%, 14% decrease). These results are consistently confirmed over smaller training sets, even up to $N = 4000$ (right panel); we used the same parameters as for the entire set. The methods combining both clustering and manifold structure at the local level (MBMSk and LTP) are the clear winners. Judging from the trend of the curves, the relative error decrease would still grow with the training set size.

Other options also reduced the error, but less so (however, in all these cases we used the same parameters as above $(9, 140, 695)$, which are not optimal anymore). Denoising each test point (with one MBMSk iteration using the entire denoised training set): 2.23%. Denoising each test point but with the original training set: 2.42%. Denoising the entire training set without class information: 2.89%. The beneficial effect of MBMSk denoising in one way or another is clear.

Figure 2.6 shows training images before and after denoising. The most obvious change is that the digits look smoother (as if they had been anti-aliased to reduce pixelation)

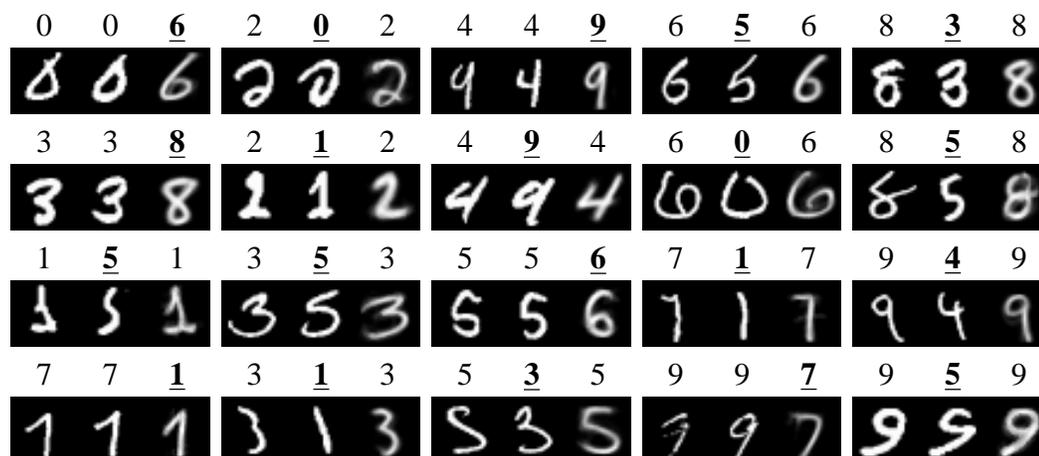


Figure 2.7: Some misclassified images. Each triplet is (test,original-nearest-neighbor,denoised-nearest-neighbor) and the corresponding label is above each image, with errors underlined. After denoising there are fewer errors, some of which are arguably wrong ground-truth labels.

and easier to read; comparing the original *0 1 2 3 4 5 6 7 8 9* vs the denoised *0 1 2 3 4 5 6 7 8 9*, one sees this would help classification. While this smoothing homogenizes the digits somewhat, it preserves distinctive style aspects of each; excessive smoothing would turn each class into a single prototype image, and result in a Euclidean distance classifier (the method of Hein and Maier 2007 shows oversmoothing). MBMSk performs a sophisticated denoising (very different from simple averaging or filtering) by intelligently closing loops, removing or shortening spurious strokes, enlarging holes, removing speckle noise and, in general, subtly reshaping the digits while respecting their orientation, slant and thickness. We emphasize that we did not do any preprocessing of the data, and in particular no image-based preprocessing such as tangent distance, deskewing, or centering the images by bounding box (known to improve the nearest-neighbor classifier LeCun et al., 1998). MBMS does not know that the data are images, and would give the same result if the pixels were reshuffled. Figure 2.7 shows misclassified images.

2.5 Discussion

In previous sections, we have stuck to the simple algorithm in Figure 2.1 which involves three parameters (L, k, σ) , each playing different role in denoising. We now discuss some extensions which may improve the performance under certain circumstances.

2.5.1 Different operators

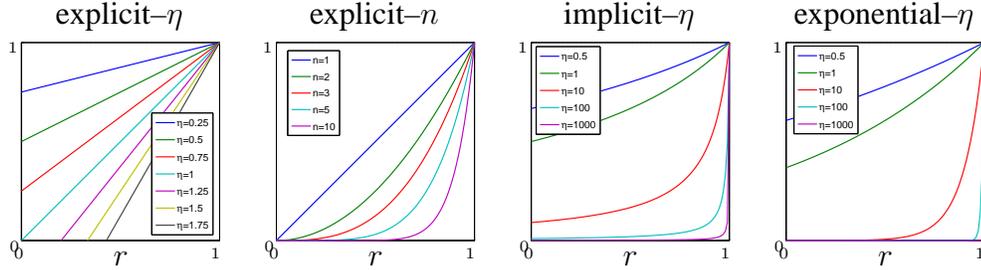
At each iteration, the GBMS predictor step builds the $N \times N$ affinity matrix $\mathbf{W} = (\exp(-\|\mathbf{x}_n - \mathbf{x}_m\|^2 / 2\sigma^2))_{nm}$, and the degree matrix $\mathbf{D} = \text{diag}(\sum_{n=1}^N w_{nm})$, which define a random-walk matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$. The GBMS update can thus be written concisely as $\bar{\mathbf{X}} = \mathbf{P}\mathbf{X}$. Note that \mathbf{P} is closely related with the well-known graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$ (or its normalized version $\mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$), which has been used extensively for denoising (e.g. Desbrun et al., 1999; Hein and Maier, 2007; Taubin, 1995), spectral clustering (Shi and Malik, 2000), dimension reduction (Belkin and Niyogi, 2003), and manifold regularization (Belkin et al., 2006). In summary, all these algorithms make essential use of the eigenspace of \mathbf{P} or \mathbf{L} .

The denoising effect of GBMS can be considered as applying a low-pass filter to the dataset \mathbf{X} . It is therefore straightforward to use a different operator $\phi(\mathbf{P})$ in place of \mathbf{P} , such that $\phi(\mathbf{P})$ has the same eigenvectors as \mathbf{P} , while tailoring the frequency distribution of \mathbf{X} in a desirable way (e.g., to approximate an ideal low-pass filter, Taubin, 1995). This approach has already been investigated by Carreira-Perpiñán (2008) in the context of clustering. Carreira-Perpiñán (2008) has explored different forms of ϕ , including explicit, implicit, rational and exponential functions, and noticed that while the different forms achieve approximately the same clustering result, their convergence rate and runtime differ significantly.

Under the same framework of Carreira-Perpiñán (2008), we investigate the following different operators $\phi(\mathbf{P})$ in the predictor step of MBMS (the corrector step is carried out as before):

Table 2.1: Summary of operators' properties and graph of their defining function $\phi(r)$, $r \in (0, 1)$. This table is taken from Carreira-Perpiñán (2008).

Method	$\phi(r)$	parameter range	convergence order	cost per iteration
explicit- η	$1 - \eta + \eta r$	$\eta \in (0, 2] \setminus \{1\}$	1	1
GBMS ($\eta = 1$)	r	$\eta = 1$	3	1
explicit- n	r^n	$n = 1, 2, 3 \dots$	$2n + 1$	n
implicit- η	$\frac{1}{1 + \eta - \eta r}$	$\eta \in (0, \infty)$	1	$\frac{1}{3} \frac{N}{D}$
exponential- η	$e^{-\eta(1-r)}$	$\eta \in (0, \infty)$	1	$2 \frac{N}{D}$



- Explicit- η : $\phi(\mathbf{P}) = (1 - \eta)\mathbf{I} + \eta\mathbf{P}$ for $\eta \in (0, 2]$.
- Explicit- n : $\phi(\mathbf{P}) = \mathbf{P}^n$ for $n = 1, 2, 3, \dots$
- Implicit- η : $\phi(\mathbf{P}) = ((1 + \eta)\mathbf{I} - \eta\mathbf{P})^{-1}$ for $\eta > 0$.
- Exponential- η : $\phi(\mathbf{P}) = e^{-\eta(\mathbf{I}-\mathbf{P})}$ for $\eta > 0$, where the matrix exponential is defined as $e^{\mathbf{A}} = \sum_{i=0}^{\infty} \mathbf{A}^i / i!$ if the series converges.

We list their corresponding convergence order and cost per iteration (proportional to that of usual GBMS, assuming full graph is used) in Table 2.1.

We demonstrate the above four different operators on the noisy spiral dataset in Figure 2.2, and use parameters such that their clustering behavior deviate from the usual GBMS. Namely, we use explicit- η with $\eta=1.8$ (over-relaxation), explicit- n with $n = 10$, implicit- η with $\eta = 10$, exponential- η with $\eta = 10$. All operators are run with MBMS ($L = 1$) and GBMSk ($L = 0$) with other parameters fixed at ($k = 10, \sigma = 1.5$), and the denoising results over iterations are shown in Figure 2.8 (computing \mathbf{P} with full graph) and Figure 2.9 (computing \mathbf{P} with k -nn graph). We make a few observations from the results.

- In terms of clustering behavior, different operators leads to similar final result.

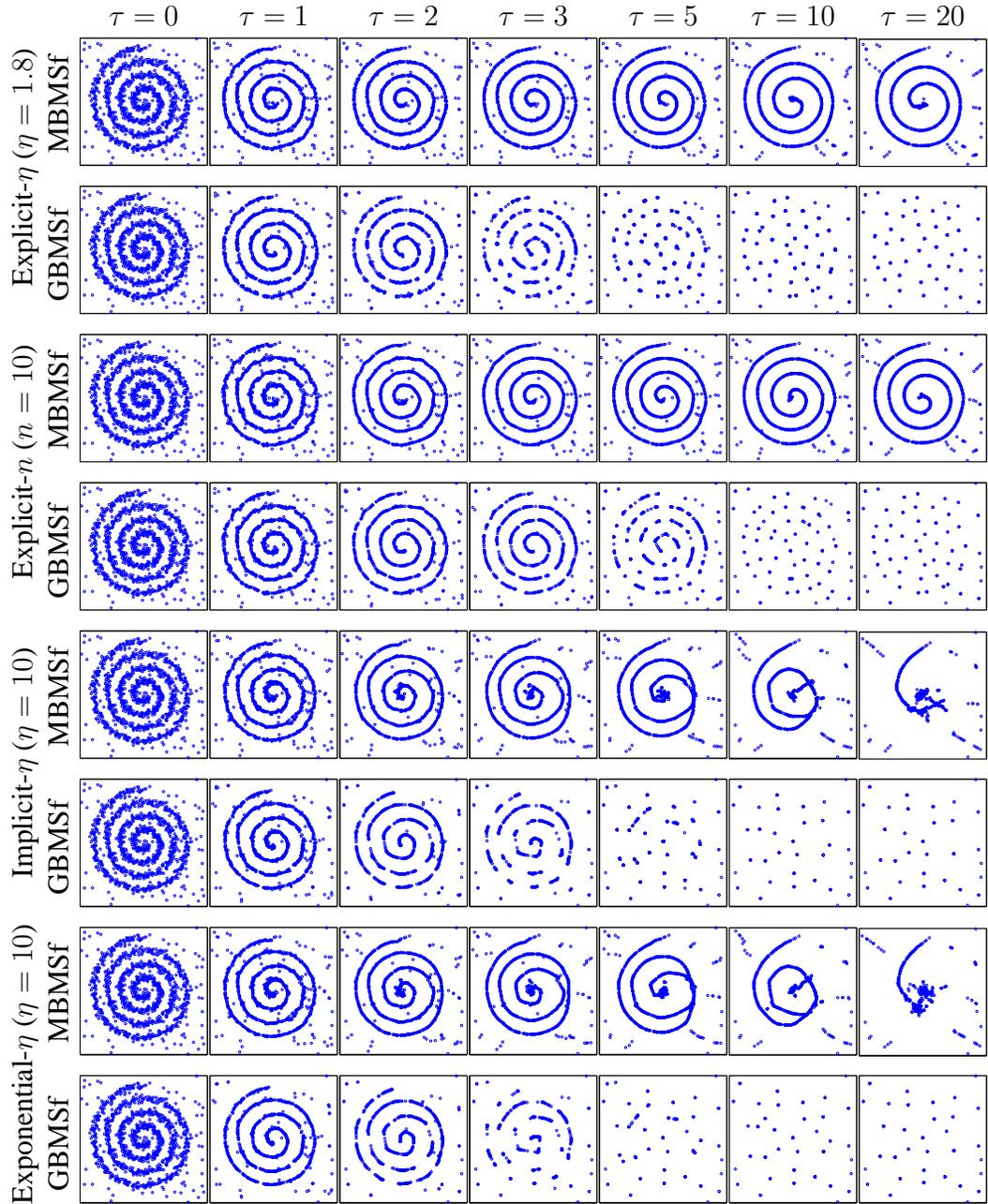


Figure 2.8: Denoising a spiral with outliers (same dataset as in Figure 2.2) using different operators. All operators are run with MBMSf (odd rows, $L = 1$) and GBMSf (even rows, $L = 0$) with other parameters fixed at ($k = 10, \sigma = 1.5$).

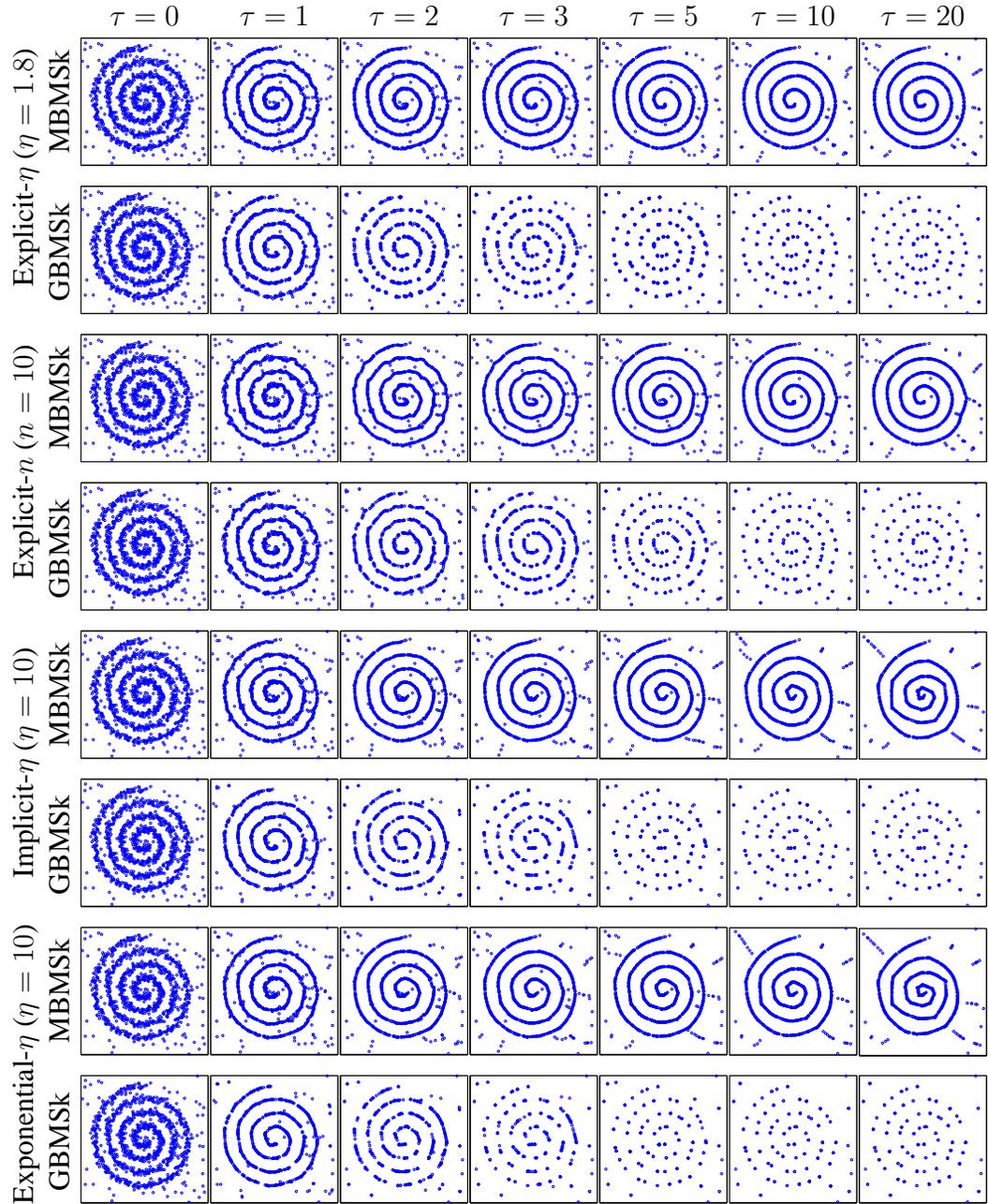


Figure 2.9: Denoising a spiral with outliers (same dataset as in Figure 2.2) using different operators. All operators are run with MBMSk (odd rows, $L = 1$) and GBMSk (even rows, $L = 0$) with other parameters fixed at ($k = 10, \sigma = 1.5$).

Comparing the strength of denoising per iteration, the ranking of operators is exponential $>$ implicit $>$ explicit, which can be seen from the iterations used by GBMSf or GBMSk to converge. And this is to be expected because the exponential and implicit operator actually apply higher orders of explicit operators implicitly through their Taylor expansions. But the stronger denoising effect comes at the price of much higher computational cost. We also notice that, when using full graph in the predictor step, the implicit and exponential operator produce too strong denoising effect to be corrected by local PCA. Overall, MBMSk leads to smaller shrinkage and preserves the manifold shape better than MBMSf.

- Generally speaking, with the k -nn graph restricting clustering to be local, and the following corrector step further limiting the motion, the difference between different operators of MBMSk are relatively small (much smaller than the difference between MBMS and GBMS). The explicit operators, whose result are very similar to that of MBMSk in Figure 2.2, preserve the shape best and produce little shrinkage on this dataset. The implicit operator and exponential operator have stronger denoising power, clearly seen in early iterations ($\tau = 1$ and 2), but cause more shrinkage in later iterations. And since they are much slower to compute, we conclude that explicit operators are practically more preferable.
- It is interesting to note that using the explicit- n operator is equivalent as running the GBMS predictor step with fixed \mathbf{P} for n times. Its good performance indicates that we can run the predictor step with fixed operator for several iterations before switching to the corrector step. In this way we reduce the frequency of updating pairwise affinities and computing local PCAs, both of which are costly compared to the predictor step, especially when the predictor step uses a sparse graph.

2.5.2 Local scaling

We have so far used a common σ value for the entire dataset in MBMS. When the dataset has different sampling density in different regions, using an individual, localized σ for each data point places more emphasis on the geometry of the underlying manifold.

The same idea is used by Zelnik-Manor and Perona (2005) in constructing the pairwise affinities between data points for spectral clustering, where σ is simply set at each point to be the distance to its 7th nearest neighbor, as an easy rule of thumb. In the following, we investigate the local scaling issue based on the more recent work of Vladymyrov and Carreira-Perpiñán (2013), which extends previous work by Hinton and Roweis (2003). In the proposed entropic affinity algorithm, σ is chosen for each point such that it has a distribution over neighbors with a desired perplexity $\log K$, or effective number of neighbors K (to be distinguished from the local neighborhood size k in the corrector step). Formally, we can define a discrete distribution $p(\mathbf{x}; \sigma)$ at any location $\mathbf{x} \in \mathbb{R}^D$ over the N data points

$$p_n(\mathbf{x}; \sigma) = \frac{G(\|(\mathbf{x} - \mathbf{x}_n)/\sigma\|^2)}{\sum_{m=1}^N G(\|(\mathbf{x} - \mathbf{x}_m)/\sigma\|^2)}. \quad (2.13)$$

Notice these probabilities have exactly the same form as the normalized affinity used in mean-shift update (1.2). The intuition behind entropic affinity is to have $p(\mathbf{x}; \sigma)$ “provide the same surprise as if we were to choose among K equiprobable neighbors”. As a result, the entropy of $p(\mathbf{x}; \sigma)$, which is a nonlinear function of σ , should match the entropy of the K equiprobable neighbors case, which is $\log K$. Given the user parameter K , it reduces to a root finding problem to compute σ , for which Vladymyrov and Carreira-Perpiñán (2013) have developed efficient derivative-based procedure. It is shown that the entropic affinity algorithm finds better local scales than the rule of thumb approach, while still being computationally efficient.

In this section, we investigate two different approaches to achieve local scaling in MBMS.

- *Approach 1:* we first obtain an individual σ_i for \mathbf{x}_i , $i = 1, \dots, N$ using the entropic affinity algorithm with certain K , and then plug in $\Sigma_i = (\gamma\sigma_i)^2\mathbf{I}$ as the covariance matrix at each \mathbf{x}_i in the kde. Therefore, there are two user parameters in this approach: K is used to estimate the “magnitude” of the local scale, while the extra factor $\gamma > 0$ is used to control the scale globally for smoothing.
- *Approach 2:* we use K as a tuning parameter instead of σ in the predictor step, i.e., we simply use the σ_i returned by entropic affinity as the kernel width at \mathbf{x}_i ,

and vary K globally to achieve optimal denoising. Notice that K can take any continuous value in $[1, N]$ in entropic affinity, and a larger K implies a larger σ value, which in turn implies a stronger denoising effect of mean-shift. The σ_i returned by entropic affinity using the same K value will be different for different point, which reflect exactly the variation in local sampling density.

In this section, we set $K = k$, the neighborhood size used in corrector step in Approach 1 for simplicity. Intuitively, the value k reflects our belief on the local geometry of each point to be a L dimensional subspace approximately. But we emphasize that K and k have different meanings and they do not need to be the same. As a result of this setting, we have one parameter (γ vs. K) to be tuned for each approach in the experiments below. The corresponding mean-shift update is derived using the new kde and applied in the MBMS predictor step.

We demonstrate them on the complex shaped dataset in Figure 2.3, which has nonuniform density and noise in different regions. We have shown previously that, using unnormalized affinities, a common σ value that is appropriate for the highly noisy region will be too large for the less noisy region and cause significant shrinkage there. In approach 1, we use the entropic affinity algorithm to find the individual σ value that gives $k = 35$ effective neighbors for each point and rescale them with $\gamma = 1.4$; in approach 2, we use $K = 70$ effective neighbors. γ and K are tuned in each approach to achieve good denoising and small shrinkage. Other than that, we run MBMSf algorithm using the same parameters as before ($k = 35$, $L = 1$), without any affinity normalization ($\alpha = 0$). The denoised dataset at different iterations are shown in Figure 2.3 (row 3 and row 4). The two approaches produce very similar results. Clearly, the manifold shape is well-preserved, as in MBMSf with the diffusion-map affinity normalization ($\alpha = 1$), and shrinkage is even less noticeable than before, thanks to the individual scales that adapt to local density.

Our results essentially show that better density estimate used in the predictor step gives better denoising. With the above approaches, we have used localized spherical (isotropic) covariance matrix $\Sigma_i = (\sigma_i)^2 \mathbf{I}$ in the kde instead of the homoscedastic formulation (1.2). Potentially, further improvement can be obtained with non-spherical or full co-

variance matrix. Before finishing this section, we discuss the interesting connection between MBMS and the Manifold Parzen Windows by Vincent and Bengio (2003). The authors propose to use manifold aligned kernel for density estimation and classification, where the important observation is that spherical Gaussian density spread equally along all directions in the input space, and that it wastes a lot of probability mass in irrelevant region when the true data density is close to a nonlinear lower dimensional manifold. Their solution is to estimate for each point an individual covariance matrix or a “pancake”-shaped Gaussian that spreads mostly along the manifold. The manifold structure in this work is estimated by first computing a local covariance matrix and then extracting the leading eigenvalues/eigenvectors—exactly the same way as we do in the corrector step. It is easy to see that the resulting density estimate will concentrate near the manifold. A disadvantage of this approach is that it requires storing the individual covariance matrices for all data points (or at least their leading eigenvalues/eigenvectors) to just evaluate the density at a test point.

2.5.3 Estimation of intrinsic dimensionality

Estimation of intrinsic dimensionality L of a data manifold is a long standing problem in manifold learning. It also plays a central role in MBMS—it distinguishes the manifold structure from noise in the corrector step. We have observed empirically that the value of L can influence the performance of MBMS significantly (e.g., compare MBMS with GBMS). Underestimating L leads to shrinkage of true signal, while overestimating it results in inadequate denoising. In this section, we investigate several approaches for estimating L in MBMS.

Existing methods of estimating intrinsic dimensionality can be roughly categorized into geometric methods and eigenvalue methods. The common approach taken by geometric methods is to design quantities that vary according to the intrinsic dimensionality L rather than the input dimensionality D . Example quantities of this kind include: the fraction of pairs of points that are within certain distance from each other (correlation dimension, Camastra and Vinciarelli, 2002; Grassberger and Procaccia, 1983), the mini-

num number of balls of certain radius that cover the dataset (capacity dimension, Kégl, 2003), Euclidean distance from a data point to its k -th nearest neighbor (Pettis et al., 1979), and length of the minimal spanning tree on the geodesic nearest-neighbor distance (Costa and Hero, 2004). In these methods, L is recovered by estimating the growth of the designed quantities over their arguments from finite samples. Levina and Bickel (2005) propose to model the number of data points within certain distance from a point in space as a Poisson process, and develop a maximum likelihood estimator for the rate parameter which is a function of L .

Eigenvalue methods estimate intrinsic dimensionality of dataset by looking for clear-cut boundary in the spectrum computed by PCA (Bruske and Sommer, 1998; Fukunaga and Olsen, 1971). Global PCA tends to overestimate L due to the curvature of nonlinear manifold, and is thus not suitable here. And since we are already computing local PCAs in the MBMS corrector step, eigen-analysis on the local covariance becomes a natural choice for us. Notice the size of the local neighborhood k is an important parameter here, which should be chosen such that the neighborhood of each point approximately lie on L -dimensional subspace. Using a too small k , the neighborhood may appear D dimensional due to the noise; using a too large k , the neighborhood is no longer local, and curvature may obscure the structure (Brand, 2003).

In this section, we investigate experimentally two methods for automatic estimation of L . The first one is the maximum likelihood method (denoted by MLE) by Levina and Bickel (2005), which is a geometric method and is shown to achieve good bias and variance balance. This method has one user parameter k , the size of neighborhood within which the Poisson process model is employed. The second method is a simple eigenvalue method through local PCA (denoted by EIG). Suppose the local covariance matrix computed on the k nearest neighbors of a data point has eigenvalues $\lambda_1 \geq \dots \geq \lambda_D$. Then our EIG estimate at this point looks for the largest eigen-gap:

$$L = \arg \max_{l=1, \dots, D-1} \lambda_l - \lambda_{l+1}. \quad (2.14)$$

This is essentially assuming that the smallest variance in direction parallel to the manifold is much larger than the largest variance in direction orthogonal to the manifold

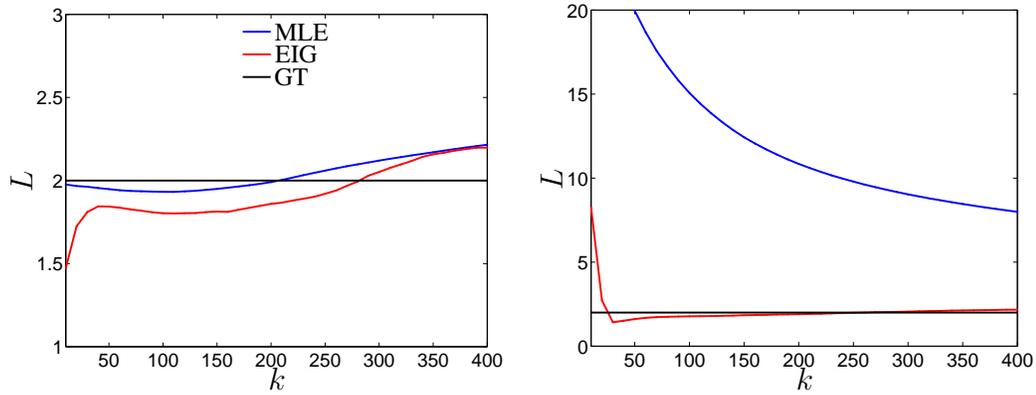


Figure 2.10: Estimation of intrinsic dimensionality L of the 100D swissroll dataset, both in the noiseless case (left) and noisy case (right). We plot the mean of estimated L at all data points vs. the parameter k used by each method. The black line denotes the ground-truth dimensionality $L = 2$.

within the neighborhood.

We first demonstrate MLE and EIG on the 100D swissroll example used in Section 2.4.3, where $N = 4000$ and $L = 2$. The first 3 dimensions of the dataset is sampled from a swissroll while the rest dimensions are pure random noise. With fixed parameter k , we can use each method to obtain an estimate of L at each point. The mean of the pointwise estimate on the noisy and noiseless dataset is shown in Figure 2.10 for a range of k values. It is clear that, both methods work well with noiseless dataset (left plot). But with the existence of noise, MLE significantly overestimates L for all values of k , whereas the EIG estimate is much more accurate and stable.

We also vary the input dimensionality D by using the first D dimensions of the 100D swissroll dataset, which effectively change the noise level. The estimated dimensionality on the dataset for both methods at $k = 150$ are shown in Figure 2.11. We observe that the MLE estimate deteriorates quickly as the noise level increases, while EIG always gives an estimate close to $L = 2$ regardless of the input dimensionality. This is because the gap (2.14) remains the same no matter how many dimensions of (moderate) noise is added.

Next, we use the intrinsic dimensionality estimated by each method in our MBMS corrector step, where each point is allowed to have a different estimate. We do not directly

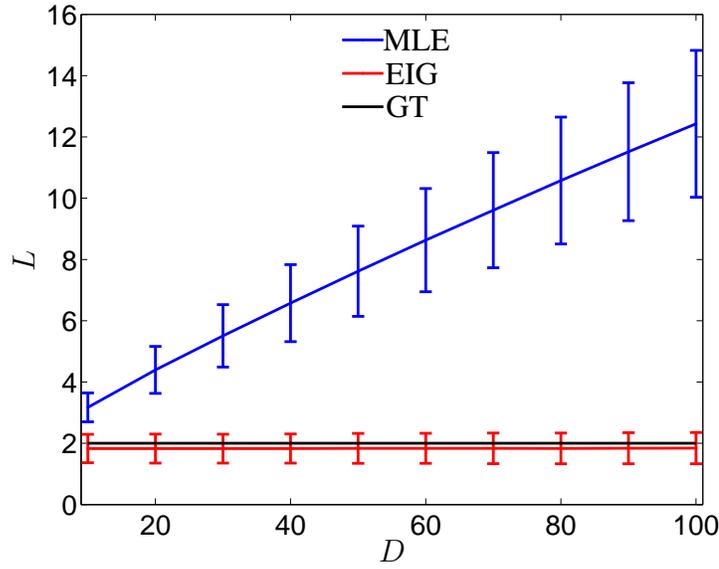


Figure 2.11: Estimation of intrinsic dimensionality L on the swissroll dataset for different input dimensionalities D . We show the mean and standard deviation of estimated L at all data points obtained by MLE and EIG using $k = 150$. The black line denotes the ground-truth dimensionality $L = 2$.

use the pointwise estimate from each method. Instead, a majority vote of L among the the k -nearest-neighbors of each point is used as a smoother version of the estimate. We find this helps improve the estimate in practice. This completely frees the user from setting a global L parameter. We also update the estimate at each iteration. With appropriate σ parameter, and as the noise level get reduced over iterations, it should become easier to estimate the intrinsic dimensionality. In the ideal case where both L and the tangent space are correctly estimated, the largest eigenvalue gap in (2.14) will be always found at L over iterations, since orthogonal variance decreases quickly while parallel variance do not change much under MBMS.

Figure 2.12 and Figure 2.13 show the denoising and dimension reduction results on the 100D swissroll dataset, where parameters σ and k are tuned for each method to achieve good estimation of L and embedding. Apparently, MLE leads to insufficient denoising in the beginning and many dimensions of noise are regarded as manifold structure mistakenly due to the overestimation of L . Even though the estimation of MLE becomes better at later iterations ($\tau = 3$ and 5), the manifold structure is not well maintained

because not all points are denoised at the same pace and tangent space estimation becomes unreliable. On the other hand, EIG gives much more accurate estimation of L in the beginning. Then it tends to underestimate L as noise level decreases, because the local neighborhood at certain points approximately lie on a subspace of dimensionality lower than the true L . MLE and particularly EIG have a noticeable effect on the manifold boundary, where points accumulate and produce a 1D alignment (see Figure 2.14 for a comparison). From the Isomap embedding, it is also clear that there are local groups of latent representations (especially around the boundary) that approximately lie on a 1D line because the estimated L is 1 within those groups. Overall, MLE produces stronger distortion of the manifold, and EIG results in much better denoising effect in this example.

In this section, we have investigated different approaches of estimating the intrinsic dimensionality of dataset with manifold structure. We find that the geometric method MLE is more sensitive to noise, and the simple eigenvalue method EIG performs well with MBMS for denoising. Potentially they can be applied to datasets involving multiple manifolds where each manifold may even have a different dimensionality.

2.6 Conclusion

With adequate parameter values, the proposed MBMS algorithm is very effective at denoising in a handful of iterations a dataset with low-dimensional structure, even with extreme outliers, and causing very small shrinkage or manifold distortion. It is non-parametric and deterministic (no local optima); its only user parameters (L, k, σ) are intuitive and good regions for them seem easy to find. We also proposed LTP (local tangent projection), a particular, simple case of MBMS that has quasi-optimal performance and only needs L and k . We showed how preprocessing with MBMS improves the quality of algorithms for manifold learning and classification that are sensitive to noise or outliers, and expect this would apply to other settings with noisy data of intrinsic low dimensionality, such as density estimation, regression or semi-supervised learning.

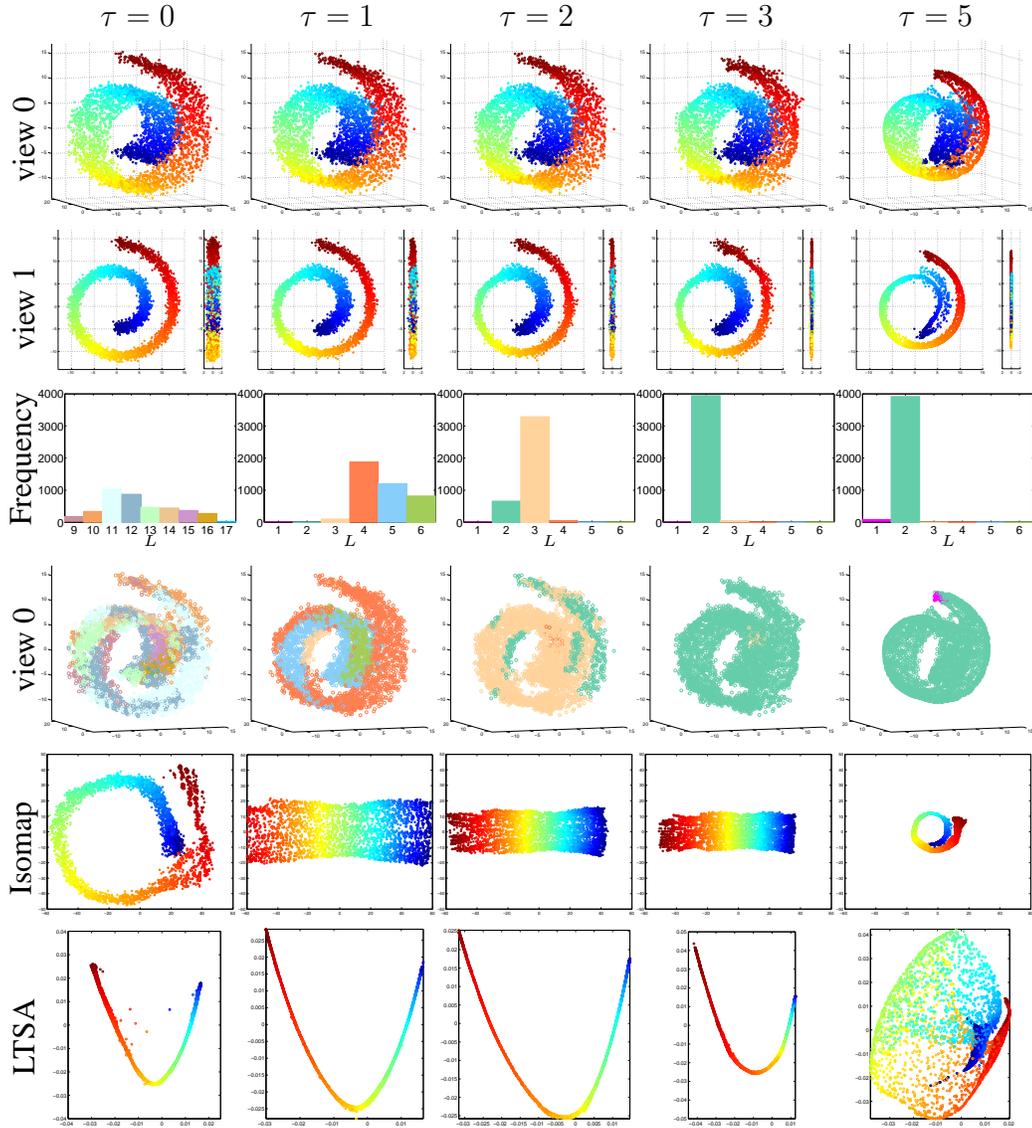


Figure 2.12: Dimensionality reduction with Isomap and LTSA for different iterations of MBMSk denoising ($k = 50, \sigma = 3$) on the 100D swissroll dataset. Intrinsic dimensionality at each data point is estimated by MLE at each iteration. Isomap/LTSA used a 10-nn graph. We show the colored histogram (frequency) of estimated dimensionalities L on the dataset over iterations in row 3 and the denoised datasets colored according to their estimated L in row 4.

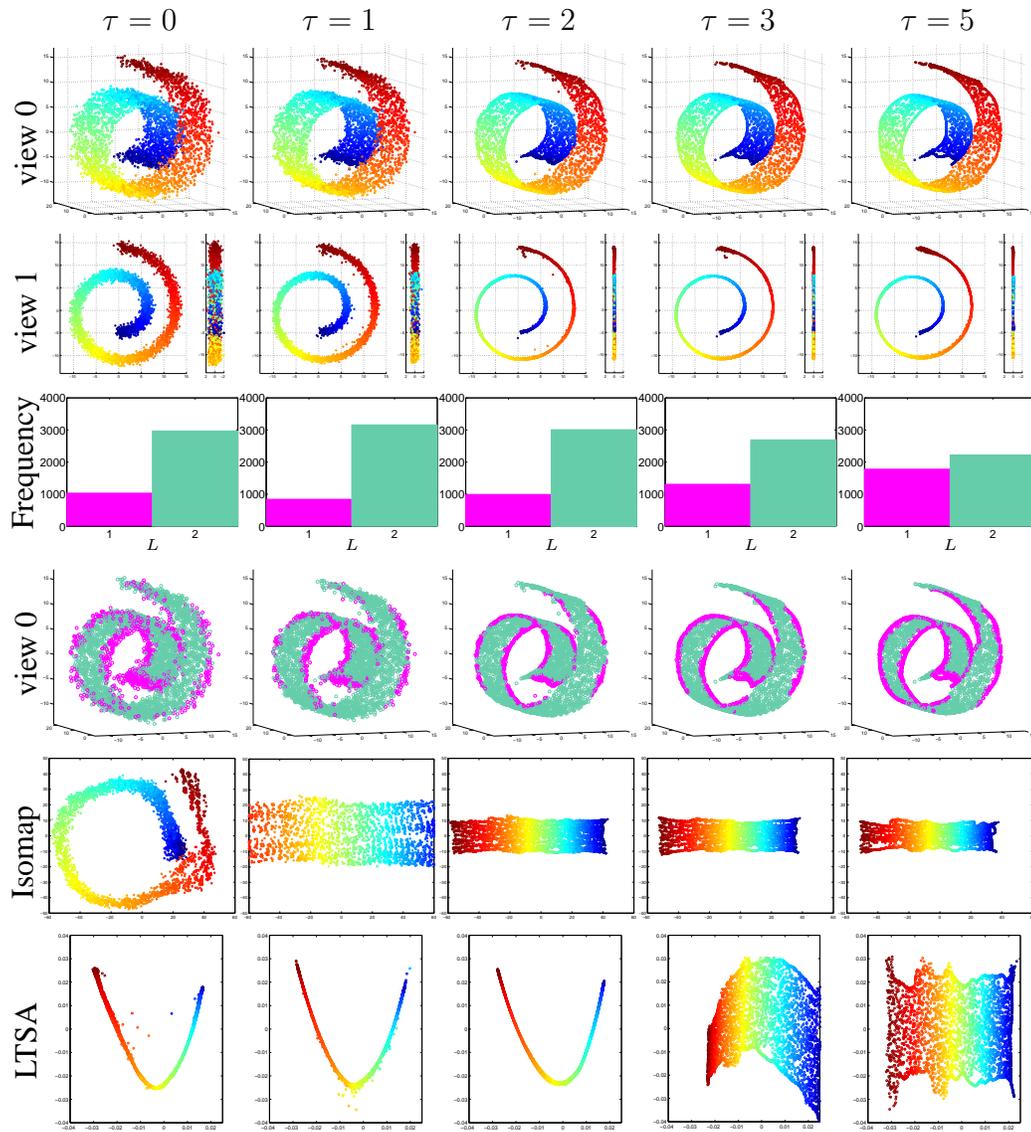


Figure 2.13: Dimensionality reduction with Isomap and LTSA for different iterations of MBMSk denoising ($k = 50$, $\sigma = 3$) on the 100D swissroll dataset. Intrinsic dimensionality at each data point is estimated by EIG at each iteration. Isomap/LTSA used a 10-nn graph. We show the colored histogram (frequency) of estimated dimensionalities L on the dataset over iterations in row 3 and the denoised datasets colored according to their estimated L in row 4.

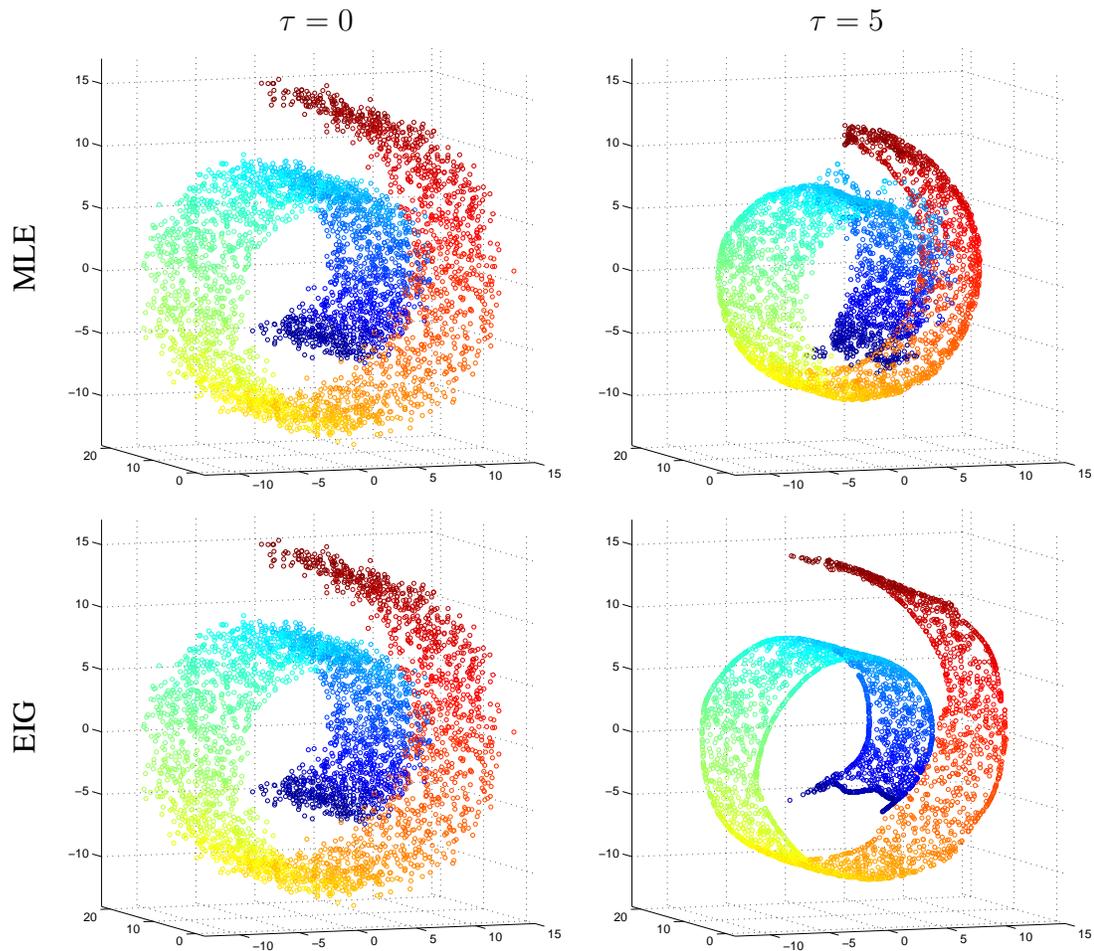


Figure 2.14: Boundary effect of MBMSk denoising ($k = 50$, $\sigma = 3$) on the 100D swiss-roll dataset, with MLE (first row) and EIG (second row) estimated intrinsic dimensionalities. MLE and particularly EIG have a noticeable effect on the manifold boundary, where points accumulate and produce a 1D alignment. We show view 0 (dimensions 1–3) of the dataset before and after denoising.

Chapter 3

A denoising view of matrix completion

In matrix completion, we are given a matrix where the values of only some of the entries are present, and we want to reconstruct the missing ones. Much work has focused on the assumption that the data matrix has low rank. We propose a more general assumption based on denoising, so that we expect that the value of a missing entry can be predicted from the values of neighboring points. We propose a nonparametric version of denoising using the MBMS algorithm developed in Chapter 2, which is based on local, iterated averaging with mean-shift, possibly constrained to preserve local low-rank manifold structure. The few user parameters required (the denoising scale, number of neighbors and local dimensionality) and the number of iterations can be estimated by cross-validating the reconstruction error. Using our algorithms as a postprocessing step on an initial reconstruction (provided by e.g. a low-rank method), we show consistent improvements with synthetic, image and motion-capture data (Wang et al., 2011).

3.1 Introduction

Completing a matrix from a few given entries is a fundamental problem with many applications in machine learning, computer vision, network engineering, and data mining. Much interest in matrix completion has been caused by recent theoretical breakthroughs

in compressed sensing (Candès and Recht, 2009; Candès and Tao, 2010) as well as by the now celebrated Netflix challenge on practical prediction problems (Bell and Koren, 2007; Koren, 2008). Since completion of arbitrary matrices is not a well-posed problem, it is often assumed that the underlying matrix comes from a restricted class. Matrix completion models almost always assume a low-rank structure of the matrix, which is partially justified through factor models (Bell and Koren, 2007) and fast convex relaxation (Candès and Tao, 2010), and often works quite well when the observations are sparse and/or noisy. The low-rank structure of the matrix essentially asserts that all the column vectors (or the row vectors) live on a low-dimensional subspace. This assumption is arguably too restrictive for problems with richer structure, e.g. when each column of the matrix represents a snapshot of a seriously corrupted motion capture sequence (see section 3.4), for which a more flexible model, namely a curved manifold, is more appropriate.

In this chapter, we present a novel view of matrix completion based on manifold denoising, which conceptually generalizes the low-rank assumption to curved manifolds. Traditional manifold denoising is performed on fully observed data (Hein and Maier, 2007), aiming to send the data corrupted by noise back to the correct surface (defined in some way). However, with a large proportion of missing entries, we may not have a good estimate of the manifold. Instead, we start with a poor estimate and improve it iteratively. Therefore the “noise” may be due not just to intrinsic noise, but mostly to inaccurately estimated missing entries. We show that our algorithm can be motivated from an objective purely based on denoising, and prove its convergence under some conditions. We then consider a more general case with a nonlinear low-dimensional manifold and use a stopping criterion that works successfully in practice. Our model reduces to a low-rank model when we require the manifold to be flat, showing a relation with a recent thread of matrix completion models (Jain et al., 2010). In our experiments, we show that our denoising-based matrix completion model can make better use of the latent manifold structure on both artificial and real-world datasets, and yields superior recovery of the missing entries.

Notation In this chapter, we use the subindex notation $\mathbf{X}_{\mathcal{M}}$ and $\mathbf{X}_{\mathcal{P}}$ to indicate selection of the missing or present values of the matrix $\mathbf{X}_{D \times N}$, where $\mathcal{P} \subset \mathcal{U}$, $\mathcal{M} = \mathcal{U} \setminus \mathcal{P}$ and $\mathcal{U} = \{(d, n): d = 1, \dots, D, n = 1, \dots, N\}$. The indices \mathcal{P} and values $\overline{\mathbf{X}}_{\mathcal{P}}$ of the present matrix entries are the data of the problem.

3.2 A brief review of related work

Matrix completion is widely studied in theoretical compressed sensing community (Candès and Recht, 2009; Candès and Tao, 2010). The minimum rank matrix completion problem is formulated as

$$\min_{\mathbf{X}} \text{rank}(\mathbf{X})_* \quad \text{s.t.} \quad \mathbf{X}_{\mathcal{P}} = \overline{\mathbf{X}}_{\mathcal{P}}, \quad (3.1)$$

which is known to be NP-hard and difficult to solve in both theory and practice (Chistov and Grigoriev, 1984; Meka et al., 2008). Candès and Recht (2009) propose to solve the matrix completion problem using the following convex objective function

$$\min_{\mathbf{X}} \|\mathbf{X}\|_* \quad \text{s.t.} \quad \mathbf{X}_{\mathcal{P}} = \overline{\mathbf{X}}_{\mathcal{P}}, \quad (3.2)$$

where $\|\mathbf{X}\|_*$ (called *nuclear norm*) is the sum of singular values of \mathbf{X} , and used as convex surrogate for minimizing the rank of \mathbf{X} . The authors show that for matrices that satisfy certain conditions and have small number of present entries sampled randomly, the unique solution of problem (3.2) actually recovers the entire matrix exactly (in which cases, (3.1) and (3.2) becomes *equivalent*) with high probability. Notice that (3.2) can be transformed into a semidefinite program and solved with interior point method (Liu and Vandenberghe, 2009). In order to solve large instances of matrix completion, Cai et al. (2010) propose a dual subgradient ascent algorithm called *singular value thresholding* for (3.2), which iteratively shrinks the singular values of certain estimate towards zero, similar to the shrinkage operations used for ℓ_1 penalty optimization in compressed sensing (Beck and Teboulle, 2009).

Keshavan et al. (2010) propose to optimize the following objective function for completion of a rank- r matrix with decomposition $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$

$$\min_{\mathbf{U} \in \mathbb{R}^{D \times r}, \mathbf{V} \in \mathbb{R}^{N \times r}} F(\mathbf{U}, \mathbf{V}) \equiv \min_{\mathbf{S} \in \mathbb{R}^{r \times r}} \frac{1}{2} \sum_{(i,j) \in \mathcal{P}} (\bar{\mathbf{X}}_{ij} - (\mathbf{U}\mathbf{S}\mathbf{V}^T)_{ij})^2 \quad (3.3)$$

$$\mathbf{U}^T \mathbf{U} = D\mathbf{I}, \quad \mathbf{V}^T \mathbf{V} = N\mathbf{I}, \quad (3.4)$$

where the rank r and initial (\mathbf{U}, \mathbf{V}) are obtained through the SVD of the *trimmed* version of $\bar{\mathbf{X}}_{\mathcal{P}}$ (by setting to zero over-represented rows and columns). The key observation here is that function $F(\mathbf{U}, \mathbf{V})$ depends on the (scaled) orthonormal matrices (\mathbf{U}, \mathbf{V}) only through their column spaces, and rotations in the column spaces do not change its value. Thus the objective function can be considered as optimizing $F(\mathbf{U}, \mathbf{V})$ over the Cartesian product of two Grassmannian manifolds, and the authors apply gradient descent technique on this Riemannian manifold.

Jain et al. (2010) approach the matrix completion problem by solving the problem

$$\min_{\mathbf{X}} \frac{1}{2} \|\mathbf{X}_{\mathcal{P}} - \bar{\mathbf{X}}_{\mathcal{P}}\|_{\text{Fro}}^2 \quad \text{s.t.} \quad \text{rank}(\mathbf{X}) \leq r, \quad (3.5)$$

and design an efficient gradient projection algorithm for it where each iteration involves projecting a certain estimate to the (non-convex) set of rank r matrix using singular value decomposition. In a related problem setting, Ji and Ye (2009) propose to use the gradient projection algorithm and Nesterov's accelerated scheme for unconstrained, trace-norm regularized learning tasks.

Matrix completion is also well studied in practical recommender systems, where it is commonly believed that there are only a few latent factors that contribute to the user ratings (Koren, 2008). Assuming a low-rank factorization model $\mathbf{X} = \mathbf{L}\mathbf{R}^T$ with $\mathbb{R}^{D \times r}$ and $\mathbf{R} \in \mathbb{R}^{N \times r}$ ($r < \min(D, N)$), a straightforward objective function would be

$$\min_{\mathbf{L}, \mathbf{R}} \sum_{(i,j) \in \mathcal{P}} (\bar{\mathbf{X}}_{ij} - \mathbf{L}_i \mathbf{R}_j^T)^2 + \lambda (\|\mathbf{L}\|_{\text{Fro}}^2 + \|\mathbf{R}\|_{\text{Fro}}^2), \quad (3.6)$$

where \mathbf{L}_i denotes the i -th row of \mathbf{L} (and \mathbf{R}_j likewise), which can be solved with *al-*

ternating least squares algorithm. Optimization of this objective function can also be done without alternation and higher order method, e.g. the damped Newton algorithm (Buchanan and Fitzgibbon, 2005). Different matrix norms and error measure (e.g. hinge loss for discrete observations) have also been used by Srebro et al. (2005). A probabilistic extension of this matrix factorization approach has been proposed by Salakhutdinov and Mnih (2008a) and later a Bayesian approach by Salakhutdinov and Mnih (2008b). An online version of this matrix factorization model is developed to handle one column of \mathbf{X} at a time by Balzano et al. (2010) for matrix completion and more general subspace tracking, where stochastic gradient descent technique on the Grassmannian manifold is used for optimization.

As we can see, most matrix completion models rely on a low-rank assumption, and cannot fully exploit a more complex structure of the problem, such as curved manifolds. Related work is on multi-task learning in a broad sense, which extracts the common structure shared by multiple related objects and achieves simultaneous learning on them. This includes applications such as alignment of noise-corrupted images (Peng et al., 2010), recovery of images with occlusion (Buchanan and Fitzgibbon, 2005), and even learning of multiple related regressors or classifiers (Argyriou et al., 2007). Again, all these works are essentially based on a subspace assumption, and do not generalize to more complex situations.

A line of work based on a nonlinear low-rank assumption (with a latent variable \mathbf{z} of dimensionality $L < D$) involves setting up a least-squares error function

$$\min_{\mathbf{f}, \mathbf{Z}} \sum_{n=1}^N \|\mathbf{x}_n - \mathbf{f}(\mathbf{z}_n)\|^2 = \sum_{(d,n) \in \mathcal{P}} (\bar{\mathbf{X}}_{dn} - f_d(\mathbf{z}_n))^2, \quad (3.7)$$

where one ignores the terms for which x_{dn} is missing, and estimates the function \mathbf{f} and the low-dimensional data projections \mathbf{Z} by alternating optimization. Linear functions \mathbf{f} have been used in the homogeneity analysis literature (Gifi, 1990), where this approach is called “missing data deleted” and is equivalent to (3.6). Nonlinear functions \mathbf{f} have been used recently (neural nets (Scholz et al., 2005); Gaussian processes for collaborative filtering (Lawrence and Urtasun, 2009)). Better results are obtained if adding

a projection term $\sum_{n=1}^N \|\mathbf{z}_n - \mathbf{F}(\mathbf{x}_n)\|^2$ and optimizing over the missing data as well (Carreira-Perpiñán and Lu, 2011).

There is also rich literature on the problem of missing values/incomplete data in the research field of statistics. One principled way of modeling incomplete data is based on mixture modeling where the missing values can be considered as “hidden” variables, similar to those indicator variables that already exist in mixture models. By marginalizing out all hidden variables and maximizing the likelihood of observed data, parameters of the mixture models can be learnt by the *Expectation Maximization* algorithm (Dempster et al., 1977; Ghahramani and Jordan, 1994). For a detailed discussion of this approach and its relationship with various other statistical approaches, we refer the readers to Ghahramani and Jordan (1994).

Prior to our denoising-based work there have been efforts to extend the low-rank models to smooth manifolds, mostly in the context of compressed sensing. Baraniuk and Wakin (2009) show that certain random measurements, e.g. random projection to a low-dimensional subspace, can preserve the metric of the manifold fairly well, if the intrinsic dimension and the curvature of the manifold are both small enough. However, these observations are not suitable for matrix completion and no algorithm is given for recovering the signal. Chen et al. (2010) explicitly model a manifold with a nonparametric mixture of factor analyzers where each mixing component has low-rank covariance structure (this model is analogous to the multiple subspaces model discussed below). The authors also propose an algorithm for recovering the signal from random linear measurements. Notice they estimate the manifold given complete data, while no complete data is assumed in our matrix completion setting.

Another relevant research area is subspace clustering (Vidal, 2011), where the dataset is assumed to lie on the union of multiple affine subspaces which may intersect with each other. This assumption generalizes the single subspace/low-rank assumption and has important applications in computer vision (e.g., motion segmentation, face clustering) and other areas. Under this assumption, Elhamifar and Vidal (2009) propose to model each data point as a sparse linear combination of the rest dataset, and show that by solving a sparse regression problem one can find neighborhood for each point from

the same subspace it is sampled from. The authors use the detected neighborhood of each point to build a graph on the dataset and use it as the input to spectral clustering algorithms to separate and recover the underlying subspaces. This approach is shown to be robust to noise and outliers (Soltanolkotabi et al., 2013) and is also extended to clustering of nonlinear manifolds (Elhamifar and Vidal, 2011).

3.3 Blurring mean-shift denoising algorithms for matrix completion

3.3.1 GBMS/MBMS revisited

In this section, we derive a objective function for the Gaussian blurring mean-shift (GBMS) algorithm, which will be used later for matrix completion. In GBMS, denoising is performed in a nonparametric way by local averaging: each data point moves to the average of its neighbors (to a certain scale), and the process is repeated. Consider a dataset $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \subset \mathbb{R}^D$ and define a Gaussian kernel density estimate (up to some normalization constant)

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N G\left(\left\|\frac{\mathbf{x} - \mathbf{x}_n}{\sigma}\right\|^2\right) \quad (3.8)$$

with bandwidth $\sigma > 0$ and kernel $G(t) = e^{-t/2}$ (other kernels may be used, such as the Epanechnikov kernel, which results in sparse affinities). The (non-blurring) *mean-shift algorithm* rearranges the stationary point equation $\nabla p(\mathbf{x}) = \mathbf{0}$ into the iterative scheme $\mathbf{x}^{(\tau+1)} = \mathbf{f}(\mathbf{x}^{(\tau)})$ with

$$p(n|\mathbf{x}^{(\tau)}) = \frac{G(\|(\mathbf{x}^{(\tau)} - \mathbf{x}_n)/\sigma\|^2)}{\sum_{n'=1}^N G(\|(\mathbf{x}^{(\tau)} - \mathbf{x}_{n'})/\sigma\|^2)}, \quad (3.9a)$$

$$\mathbf{x}^{(\tau+1)} = \mathbf{f}(\mathbf{x}^{(\tau)}) = \sum_{n=1}^N p(n|\mathbf{x}^{(\tau)})\mathbf{x}_n. \quad (3.9b)$$

This converges to a mode of p from almost every initial $\mathbf{x} \in \mathbb{R}^D$, and can be seen as *taking self-adapting step sizes along the gradient* (since the mean shift vector $\mathbf{f}(\mathbf{x}) - \mathbf{x}$ is parallel to $\nabla p(\mathbf{x})$).

The *blurring mean-shift algorithm* applies one step of the previous scheme, initialized from every point, in parallel for all points. That is, given the dataset $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, for each $\mathbf{x}_n \in \mathbf{X}$ we obtain a new point $\tilde{\mathbf{x}}_n = \mathbf{f}(\mathbf{x}_n)$ by applying one step of the mean-shift algorithm, and then we replace \mathbf{X} with the new dataset $\tilde{\mathbf{X}}$, which is a blurred (shrunk) version of \mathbf{X} . By iterating this process we obtain a sequence of datasets $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \dots$ (and a corresponding sequence of kernel density estimates $p^{(0)}(\mathbf{x}), p^{(1)}(\mathbf{x}), \dots$) where $\mathbf{X}^{(0)}$ is the original dataset and $\mathbf{X}^{(\tau)}$ is obtained by blurring $\mathbf{X}^{(\tau-1)}$ with one mean-shift step. We can see this process as maximizing the following objective function (Cheng, 1995) by taking parallel steps of the form (3.9) for each point:

$$E(\mathbf{X}) = \sum_{n=1}^N p(\mathbf{x}_n) = \frac{1}{N} \sum_{n,m=1}^N G(\|(\mathbf{x}_n - \mathbf{x}_m)/\sigma\|^2) \propto \sum_{n,m=1}^N e^{-\frac{1}{2}\|\frac{\mathbf{x}_n - \mathbf{x}_m}{\sigma}\|^2}. \quad (3.10)$$

This process eventually converges to a dataset $\mathbf{X}^{(\infty)}$ where all points are coincident: a completely denoised dataset where all structure has been erased. As shown by Carreira-Perpiñán (2006b), this process can be stopped early to return clusters (= locally denoised subsets of points); the number of clusters obtained is controlled by the bandwidth σ . However, here we are interested in the denoising behavior of GBMS.

The GBMS step can be formulated in a matrix form reminiscent of spectral clustering (Carreira-Perpiñán, 2006b) as $\tilde{\mathbf{X}} = \mathbf{X} \mathbf{P}$ where $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ is a $D \times N$ matrix of data points; \mathbf{W} is the $N \times N$ matrix of Gaussian affinities $w_{nm} = G(\|(\mathbf{x}_n - \mathbf{x}_m)/\sigma\|^2)$; $\mathbf{D} = \text{diag}(\sum_{n=1}^N w_{nm})$ is the degree matrix; and $\mathbf{P} = \mathbf{W} \mathbf{D}^{-1}$ is an $N \times N$ stochastic matrix: $p_{nm} = p(n|\mathbf{x}_m) \in (0, 1)$ and $\sum_{n=1}^N p_{nm} = 1$. \mathbf{P} (or rather its transpose) is the stochastic matrix of the random walk in a graph (Chung, 1997), which in GBMS represents the posterior probabilities of each point under the kernel density estimate (1.1). \mathbf{P} is similar to the matrix $\mathbf{N} = \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}$ derived from the normalized graph Laplacian commonly used in spectral clustering, e.g. in the normalized cut (Shi and Malik, 2000). Since, by the Perron-Frobenius theorem (Horn and Johnson, 1986, Chapter 8), all left

eigenvalues of $\mathbf{P}(\mathbf{X})$ have magnitude less than 1 except for one that equals 1 and is associated with an eigenvector of constant entries, iterating $\tilde{\mathbf{X}} = \mathbf{X} \mathbf{P}(\mathbf{X})$ converges to the stationary distribution of each $\mathbf{P}(\mathbf{X})$, where all points coincide.

From this point of view, the product $\tilde{\mathbf{X}} = \mathbf{X} \mathbf{P}(\mathbf{X})$ can be seen as filtering the dataset \mathbf{X} with a data-dependent low-pass filter $\mathbf{P}(\mathbf{X})$, which makes clear the denoising behavior. This also suggests using other filters (Carreira-Perpiñán, 2008) $\tilde{\mathbf{X}} = \mathbf{X} \phi(\mathbf{P}(\mathbf{X}))$ as long as $\phi(1) = 1$ and $|\phi(r)| < 1$ for $r \in [0, 1)$, such as explicit schemes $\phi(\mathbf{P}) = (1 - \eta)\mathbf{I} + \eta\mathbf{P}$ for $\eta \in (0, 2]$, power schemes $\phi(\mathbf{P}) = \mathbf{P}^n$ for $n = 1, 2, 3 \dots$ or implicit schemes $\phi(\mathbf{P}) = ((1 + \eta)\mathbf{I} - \eta\mathbf{P})^{-1}$ for $\eta > 0$.

One important problem with GBMS is that it denoises equally in all directions. When the data lies on a low-dimensional manifold, denoising orthogonally to it removes out-of-manifold noise, but denoising tangentially to it perturbs intrinsic degrees of freedom of the data and causes shrinkage of the entire manifold (most strongly near its boundary). To prevent this, the *manifold blurring mean-shift algorithm (MBMS)* (Chapter 2) first computes a predictor averaging step with GBMS, and then for each point \mathbf{x}_n a corrector projective step removes the step direction that lies in the local tangent space of \mathbf{x}_n (obtained from local PCA run on its k nearest neighbors). In practice, both GBMS and MBMS must be stopped early to prevent excessive denoising and manifold distortions.

3.3.2 GBMS/MBMS for matrix completion

We consider the natural extension of GBMS to the matrix completion case by adding the constraints given by the present values. Then we have the following constrained optimization problem:

$$\max_{\mathbf{X}} E(\mathbf{X}) = \sum_{n,m=1}^N G(\|(\mathbf{x}_n - \mathbf{x}_m)/\sigma\|^2) \quad \text{s.t.} \quad \mathbf{X}_{\mathcal{P}} = \bar{\mathbf{X}}_{\mathcal{P}}. \quad (3.11)$$

This is similar to low-rank formulation (3.6) for matrix completion that have the same constraints but use as objective function the reconstruction error with a low-rank assumption.

We initialize $\mathbf{X}_{\mathcal{M}}$ to the output of some other method for matrix completion, such as singular value projection (SVP; Jain et al., 2010). For simple constraints such as ours, gradient projection algorithms are attractive. The gradient of E wrt \mathbf{X} is a matrix of $D \times N$ whose n th column is:

$$\begin{aligned} \nabla_{\mathbf{x}_n} E(\mathbf{X}) &= \frac{2}{\sigma^2} \sum_{m=1}^N e^{-\frac{1}{2} \left\| \frac{\mathbf{x}_n - \mathbf{x}_m}{\sigma} \right\|^2} (\mathbf{x}_m - \mathbf{x}_n) \\ &\propto \frac{2}{\sigma^2} p(\mathbf{x}_n) \left(-\mathbf{x}_n + \sum_{m=1}^N p(m|\mathbf{x}_n) \mathbf{x}_m \right) \end{aligned} \quad (3.12)$$

and its projection on the constraint space is given by zeroing its entries having indices in \mathcal{P} ; call $\Pi_{\mathcal{P}}$ this projection operator. Then, we have the following step of length $\alpha \geq 0$ along the projected gradient:

$$\begin{aligned} \mathbf{X}^{(\tau+1)} &= \mathbf{X}^{(\tau)} + \alpha \Pi_{\mathcal{P}}(\nabla_{\mathbf{X}} E(\mathbf{X}^{(\tau)})) \\ \iff \mathbf{X}_{\mathcal{M}}^{(\tau+1)} &= \mathbf{X}_{\mathcal{M}}^{(\tau)} + \alpha (\Pi_{\mathcal{P}}(\nabla_{\mathbf{X}} E(\mathbf{X}^{(\tau)})))_{\mathcal{M}} \end{aligned} \quad (3.13)$$

which updates only the missing entries $\mathbf{X}_{\mathcal{M}}$. Since our search direction is ascent and makes an angle with the gradient that is bounded away from $\pi/2$, and E is upper bounded, continuously differentiable and has bounded Hessian (thus a Lipschitz continuous gradient) in \mathbb{R}^{NL} , by carrying out a line search that satisfies the Wolfe conditions, we are guaranteed convergence to a local stationary point, typically a maximizer (Nocedal and Wright, 2006, Theorem 3.2). However, as reasoned later, we do not perform a line search at all, instead we fix the step size to the GBMS self-adapting step size, which results in a simple and faster algorithm consisting of carrying out a GBMS step on \mathbf{X} (i.e., $\mathbf{X}^{(\tau+1)} = \mathbf{X}^{(\tau)} \mathbf{P}(\mathbf{X}^{(\tau)})$) and then refilling $\mathbf{X}_{\mathcal{P}}$ to the present values. While we describe the algorithm in this way for ease of explanation, in practice we do not actually compute the GBMS step for all x_{dn} values, but only for the missing ones, which is all we need. Thus, our algorithm carries out GBMS denoising steps *within the missing-data subspace*. We can derive this result in a different way by starting from the unconstrained optimization problem $\max_{\mathbf{X}_{\mathcal{P}}} E(\mathbf{X}) = \sum_{n,m=1}^N G(\|(\mathbf{x}_n - \mathbf{x}_m)/\sigma\|^2)$ (equivalent to (3.11)), computing its gradient wrt $\mathbf{X}_{\mathcal{P}}$, equating it to zero and rearranging

(in the same way the mean-shift algorithm is derived) to obtain a fixed-point iteration identical to our update above.

Figure 3.1 shows the pseudocode for our denoising-based matrix completion algorithms (using three nonparametric denoising algorithms: GBMS, MBMS and LTP).

Convergence and stopping criterion As noted above, we have guaranteed convergence by simply satisfying standard line search conditions, but a line search is costly. At present we do not have a proof that the GBMS step size satisfies such conditions, or indeed that the new iterate $\mathbf{X}_{\mathcal{M}}^{(\tau+1)}$ increases or leaves unchanged the objective, although we have never encountered a counterexample. In fact, it turns out that none of the work about GBMS that we know about proves that either: Cheng (1995) proves that $\varnothing(\mathbf{X}^{(\tau+1)}) \leq \varnothing(\mathbf{X}^{(\tau)})$ for $0 < \rho < 1$, where $\varnothing(\cdot)$ is the set diameter, while Carreira-Perpiñán (2006b, 2008) notes that $\mathbf{P}(\mathbf{X})$ has a single eigenvalue of value 1 and all others of magnitude less than 1. While this shows that all points converge to the same location, which indeed is the global maximum of (3.10), it does not necessarily follow that each step decreases E .

However, the question of convergence as $\tau \rightarrow \infty$ has no practical interest in a denoising setting, because achieving a total denoising almost never yields a good matrix completion. What we want is to achieve *just enough* denoising and stop the algorithm, as was the case with GBMS clustering, and as is the case in algorithms for image denoising. We propose to determine the optimal number of iterations, as well as the bandwidth σ and any other parameters, by cross-validation. Specifically, we select a held-out set by picking a random subset of the present entries and considering them as missing; this allows us to evaluate an error between our completion for them and the ground truth. We stop iterating when this error increases.

This argument justifies an algorithmic, as opposed to an optimization, view of denoising-based matrix completion: *apply a denoising step, refill the present values, iterate until the validation error increases*. This allows very general definitions of denoising, and indeed a low-rank projection is a form of denoising where points are not allowed outside

<p>GBMS (k, σ) with full or k-nn graph: given $\mathbf{X}_{D \times N}, \mathcal{M}$</p> <p>repeat</p> <p> for $n = 1, \dots, N$</p> <p> $\mathcal{N}_n \leftarrow \{1, \dots, N\}$ (full graph) or k nearest neighbors of \mathbf{x}_n (k-nn graph)</p> <p> $\partial \mathbf{x}_n \leftarrow -\mathbf{x}_n + \sum_{m \in \mathcal{N}_n} \frac{G(\ \mathbf{x}_n - \mathbf{x}_m\ /\sigma)^2}{\sum_{m' \in \mathcal{N}_n} G(\ \mathbf{x}_n - \mathbf{x}_{m'}\ /\sigma)^2} \mathbf{x}_m$ mean-shift step</p> <p> end</p> <p> $\mathbf{X}_{\mathcal{M}} \leftarrow \mathbf{X}_{\mathcal{M}} + (\partial \mathbf{X})_{\mathcal{M}}$ move points' missing entries</p> <p>until validation error increases</p> <p>return \mathbf{X}</p>	
<p>MBMS (L, k, σ) with full or k-nn graph: given $\mathbf{X}_{D \times N}, \mathcal{M}$</p> <p>repeat</p> <p> for $n = 1, \dots, N$</p> <p> $\mathcal{N}_n \leftarrow \{1, \dots, N\}$ (full graph) or k nearest neighbors of \mathbf{x}_n (k-nn graph)</p> <p> $\partial \mathbf{x}_n \leftarrow -\mathbf{x}_n + \sum_{m \in \mathcal{N}_n} \frac{G(\ \mathbf{x}_n - \mathbf{x}_m\ /\sigma)^2}{\sum_{m' \in \mathcal{N}_n} G(\ \mathbf{x}_n - \mathbf{x}_{m'}\ /\sigma)^2} \mathbf{x}_m$ mean-shift step</p> <p> $\mathcal{X}_n \leftarrow k$ nearest neighbors of \mathbf{x}_n</p> <p> $(\boldsymbol{\mu}_n, \mathbf{U}_n) \leftarrow \text{PCA}(\mathcal{X}_n, L)$ estimate L-dim tangent space at \mathbf{x}_n</p> <p> $\partial \mathbf{x}_n \leftarrow (\mathbf{I} - \mathbf{U}_n \mathbf{U}_n^T) \partial \mathbf{x}_n$ subtract parallel motion</p> <p> end</p> <p> $\mathbf{X}_{\mathcal{M}} \leftarrow \mathbf{X}_{\mathcal{M}} + (\partial \mathbf{X})_{\mathcal{M}}$ move points' missing entries</p> <p>until validation error increases</p> <p>return \mathbf{X}</p>	
<p>LTP (L, k) with k-nn graph: given $\mathbf{X}_{D \times N}, \mathcal{M}$</p> <p>repeat</p> <p> for $n = 1, \dots, N$</p> <p> $\mathcal{X}_n \leftarrow k$ nearest neighbors of \mathbf{x}_n</p> <p> $(\boldsymbol{\mu}_n, \mathbf{U}_n) \leftarrow \text{PCA}(\mathcal{X}_n, L)$ estimate L-dim tangent space at \mathbf{x}_n</p> <p> $\partial \mathbf{x}_n \leftarrow (\mathbf{I} - \mathbf{U}_n \mathbf{U}_n^T)(\boldsymbol{\mu}_n - \mathbf{x}_n)$ project point onto tangent space</p> <p> end</p> <p> $\mathbf{X}_{\mathcal{M}} \leftarrow \mathbf{X}_{\mathcal{M}} + (\partial \mathbf{X})_{\mathcal{M}}$ move points' missing entries</p> <p>until validation error increases</p> <p>return \mathbf{X}</p>	

Figure 3.1: Our denoising matrix completion algorithms, based on Manifold Blurring Mean Shift (MBMS) and its particular cases Local Tangent Projection (LTP, k -nn graph, $\sigma = \infty$) and Gaussian Blurring Mean Shift (GBMS, $L = 0$); see Section 2.3 for details. \mathcal{N}_n contains all N points (full graph) or only \mathbf{x}_n 's nearest neighbors (k -nn graph). The index \mathcal{M} selects the components of its input corresponding to missing values. Parameters: denoising scale σ , number of neighbors k , local dimensionality L .

the linear manifold. Our formulation using the objective function (3.11) is still useful in that it connects our denoising assumption with the more usual low-rank assumption that has been used in much matrix completion work, and justifies the refilling step as resulting from the present-data constraints under a gradient-projection optimization.

MBMS denoising for matrix completion Following our algorithmic-based approach to denoising, we could consider generalized GBMS steps of the form $\tilde{\mathbf{X}} = \mathbf{X} \phi(\mathbf{P}(\mathbf{X}))$. For clustering, Carreira-Perpiñán Carreira-Perpiñán (2008) found an overrelaxed explicit step $\phi(\mathbf{P}) = (1 - \eta)\mathbf{I} + \eta\mathbf{P}$ with $\eta \approx 1.25$ to achieve similar clusterings but faster. Here, we focus instead on the MBMS variant of GBMS that allows only for orthogonal, not tangential, point motions (defined wrt their local tangent space as estimated by local PCA), with the goal of preserving low-dimensional manifold structure. MBMS has 3 user parameters: the bandwidth σ (for denoising), and the latent dimensionality L and the number of neighbors k (for the local tangent space and the neighborhood graph). A special case of MBMS called *local tangent projection (LTP)* results by using a neighborhood graph and setting $\sigma = \infty$ (so only two user parameters are needed: L and k). LTP can be seen as doing a low-rank matrix completion locally. LTP was found in Chapter 2 to have nearly as good performance as the best σ in several problems. MBMS also includes as particular cases GBMS ($L = 0$), PCA ($k = N, \sigma = \infty$), and no denoising ($\sigma = 0$ or $L = D$).

Note that if we apply MBMS to a dataset that lies on a linear manifold of dimensionality d using $L \geq d$ then no denoising occurs whatsoever because the GBMS updates lie on the d -dimensional manifold and are removed by the corrector step. In practice, even if the data are assumed noiseless, the reconstruction from a low-rank method will lie close to but not exactly on the d -dimensional manifold. However, this suggests using largish ranks for the low-rank method used to reconstruct \mathbf{X} and lower L values in the subsequent MBMS run.

In summary, this yields a matrix completion algorithm where we apply an MBMS step, refill the present values, and iterate until the validation error increases. Again, in an actual implementation we compute the MBMS step only for the missing entries of \mathbf{X} .

The shrinking problem of GBMS is less pronounced in our matrix completion setting, because we constrain some values not to change. Still, in agreement with Chapter 2, we find MBMS to be generally superior to GBMS.

A special case of our algorithm is directly related to low-rank matrix completion algorithms. If we take $k = N$ neighbors and $\sigma = \infty$ then MBMS becomes PCA in L dimensions, and our algorithm iterates between projecting \mathbf{X} onto the PCA subspace it defines (equivalent to the SVD of \mathbf{X} if it has zero mean) and resetting the present entries. This is a method of alternating projections (Lewis and Malick, 2008), similar to previous SVD-based work such as SVP (Jain et al., 2010), and to the linear version of the method of (Carreira-Perpiñán and Lu, 2011). Finally, our expectation that the value of a missing entry can be predicted from the values of neighboring points is similar to one category of collaborative filtering methods that essentially use similar users/items to predict missing values (Bell and Koren, 2007).

Computational cost With a full graph, the cost per iteration of GBMS and MBMS is $\mathcal{O}(N^2D)$ and $\mathcal{O}(N^2D + N(D + k) \min(D, k)^2)$, respectively. In practice with high-dimensional data, best denoising results are obtained using a neighborhood graph (see Section 2.3), so that the sums over points in eqs. (3.10) or (3.11) extend only to the neighbors. With a k -nearest-neighbor graph and if we do not update the neighbors at each iteration (which affects the result little), the respective cost per iteration is $\mathcal{O}(NkD)$ and $\mathcal{O}(NkD + N(D + k) \min(D, k)^2)$, thus linear in N . The graph is constructed on the initial \mathbf{X} we use, consisting of the present values and an imputation for the missing ones achieved with a standard matrix completion method, and has a one-off cost of $\mathcal{O}(N^2D)$. The cost when we have a fraction $\mu = \frac{|\mathcal{M}|}{ND} \in [0, 1]$ of missing data is simply the above times μ . Hence the run time of our mean-shift-based matrix completion algorithms is faster the more present data we have, and thus faster than the usual GBMS or MBMS case, where all data are effectively missing.

Table 3.1: Swissroll dataset: reconstruction errors obtained by different algorithms along with their optimal parameters (σ , k , L , no. iterations τ). The three columns show the root sum of squared errors on missing entries, the mean, and the standard deviation of the pointwise reconstruction error, resp.

Methods	RSSE	mean	stdev
Gaussian	168.1	2.63	1.59
+ GBMS (∞ , 10, 0, 1)	165.8	2.57	1.61
+ MBMS (1, 20, 2, 25)	157.2	2.36	1.63
SVP	156.8	1.94	2.10
+ GBMS (3, 50, 0, 1)	151.4	1.89	2.02
+ MBMS (3, 50, 2, 2)	151.8	1.87	2.05

3.4 Experimental results

We compare with representative methods of several approaches: a low-rank matrix completion method, singular value projection (SVP Jain et al., 2010, whose performance we found similar to that of alternating least squares, ALS (Koren, 2008)); fitting a D -dimensional Gaussian model with EM and imputing the missing values of each \mathbf{x}_n as the conditional mean $E\{\mathbf{x}_{n,\mathcal{M}_n}|\mathbf{x}_{n,\mathcal{P}_n}\}$ (we use the implementation of (Schneider, 2001)); and the nonlinear method of (Scholz et al., 2005) (nlPCA). We initialize GBMS and MBMS from some or all of these algorithms. For methods with user parameters, we set them by cross-validation in the following way: we randomly select 10% of the present entries and pretend they are missing as well, we run the algorithm on the remaining 90% of the present values, and we evaluate the reconstruction at the 10% entries we kept earlier. We repeat this over different parameters' values and pick the one with lowest reconstruction error. We then run the algorithm with these parameters values on the entire present data and report the (test) error with the ground truth for the missing values.

100D Swissroll We created a 3D swissroll dataset with 3 000 points and lifted it to 100D with a random orthonormal mapping, and added a little noise (spherical Gaussian with stdev 0.1). We selected uniformly at random 6.76% of the entries to be present. We use the Gaussian model and SVP (fixed rank = 3) as initialization for our algorithm. We typically find that these initial \mathbf{X} are very noisy (Figure 3.3), with some reconstructed

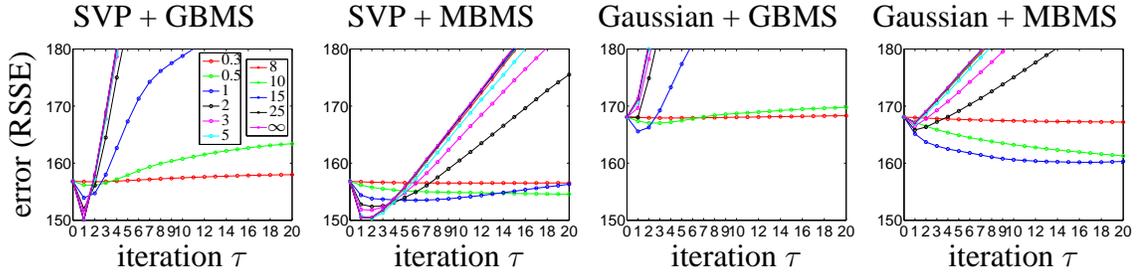


Figure 3.2: Reconstruction error of GBMS/MBMS over iterations on 100D swissroll (each curve is a different σ value).

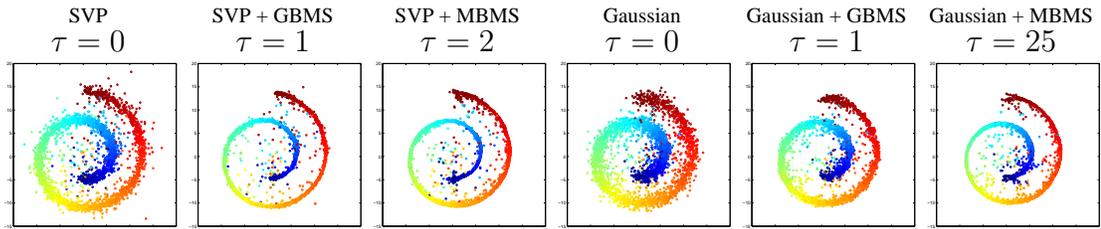


Figure 3.3: Denoising effect of the different algorithms on 100D swissroll. For visualization, we project the 100D data to 3D with the projection matrix used for creating the data. Present values are refilled for all plots.

points lying between different branches of the manifold and causing a big reconstruction error. We fixed $L = 2$ (the known dimensionality) for MBMS and cross-validated the other parameters: σ and k for MBMS and GBMS (both using k -nn graph), and the number of iterations τ to be used. Table 3.1 gives the performance of MBMS and GBMS for testing, along with their optimal parameters. Figure 3.3 shows the results of different methods at a few iterations. MBMS initialized from the Gaussian model gives the most remarkable denoising effect. To show that there is a wide range of σ and number of iterations τ that give good performance with GBMS and MBMS, we fix $k = 50$ and run the algorithm with varying σ values and plot the reconstruction error for missing entries over iterations in Figure 3.2. Both GBMS can achieve good denoising (and reconstruction), but MBMS is more robust, with good results occurring for a wide range of iterations, indicating it is able to preserve the manifold structure better.

Mocap data We use the running-motion sequence 09_01 from the CMU mocap database with 148 samples (≈ 1.7 cycles) with 150 sensor readings (3D positions of 50 joints on

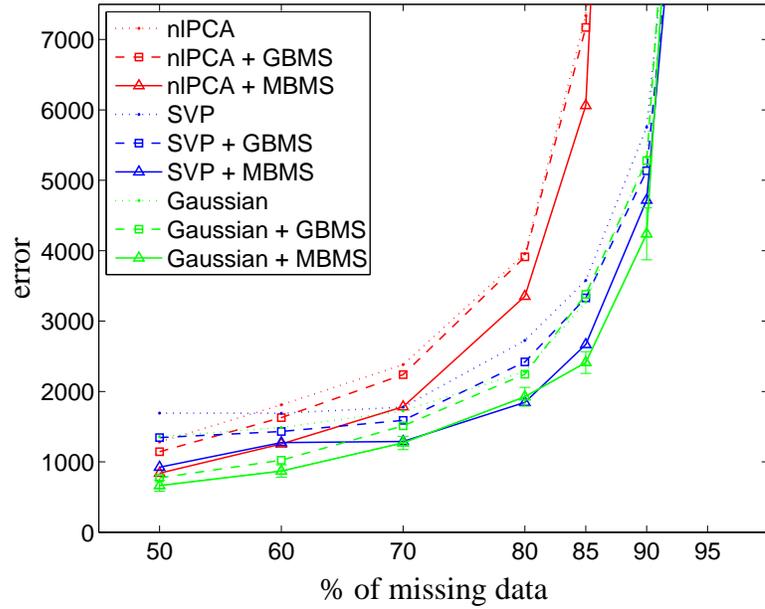


Figure 3.4: Results on Mocap dataset. Mean of errors (RSSE) of 5 runs obtained by different algorithms for varying percentage of missing values. Errorbars shown only for Gaussian + MBMS to avoid clutter.

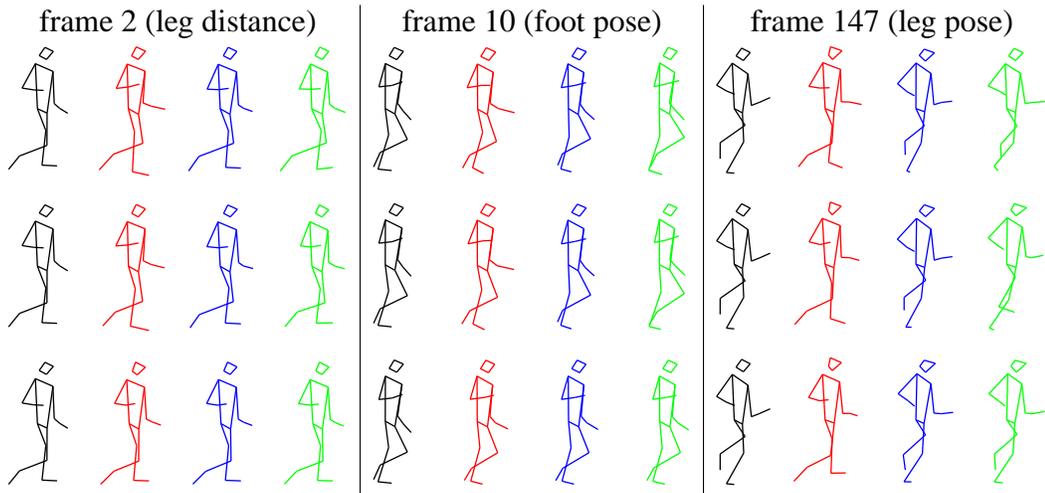


Figure 3.5: Results on Mocap dataset. Sample reconstructions when 85% percent data is missing. Row 1: initialization. Row 2: init+GBMS. Row 3: init+MBMS. Color indicates different initialization: black, original data; red, nIPCA; blue, SVP; green, Gaussian.

Table 3.2: MNIST-7 dataset: errors of the different algorithms and their optimal parameters (σ , k , L , no. iterations τ). The three columns show the root sum of squared errors on missing entries ($\times 10^{-4}$), the mean, and the standard deviation of pixel errors, respectively.

Methods	RSSE	mean	stdev
nLPCA	7.77	26.1	42.6
SVP	6.99	21.8	39.3
+ GBMS (400,140,0,1)	6.54	18.8	37.7
+ MBMS (500,140,9,5)	6.03	17.0	34.9

a human body). The motion is intrinsically 1D, tracing a loop in 150D. We compare nLPCA, SVP, the Gaussian model, and MBMS initialized from the first three algorithms. For nLPCA, we do a grid search for the weight decay coefficient while fixing its structure to be $2 \times 10 \times 150$ units, and use an early stopping criterion. For SVP, we do grid search on $\{1, 2, 3, 5, 7, 10\}$ for the rank. For MBMS ($L = 1$) and GBMS ($L = 0$), we do grid search for σ and k .

We report the reconstruction error as a function of the proportion of missing entries from 50% to 95%. For each missing-data proportion, we randomly select 5 different sets of present values and run all algorithms for them. Figure 3.4 gives the mean errors of all algorithms. All methods perform well when missing-data proportion is small. nLPCA, being prone to local optima, is less stable than SVP and the Gaussian model, especially when the missing-data proportion is large. The Gaussian model gives the best and most stable initialization. At 95%, all methods fail to give an acceptable reconstruction, but up to 90% missing entries, MBMS and GBMS always beat the other algorithms. Figure 3.5 shows selected reconstructions from all algorithms.

MNIST digit ‘7’ The MNIST digit ‘7’ dataset contains 6 265 greyscale (0–255) images of size 28×28 . We create missing entries in a way reminiscent of run-length errors in transmission. We generate 16 to 26 rectangular boxes of an area approximately 25 pixels at random locations in each image and use them to black out pixels. In this way, we create a high dimensional dataset (784 dimensions) with about 50% entries missing on average. Because of the loss of spatial correlations within the blocks, this missing data pattern is harder than random.

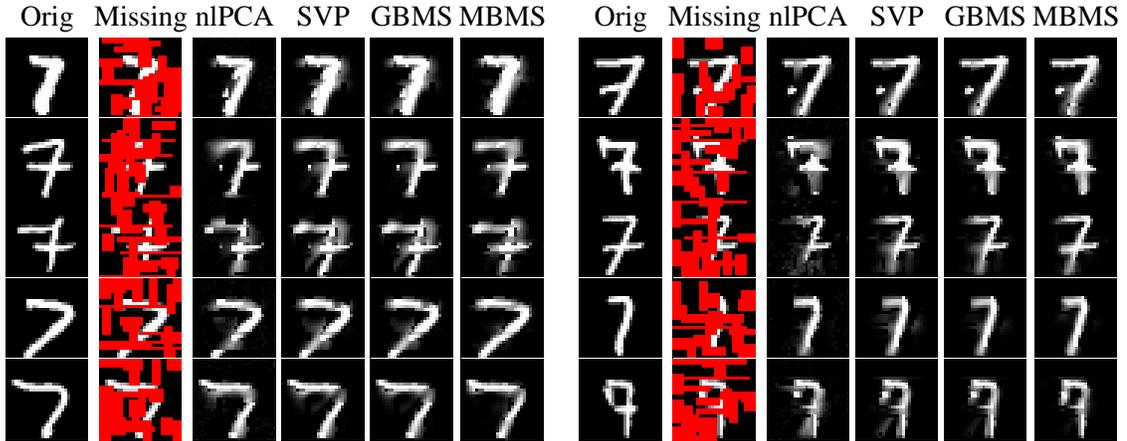


Figure 3.6: Selected reconstructions of MNIST block-occluded digits ‘7’ with different methods.

The Gaussian model cannot handle such a big dataset because it involves inverting large covariance matrices. nlPCA is also very slow and we cannot afford cross-validating its structure or the weight decay coefficient, so we picked a reasonable structure ($10 \times 30 \times 784$ units), used the default weight decay parameter in the code (10^{-3}), and allowed up to 500 iterations. We only use SVP as initialization for our algorithm. Since the intrinsic dimension of MNIST is suspected to be not very high, we used rank 10 for SVP and $L = 9$ for MBMS. We also use the same $k = 140$ as in Section 2.4. So we only had to choose σ and the number of iterations via cross-validation.

Table 3.2 shows the methods and their corresponding error. Figure 3.6 shows some representative reconstructions from different algorithms, with present values refilled. The mean-shift averaging among closeby neighbors (a soft form of majority voting) helps to eliminate noise, unusual strokes and other artifacts created by SVP, which by their nature tend to occur in different image locations over the neighborhood of images.

Random initialization In previous examples, we have mainly used the reconstructions from other models as the initializations for our algorithm. While these initializations do seem very noisy, they have basically captured the overall shape of the data manifold. A natural question to ask is whether MBMS is robust to unstructured initializations. We investigate this problem empirically on the 100D swissroll dataset, using

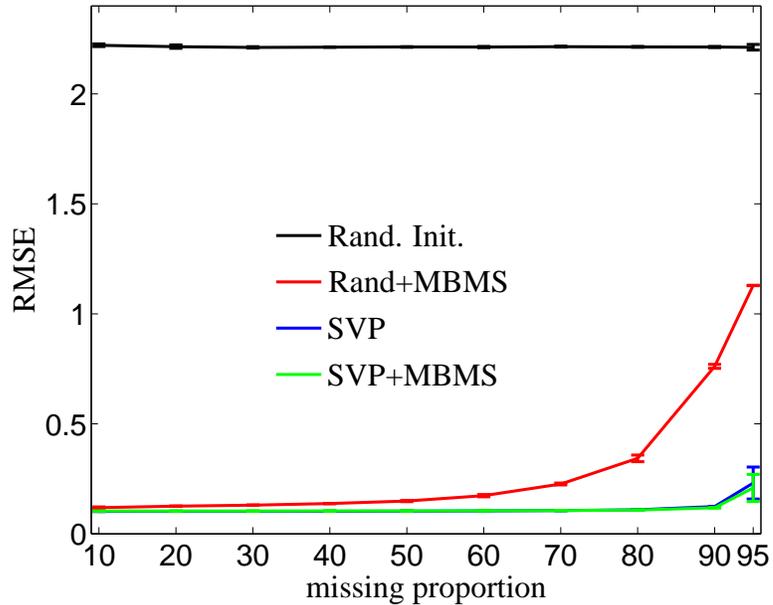


Figure 3.7: Root mean squared error (RMSE) per entry obtained by SVP and MBMS with different initializations on the 100D swissroll dataset at different missing proportions. We show the mean and standard deviation of RMSE averaged over 5 different missing/present partitions of the the data matrix.

random initial values for the missing entries. We now randomly select $p \cdot D$ dimensions for each sample to be missing, where p is the missing proportion ranging from 10% to 95%. Furthermore, we generate random values from the Gaussian distribution $\mathcal{N}(m, s)$ for missing entries where m and s are the empirical mean and standard deviation of the present entries respectively. We then apply MBMS to this initialization, and cross-validate σ for best reconstruction error on the missing entries by a grid search, while fixing $k = 100$ and $L = 2$. In Figure 3.7, we show the root mean squared error (RMSE) per entry of the random initialization, MBMS with random initialization, SVP (fixed rank=3), and MBMS with SVP initialization, each averaged over 5 different missing/present partitions of the the data matrix, at different missing proportions p . We also show the sample initialization and reconstructions at different missing proportions in Figure 3.8.

In general, the performance of MBMS with random initialization degrades as p increases. This to be expected since the initial pairwise distances and local neighbor-

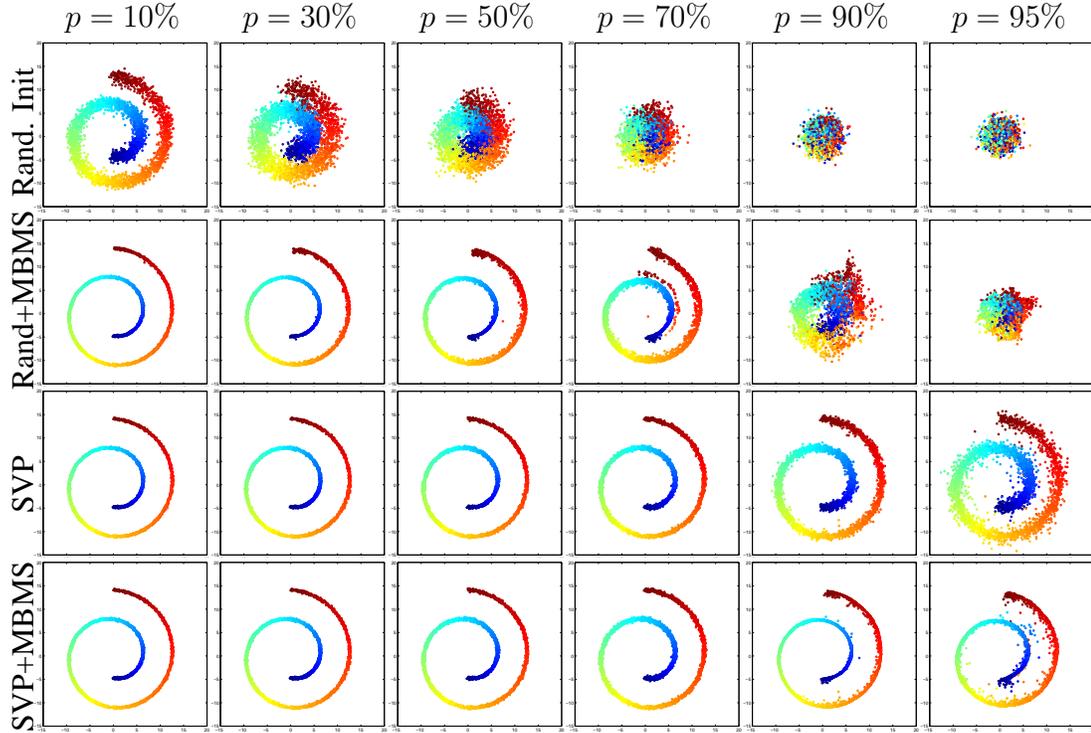


Figure 3.8: Sample reconstructions of the 100D swissroll dataset using MBMS with random initializations or SVP initialization for different missing proportions. For visualization, we project the 100D data to 3D with the projection matrix used for creating the data.

hoods also become more and more corrupted as p increases, and it is then more difficult to estimate the tangent space and manifold structure. This can also be seen from the sample random initialization (Figure 3.8, row 1) and MBMS reconstruction (row 2) at $p \geq 90\%$, where the basic shape of swissroll is completely lost. From an optimization point of view, initialization is important to our non-convex objective function and gradient-based approach. On the other hand, the low rank model SVP (row 3) has superior and more robust performance (the reconstruction error did not increase much until 95% of the entries are missing), and MBMS (row 4) could always further improve over SVP when it performs poorly. As a result, it is practically a good idea to apply a simpler and robust model first to obtain a reasonable initial reconstruction and then apply MBMS to refine the details for matrix completion.

3.5 Conclusion

We have proposed a new paradigm for matrix completion, denoising, which generalizes the commonly used assumption of low rank. Assuming low-rank implies a restrictive form of denoising where the data is forced to have zero variance away from a linear manifold. More general definitions of denoising can potentially handle data that lives in a low-dimensional manifold that is nonlinear, or whose dimensionality varies (e.g. a set of manifolds), or that does not have low rank at all, and naturally they handle noise in the data. Denoising works because of the fundamental fact that a missing value can be predicted by averaging nearby present values.

Although we motivate our framework from a constrained optimization point of view (*denoise subject to respecting the present data*), we argue for an algorithmic view of denoising-based matrix completion: *apply a denoising step, refill the present values, iterate until the validation error increases*. In turn, this allows different forms of denoising, such as based on low-rank projection (earlier work) or local averaging with blurring mean-shift (this chapter). Our nonparametric choice of mean-shift averaging further relaxes assumptions about the data and results in a simple algorithm with very few user parameters that afford user control (denoising scale, local dimensionality) but can be set automatically by cross-validation. Our algorithms are intended to be used as a postprocessing step over a user-provided initialization of the missing values, and we show they consistently improve upon existing algorithms.

The MBMS-based algorithm bridges the gap between pure denoising (GBMS) and local low rank. Other definitions of denoising should be possible, for example using temporal as well as spatial neighborhoods, and even applicable to discrete data if we consider denoising as a majority voting among the neighbours of a vector (with suitable definitions of votes and neighborhood).

Chapter 4

The K -modes algorithm for clustering

Many clustering algorithms exist that estimate a cluster centroid, such as K -means, K -medoids or mean-shift, but no algorithm seems to exist that clusters data by returning exactly K meaningful modes. We propose a natural definition of a K -modes objective function by combining the notions of density and cluster assignment. The algorithm becomes K -means and K -medoids in the limit of very large and very small scales. Computationally, it is slightly slower than K -means but much faster than mean-shift or K -medoids. Unlike K -means, it is able to find centroids that are valid patterns, truly representative of a cluster, even with nonconvex clusters, and appears robust to outliers and misspecification of the scale and number of clusters (Carreira-Perpiñán and Wang, 2013a).

4.1 Introduction

Given a dataset $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^D$, we consider clustering algorithms based on centroids, i.e., that estimate a representative $\mathbf{c}_k \in \mathbb{R}^D$ of each cluster k in addition to assigning data points to clusters. Two of the most widely used algorithms of this type are K -means and mean-shift. K -means has the number of clusters K as a user parameter and tries to

minimize the objective function

$$\begin{aligned} \min_{\mathbf{Z}, \mathbf{C}} E(\mathbf{Z}, \mathbf{C}) &= \sum_{k=1}^K \sum_{n=1}^N z_{nk} \|\mathbf{x}_n - \mathbf{c}_k\|^2 \\ \text{s.t. } z_{nk} &\in \{0, 1\}, \sum_{k=1}^K z_{nk} = 1, n = 1, \dots, N, k = 1, \dots, K \end{aligned} \quad (4.1)$$

where \mathbf{Z} are binary assignment variables (of point n to cluster k) and $\mathbf{C} = (\mathbf{c}_1, \dots, \mathbf{c}_K)$ are centroids, free to move in \mathbb{R}^D . At an optimum, centroid \mathbf{c}_k is the mean of the points in its cluster. Gaussian mean-shift (Carreira-Perpiñán, 2000; Cheng, 1995; Comaniciu and Meer, 2002; Fukunaga and Hostetler, 1975) assumes we have a kernel density estimate (kde) with bandwidth $\sigma > 0$ and kernel $G(t) = e^{-t/2}$ (up to some normalization constant)

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N G(\|\mathbf{x} - \mathbf{x}_n\|/\sigma)^2 \quad \mathbf{x} \in \mathbb{R}^D \quad (4.2)$$

and applies the iteration (started from each data point):

$$p(n|\mathbf{x}) = \frac{G(\|\mathbf{x} - \mathbf{x}_n\|/\sigma)^2}{\sum_{n'=1}^N G(\|\mathbf{x} - \mathbf{x}_{n'}\|/\sigma)^2}, \quad \mathbf{x} \leftarrow \mathbf{f}(\mathbf{x}) = \sum_{n=1}^N p(n|\mathbf{x}) \mathbf{x}_n. \quad (4.3)$$

which converges to a mode (local maximum) of p from nearly any initial \mathbf{x} (Carreira-Perpiñán, 2007). Each mode is the centroid for one cluster, which contains all the points that converge to its mode. The user parameter is the bandwidth σ and the resulting number of clusters depends on it implicitly.

The pros and cons of both algorithms are well known. K -means tends to define round clusters; mean-shift can obtain clusters of arbitrary shapes and has been very popular in low-dimensional clustering applications such as image segmentation (Comaniciu and Meer, 2002), but does not work well in high dimension. Both can be seen as special EM algorithms (Bishop, 2006; Carreira-Perpiñán, 2007). Both suffer from outliers, which can move centroids outside their cluster in K -means or create singleton modes in mean-shift. Computationally, K -means is much faster than mean-shift, at $\mathcal{O}(KND)$ and $\mathcal{O}(N^2D)$ per iteration, respectively, particularly with large datasets. In fact, accel-

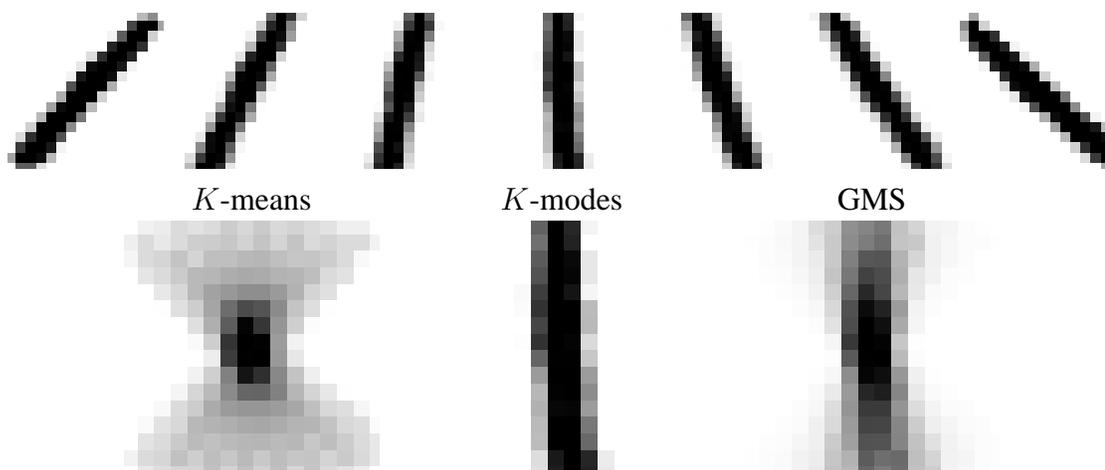


Figure 4.1: A cluster of 7 rotated-1 USPS digit images and the centroids found by K -means, K -modes (both with $K = 1$) and mean-shift (with σ so there is one mode).

erating mean-shift has been a topic of active research (Carreira-Perpiñán, 2006a; Yuan et al., 2010). Mean-shift does not require a value of K , which is sometimes convenient, although many users often find it desirable to force an algorithm to produce exactly K clusters (e.g. if prior information is available).

One important aspect in many applications concerns the validity of the centroids as patterns in the input space, as well as how representative they are of their cluster. Figure 4.1 illustrates this with a single cluster consisting of continuously rotated digit-1 images. Since these images represent a nonconvex cluster in the high-dimensional pixel space, their mean (which averages all orientations) is not a valid digit-1 image, which makes the centroid not interpretable and hardly representative of a digit 1. Mean-shift does not work well either: to produce a single mode, a large bandwidth is required, which makes the mode lie far from the manifold; a smaller bandwidth does produce valid digit-1 images, but then multiple modes arise for the same cluster, and under mean-shift they define each a cluster. Clustering applications that require valid centroids for nonconvex or manifold data abound (e.g. images, shapes or proteins).

A third type of centroid-based algorithms are exemplar-based or K -medoid clustering (Bishop, 2006; Hastie et al., 2009; Kaufman and Rousseeuw, 1990). These constrain the centroids to be points from the dataset (“exemplars”), such as K -medians, and often

minimize a K -means objective function (4.1) with a non-Euclidean distance. They are slow, since updating centroid \mathbf{c}_k requires testing all pairs of points in cluster k . Forcing the centroids to be exemplars is often regarded as a way to ensure the centroids are valid patterns. However, the exemplars themselves are often noisy and thus not that representative of their neighborhood. Not constraining a centroid to be an exemplar can remove such noise and produce a more typical representative.

Given that most location statistics have been used for clustering (mean, mode, median), it is remarkable that no K -modes formulation for clustering seems to exist, that is, an algorithm that will find exactly K modes that correspond to meaningful clusters. An obvious way to define a K -modes algorithm is to pick K modes from a kde, but it is not clear what modes to pick (assuming it has at least K modes, which will require a sufficiently small bandwidth). Picking the modes with highest density need not correlate well with clusters that have an irregular density, or an approximately uniform density with close but distinct high-density modes.

We define a K -modes objective as a natural combination of two ideas: the cluster assignment idea from K -means and the density maximization idea of mean-shift. The algorithm has two interesting special cases, K -means and a version of K -medoids, in the limits of large and small bandwidth, respectively. For small enough bandwidth, the centroids are denoised, valid patterns and typical representatives of their cluster. Computationally, it is slightly slower than K -means but much faster than mean-shift or K -medoids.

4.2 A K -modes Objective Function

We maximize the objective function

$$\begin{aligned} \max_{\mathbf{Z}, \mathbf{C}} L(\mathbf{Z}, \mathbf{C}) &= \sum_{k=1}^K \sum_{n=1}^N z_{nk} G(\|(\mathbf{x}_n - \mathbf{c}_k)/\sigma\|^2) \\ \text{s.t. } z_{nk} \in \{0, 1\}, \sum_{k=1}^K z_{nk} &= 1, n = 1, \dots, N, k = 1, \dots, K. \end{aligned} \quad (4.4)$$

For a given assignment \mathbf{Z} , this can be seen as (proportional to) the sum of a kde as in (4.2) but separately for each cluster. Thus, a good clustering must move centroids to local modes, but also define K separate kdes. This naturally combines the idea of clustering through binary assignment variables with the idea that high-density points are representative of a cluster (for suitable bandwidth values).

As a function of the bandwidth σ , the K -modes objective function has two interesting limit cases. When $\sigma \rightarrow \infty$, it becomes K -means. This can be seen from the centroid update (which becomes the mean), or from the objective function directly. Indeed, approximating it with Taylor's theorem for very large σ and using the fact that $\sum_{k=1}^K z_{nk} = 1$ gives

$$\begin{aligned} L(\mathbf{Z}, \mathbf{C}) &\approx \sum_{k=1}^K \sum_{n=1}^N z_{kn} (1 - \|\mathbf{x}_n - \mathbf{c}_k\|^2 / 2\sigma^2) \\ &= N - E(\mathbf{Z}, \mathbf{C}) / 2\sigma^2 \end{aligned}$$

where $E(\mathbf{Z}, \mathbf{C})$ is the same as in (4.1) and is subject to the same constraints. Thus, maximizing L becomes minimizing E , exactly the K -means problem. When $\sigma \rightarrow 0$, it becomes a K -medoids algorithm, since the centroids are driven towards data points. Thus, K -modes interpolates smoothly between these two algorithms, creating a continuous path that links a K -mean to a K -medoid. However, its most interesting behavior is for intermediate σ .

4.3 Two K -modes Algorithms

As is the case for K -means and K -medoids, minimizing the K -modes objective function is NP-hard. We focus on iterative algorithms that find a locally optimum clustering in the sense that no improvement is possible on the centroids given the current assignments, and vice versa. We give first an algorithm for fixed σ and then use it to construct a homotopy algorithm that sweeps over a σ interval.

4.3.1 For Fixed σ

It is convenient to use alternating optimization:

- **Assignment step** Over assignments \mathbf{Z} for fixed \mathbf{C} , the constrained problem separates into a constrained problem for each point \mathbf{x}_n , of the form

$$\max_{\mathbf{Z}_n} \sum_{k=1}^K z_{nk} g_{nk} \quad \text{s.t.} \quad \sum_{k=1}^K z_{nk} = 1, z_{nk} \in \{0, 1\},$$

with $g_{nk} = G(\|(\mathbf{x}_n - \mathbf{c}_k)/\sigma\|^2)$. The solution is given by assigning point \mathbf{x}_n to its closest centroid in Euclidean distance (assuming the kernel G is a decreasing function of the Euclidean distance).

- **Mode-finding step** Over centroids \mathbf{C} for fixed \mathbf{Z} , we have a separate unconstrained maximization for each centroid, of the form

$$L(\mathbf{c}_k) = \sum_{n=1}^N z_{nk} G(\|(\mathbf{x}_n - \mathbf{c}_k)/\sigma\|^2),$$

which is proportional to the cluster kde, and can be done with mean-shift. Note the step over \mathbf{C} need not be exact, i.e., the centroids need not converge to their corresponding modes. We exit when a tolerance is met or I mean-shift iterations have been run.

Thus, the algorithm operates similarly to K -means but finding modes instead of means: it interleaves a hard assignment step of data points to centroids with a mode-finding step that moves each centroid to a mode of the kde defined by the points currently assigned to it.

Convergence of this algorithm (in value) follows from the facts that each step (over \mathbf{Z} or over \mathbf{C}) is strictly feasible and decreases the objective or leaves it unchanged, and that the objective function is lower bounded by 0 within the feasible set. Besides, since there is a finite number of assignments, convergence occurs in a finite number of outer-loop steps (as happens with K -means) if the step over \mathbf{C} is exact and deterministic. By this we mean that for each \mathbf{c}_k we find deterministically a maximum of its objective function (i.e., the mode for \mathbf{c}_k is a deterministic function of \mathbf{Z}). This prevents the possibility that for the same assignment \mathbf{Z} we find different modes for a given \mathbf{c}_k , which could lead the algorithm to cycle. This condition can be simply achieved by using an optimization algorithm that either has no user parameters (such as step sizes; mean-shift is an example), or has user parameters set to fixed values, and running it to convergence. The $(\mathbf{Z}^*, \mathbf{C}^*)$ convergence point is a local maximum in the sense that $L(\mathbf{Z}^*, \mathbf{C})$ has a local maximum at $\mathbf{C} = \mathbf{C}^*$ and $L(\mathbf{Z}, \mathbf{C}^*)$ has a global maximum at $\mathbf{Z} = \mathbf{Z}^*$.

The computational cost per outer-loop iteration of this algorithm (setting $I = 1$ for simplicity in the mean-shift step) is identical to that of K -means: the step over \mathbf{Z} is $\mathcal{O}(KND)$ and the step over \mathbf{C} is $\mathcal{O}(N_1D + \dots + N_KD) = \mathcal{O}(ND)$ (where N_k is the number of points currently assigned to \mathbf{c}_k), for a total of $\mathcal{O}(KND)$. And also as in K -means, the steps parallelize: over \mathbf{C} , the mean-shift iteration proceeds independently in each cluster; over \mathbf{Z} , each data point can be processed independently.

4.3.2 Homotopy Algorithm

We start with $\sigma = \infty$ (i.e., run K -means, possibly several times and picking the best optimum). Then, we gradually decrease σ while running J iterations of the fixed- σ K -modes algorithm for each value of σ , until we reach a target value σ^* . This follows an optimum path $(\mathbf{Z}(\sigma), \mathbf{C}(\sigma))$ for $\sigma \in [\sigma^*, \infty)$. In practice, as is well known with

homotopy techniques, this tends to find better optima than starting directly at the target value σ^* . We use this homotopy algorithm in our experiments. Given we have to run K -means multiple times to find a good initial optimum (as commonly done in practice), the homotopy does not add much computation. Note that the homotopy makes K -modes a deterministic algorithm given the local optimum found by K -means.

4.3.3 User Parameters

The basic user parameter of K -modes is the desired number of clusters K . The target bandwidth σ^* in the homotopy is simply used as a scaling device to refine the centroids. We find that representative, valid centroids are obtained for a wide range of intermediate σ values. A good target σ^* can be obtained with a classical bandwidth selection criterion for kernel density estimates (Wand and Jones, 1994), such as the average distance to the k th nearest neighbor.

Practically, a user will typically be interested in the K centroids and clusters resulting for the target bandwidth. However, examining the centroid paths $\mathbf{c}_k(\sigma)$ can also be interesting for exploratory analysis of a dataset, as illustrated in our experiments with handwritten digit images.

4.4 Relation with Other Algorithms

K -modes is most closely related to K -means and to Gaussian mean-shift (GMS), since it essentially introduces the kernel density estimate into the K -means objective function. This allows K -modes to find exactly K true modes in the data (in its mathematical sense, i.e., maxima of the kde for each cluster), while achieving assignments as in K -means, and with a fast runtime, thus enjoying some of the best properties from both K -means and GMS.

K -means and K -modes have the same update step for the assignments, but the update step for the centroids is given by setting each centroid to a different location statistic

of the points assigned to it: the mean for K -means, a mode for K -modes. K -means and K -modes also define the same class of clusters (a Voronoi tessellation, thus convex clusters), while GMS can produce possibly nonconvex, disconnected clusters.

In GMS, the number of clusters equals the number of modes, which depends on the bandwidth σ . If one wants to obtain exactly K modes, there are two problems. The first one is computational: since K is an implicit, nonlinear function of σ , finding a σ value that produces K modes requires inverting this function. This can be achieved numerically by running mean-shift iterations while tracking $K(\sigma)$ as in scale-space approaches (Collins, 2003), but this is very slow. Besides, particularly for high-dimensional data, the kde only achieves K modes for a very narrow (even empty) interval of σ . The second problem is that even with an optimally tuned bandwidth, a kde will usually create undesirable, spurious modes where data points are sparse (e.g. outliers or cluster boundaries), again particularly with high-dimensional data. We avoid this problem in the homotopy version of K -modes by starting with large σ , which tracks important modes. The difference between K -modes and GMS is clearly seen in the particular case where we set $K = 1$ (as in Figure 4.1): K -modes runs the mean-shift update initialized from the data mean, so as σ decreases, this will tend to find a single, major mode of the kde. However, the kde itself will have many modes, all of which would become clusters under GMS.

The fundamental problem in GMS is equating modes with clusters. The true density of a cluster may well be multimodal to start with. Besides, in practice a kde will tend to be bumpy unless the bandwidth is unreasonably large, because it is by nature a sum of bumpy kernels centered at the data points. This is particularly so with outliers (which create small modes) or in high dimensions. There is no easy way to smooth out a kde (increasing the bandwidth does smooth it, but at the cost of distorting the overall density) or to postprocess the modes to select “good” ones. One has to live with the fact that a good kde will often have multiple modes per cluster.

K -modes provides one approach to this problem, by separating the roles of cluster assignment and of density. Each cluster has its own kde, which can be multimodal, and the homotopy algorithm tends to select an important mode among these within each cluster. This allows K -modes to achieve good results even in high-dimensional problems, where

GMS fails.

Computationally, K -modes and K -means are $\mathcal{O}(KND)$ per iteration for a dataset of N points in D dimensions. While K -modes in its homotopy version will usually take more iterations, this extra runtime is small because in practice one runs K -means multiple times from different initializations to achieve a better optimum. GMS is $\mathcal{O}(N^2D)$ per iteration, which is far slower, particularly with large datasets. The reason is that in GMS the kde involves all N points and one must run mean-shift iterations started from each of the N points. However, in K -modes the kde for cluster k involves only the N_k points assigned to it and one must run mean-shift iterations only for the centroid \mathbf{c}_k . Much work has addressed approximating GMS so that it runs faster, and some of it could be applied to the mean-shift step in K -modes, such as using Newton or sparse EM iterations (Carreira-Perpiñán, 2006a).

In addition to these advantages, our experiments show that K -modes can be more robust than K -means and GMS with outliers and with misspecification of either K or σ .

There are two variations of mean-shift that replace the local mean step of (4.3) with a different statistic: the local (Tukey) median (Shapira et al., 2009) and a medoid defined as any dataset point which minimizes a weighted sum of squared distances (Sheikh et al., 2007). Both are really medoid algorithms, since they constrain the centroids to be data points, and do not find true modes (maxima of the density). In general, K -medoid algorithms such as K -centers or K -medians are combinatorial problems, typically NP-hard (Hochbaum and Shmoys, 1985; Kaufman and Rousseeuw, 1990; Meyerson et al., 2004). In the limit $\sigma \rightarrow 0$, K -modes can be seen as a deterministic annealing approach to a K -medoids objective (just as the elastic net (Durbin and Willshaw, 1987) is for the traveling salesman problem).

There exists another algorithm called “ K -modes” (Chaturvedi et al., 2001; Huang, 1998). This is defined for categorical data and uses ℓ_0 error in its K -means type objective function. The “centroids” step boils down to finding for each dimension the most frequent value (thus the term “mode”, but mode is not well defined for high dimensional categorical data). It is quite different from our algorithm, which is defined for continuous data

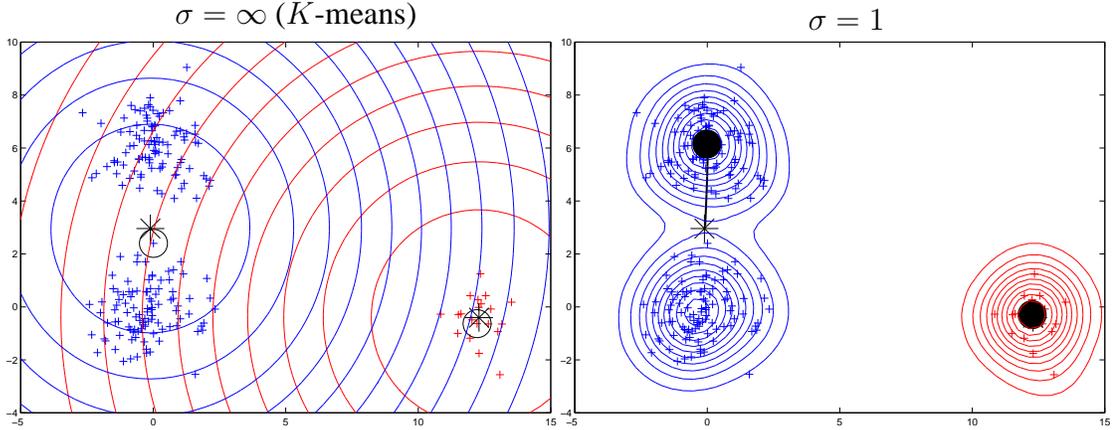


Figure 4.2: K -modes results for two bandwidth values using $K = 2$. We show the means $*$, their within-cluster nearest neighbor \circ , the modes \bullet , the paths followed by each mode as σ decreases, and the contours of each kde. Each K -modes cluster uses a different color.

and uses “mode” in its mathematical sense of density maximum.

4.5 Experimental results

We compare with K -means and Gaussian mean-shift (GMS) clustering. For K -means, we run it 20 times with different initializations and pick the one with minimum value of E in (4.1). For K -modes, we use its homotopy version initialized from the best K -means result and finishing at a target bandwidth (whose value is set either by using a kde bandwidth estimation rule or by hand, depending on the experiment).

4.5.1 Toy Examples

Figures 4.2 and 4.3 illustrate the three algorithms in 2D examples. They show the K modes and the kde contours for each cluster, for $\sigma = \infty$ or equivalently K -means (left panel) and for an intermediate σ (right panel). We run K -modes decreasing σ geometrically in 20 steps from 3 to 1 in Figure 4.2 and from 1 to 0.1 in Figure 4.3.

In Figure. 4.2, which has 3 Gaussian clusters, we purposefully set $K = 2$ (both K -

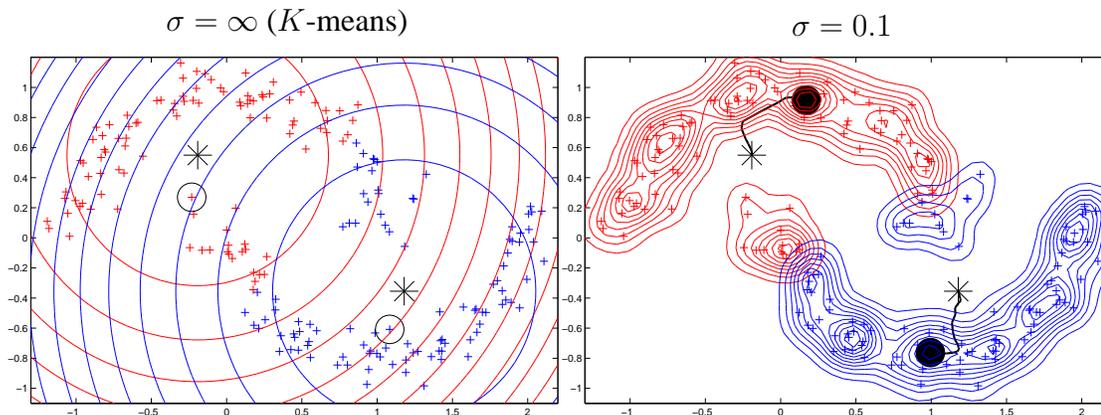


Figure 4.3: Like Figure 4.2 but for the two-moons dataset.

means and K -modes work well with $K = 3$). This makes K -means put one of the centroids in a low-density area, where no input patterns are found. K -modes moves the centroid inside a cluster in a maximum-density area, where many input patterns lie, and is then more representative.

In Figure 4.3, the “two-moons” dataset has two nonconvex, interleaved clusters and we set $K = 2$. The “moons” cannot be perfectly separated by either K -means or K -modes, since both define Voronoi tessellations. However, K -modes does improve the clusters over those of K -means, and as before it moves the centroids from a region where no patterns are found to a more typical location within each cluster. Note how, although the bandwidth used ($\sigma = 0.1$) yields a very good kde for each cluster and would also yield a very good kde for the whole dataset, it results in multiple modes for each “moon”, which means that GMS would return around 13 clusters. In this dataset, no value of σ results in two modes that separate the moons.

One might argue that, if a K -means centroid is not a valid pattern, one could simply replace it with the data point from its cluster that is closest to it. While this sometimes works, as would be the case in the rotated-digit-1 of Figure 4.1, it often does not: the same-cluster nearest neighbor could be a point on the cluster boundary, therefore atypical (Figure 4.2) or even a point in the wrong cluster (Figure 4.3). K -modes will find points interior to the clusters, with higher density and thus more typical.

4.5.2 Degree Distribution of a Graph

We construct an undirected graph similar to many real-world graphs and estimate the distribution of the degree of each vertex (Newman, 2010). To construct the graph, we generated a random (Erdős-Rényi) graph (with 1 000 vertices and 9 918 edges), which has a Gaussian degree distribution, and a graph with a power-law (long-tailed) distribution (with 3 000 vertices and 506 489 edges), and then took the union of both graphs and added a few edges at random connecting the two subgraphs. The result is a connected graph with two types of vertices, reminiscent of real-world networks such as the graph of web pages and their links in the Internet. Thus, our dataset has $N = 4\,000$ points in 1D (the degree of each vertex). As shown in Figure 4.4, the degree distribution is a mixture of two distributions that are well-separated but have a very different character: a Gaussian and a skewed, power-law distribution. The latter results in a few vertices having a very large degree (e.g. Internet hubs), which practically appear as outliers to the far right (outside the plots).

We set $K = 2$. K -means obtains a wrong clustering. One centroid is far to the right, in a low-density (thus unrepresentative) region, and determines a cluster containing the tail of the power-law distribution; this is caused by the outliers. The other centroid is on the head of the power-law distribution and determines a cluster containing the Gaussian and the head of the power-law distribution.

We run K -modes decreasing σ from 200 to 1 geometrically in 40 steps. K -modes shifts the centroids to the two principal modes of the distributions and achieves a perfect clustering. Note that the kde for the power-law cluster has many modes, but K -modes correctly converges to the principal one.

GMS cannot separate the two distributions for any value of σ . Setting σ small enough that the kde has the two principal modes implies it also has many small modes in the tail because of the outliers (partly visible in the second panel). This is a well-known problem with kernel density estimation.

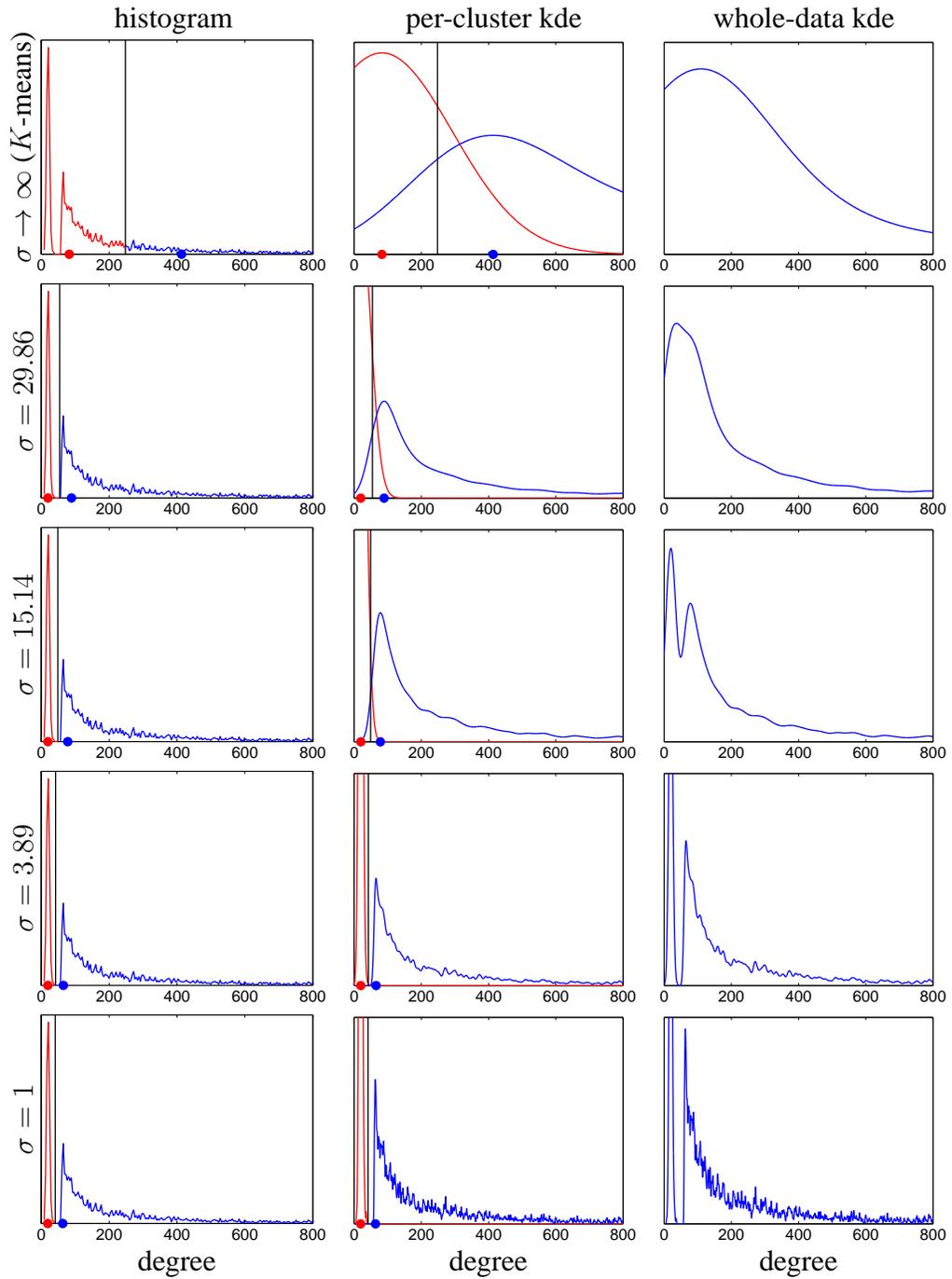


Figure 4.4: Degree distribution of a graph. *Left column:* a histogram of the distribution, colored according to the K -modes clustering for $\sigma = \infty$ (K -means) to $\sigma = 1$; the black vertical bar indicates the cluster boundary. *Middle column:* the kde for each cluster with K -modes. *Right column:* the kde for the whole dataset with GMS. The X axis is truncated to a degree of 800, so many outlying modes to the right are not shown.

4.5.3 Handwritten Digit Images

We selected 100 random images (16×16 grayscale) from the USPS dataset for each digit 0–9. This gives a dataset of $N = 1\,000$ points in $[0, 1]^{256}$. We ran K -means and K -modes with $K = 10$, decreasing σ from 10 to 1 geometrically in 100 steps.

Figure 4.5 shows that most of the centroids for K -means are blurry images consisting of an average of digits of different identity and style (slant, thickness, etc.), as seen from the 20 nearest-neighbor images of each centroid (within its cluster). Such centroids are hard to interpret and are not valid digit images. This also shows how the nearest neighbor to the centroid may be an unusual or noisy input pattern that is not representative of anything except itself.

K -modes unblurs the centroids as σ decreases. The class histograms for the 20 nearest-neighbors show how the purity of each cluster improves: for K -means most histograms are widely distributed, while K -modes concentrates the mass into mostly a single bin. This means K -modes moves the centroids onto typical regions that both look like valid digits, and are representative of their neighborhood. This can be seen not just from the class labels, but also from the style of the digits, which becomes more homogeneous under K -modes (e.g. see cluster c_2 , containing digit-6 images, or c_4 and c_5 , containing digit-0 images of different style).

Stopping K -modes at an intermediate σ (preventing it from becoming too small) achieves just the right amount of smoothing. It allows the centroids to look like valid digit images, but at the same time to average out noise, unusual strokes or other idiosyncrasies of the dataset images (while not averaging digits of different identities or different styles, as K -means does). This yields centroids that are more representative even than individual images of the dataset. In this sense, K -modes achieves a form of intelligent denoising similar to that of manifold denoising algorithms (Chapter 2).

Note that, for K -modes, centroids c_6 and c_9 look very similar, which suggests one of them is redundant (while none of the K -means centroids looked very similar to each other). Indeed, removing c_6 and rerunning K -modes with $K = 9$ simply reassigns

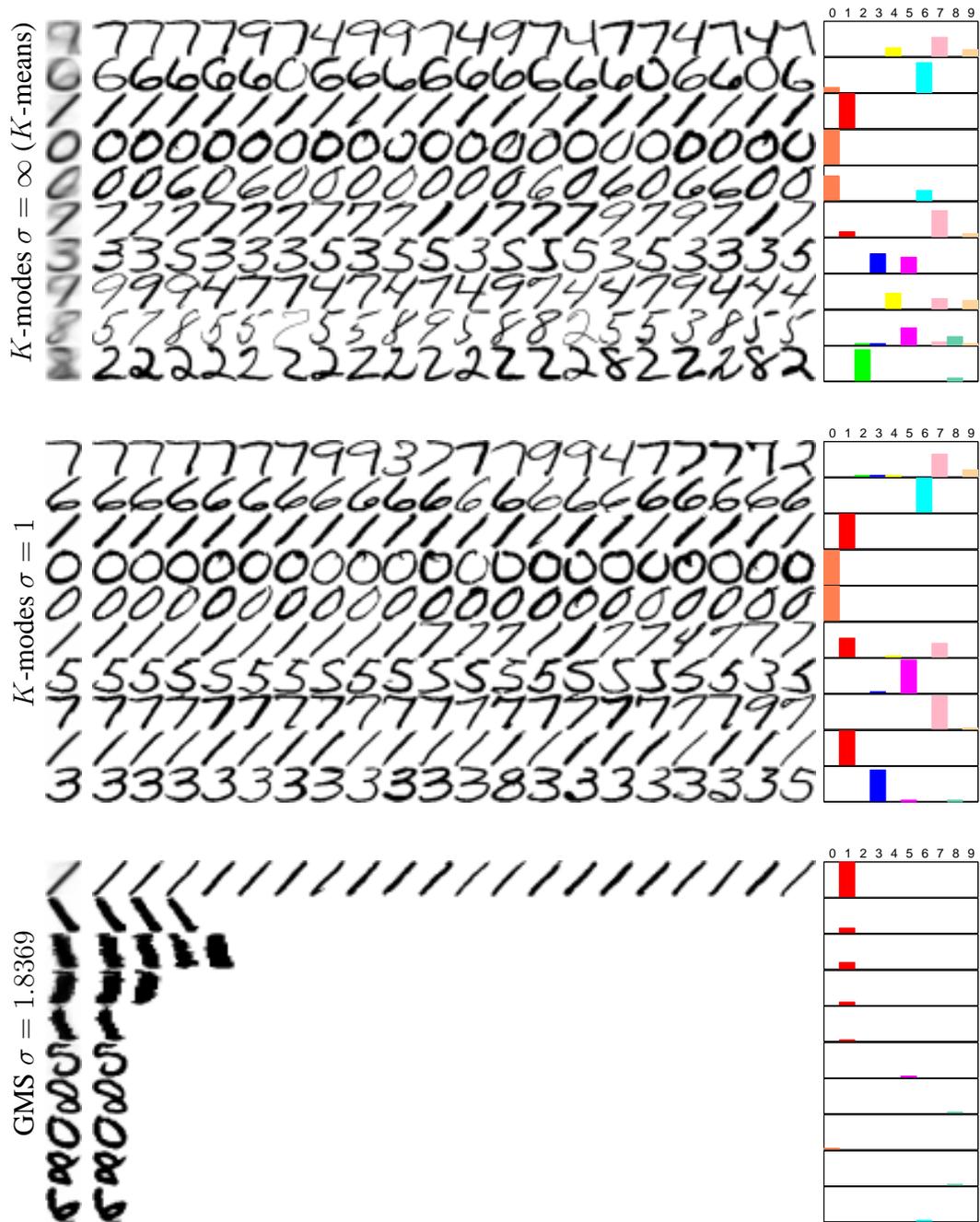


Figure 4.5: Clustering results on USPS data with K -modes with $K = 10$ for $\sigma = \infty$ (i.e., K -means, top panel) and $\sigma = 1$ (middle panel), and for GMS with $\sigma = 1.8369$ (bottom panel), which achieves $K = 10$ modes. In each panel, each row corresponds to a cluster $k = 1, \dots, K = 10$. The leftmost image shows the centroid \mathbf{c}_k and the right 20 images are the 20 nearest neighbors to it within cluster k . The right panel shows the histogram of class labels (color-coded) for the neighbors.

nearly all data points in the cluster of c_6 to that of c_9 and the centroid itself barely changes. This is likely not a casuality. If we have a single Gaussian cluster but use $K > 1$, it will be split into sectors like a pie, but in K -means the centroids will be apart from each other, while in K -modes they will all end up near the Gaussian center, where the mode of each kde will lie. This suggests that redundancy may be easier to detect in K -modes than in K -means.

GMS with $\sigma = 1.8369$ gives exactly 10 modes, however of these one is a slanted-digit-1 cluster like c_9 in K -modes and contains 98.5% of the training set points, and the remaining 9 modes are associated with clusters containing between 1 and 4 points only, and their centroids look like digits with unusual shapes, i.e., outliers. As noted before, GMS is sensitive to outliers, which create modes at nearly all scales. This is particularly so with high-dimensional data, where data is always sparse, or with data lying on a low-dimensional manifold (both of which occur here). In this case, the kde changes from a single mode for large σ to a multiplicity of modes over a very narrow interval of σ .

4.5.4 Summary

The previous experiments suggest that K -modes is more robust than K -means and GMS to outliers and parameter misspecification (K or σ). Outliers shift centroids away from the main mass of a cluster in K -means or create spurious modes in GMS, but K -modes is attracted to a major mode within each cluster. GMS is sensitive to the choice of bandwidth, which determines the number of modes in the kde. However, K -modes will return exactly K modes (one per cluster) no matter the value of the bandwidth, and whether the kde of the whole dataset has more or fewer than K modes. K -means is sensitive to the choice of K : if it is smaller than the true number of clusters, it may place centroids in low-density regions between clusters (which are invalid patterns); if it is larger than the true number of clusters, multiple centroids will compete for a cluster and partition it, yet the resulting centroids may show no indication that this happened. With K -modes, if K is too small the centroids will move inside the mass of each cluster and become valid patterns. If K is too large, centroids from different portions of a

cluster may look similar enough that their redundancy can be detected.

4.6 Discussion

While K -modes is a generic clustering algorithm, an important use is in applications where one desires representative centroids in the sense of being valid patterns, typical of their cluster, as described earlier. By making σ small enough, K -modes can always force the centroids to look like actual patterns in the training set (thus, by definition, valid patterns). However, an individual pattern is often noisy or idiosyncratic, and a more typical and still valid pattern should smooth out noise and idiosyncrasies—just as the idea of an “everyman” includes features common to most men, but does not coincide with any actual man. Thus, best results are achieved with intermediate bandwidth values: neither too large that they average widely different patterns, not too small that they average a single pattern, but just small enough that they average a local subset of patterns—where the average is weighted, as given by (4.3) but using points from a single cluster. Then, the bandwidth can be seen as a smoothing parameter that controls the representativeness of the centroids. Crucially, this role is separate from that of K , which sets the number of clusters, while in mean-shift both roles are conflated, since the bandwidth determines both the smoothing and the number of clusters.

How to determine the best bandwidth value? Intuitively, one would expect that bandwidth values that produce good densities should also give reasonable results with K -modes. Indeed, this was the case in our experiments using a simple bandwidth estimation rule (the average distance to the k th nearest neighbor). In general, what “representative” means depends on the application, and K -modes offers potential as an exploratory data analysis tool. By running the homotopy algorithm from large bandwidths to small bandwidths (where “small” can be taken as, say, one tenth of the result from a bandwidth estimator), the algorithm conveniently presents to the user a sequence of centroids spanning the smoothing spectrum. As mentioned before, the computational cost of this is comparable to that of running K -means multiple times to achieve a good optimum in the first place. Finally, in other applications, one may want to use K -modes

as a post-processing of the K -means centroids to make them more representative.

4.7 Conclusion and Future Work

Our K -modes algorithm allows the user to work with a kernel density estimate of bandwidth σ (like mean-shift clustering) but produce exactly K clusters (like K -means). It finds centroids that are valid patterns and lie in high-density areas (unlike K -means), are representative of their cluster and neighborhood, yet they average out noise or idiosyncrasies that exist in individual data points. Computationally, it is slightly slower than K -means but far faster than mean-shift. Theory and experiments suggest that it may also be more robust to outliers and parameter misspecification than K -means and mean-shift.

Our K -modes algorithm can use a local bandwidth at each point rather than a global one, and non-Gaussian kernels, in particular finite-support kernels (such as the Epanechnikov kernel) may lead to a faster algorithm.

A main application for K -modes is in clustering problems where the centroids must be interpretable as valid patterns. Beyond clustering, K -modes may also find application in problems where the data fall in a nonconvex low-dimensional manifold, as in finding landmarks for dimensionality reduction methods (de Silva and Tenenbaum, 2003), where the landmarks should lie on the data manifold; or in spectral clustering (Ng et al., 2002), where the projection of the data on the eigenspace of the graph Laplacian defines a hypersphere.

Chapter 5

The Laplacian K -modes algorithm for clustering

Aside from the merits of the K -modes algorithm proposed in Chapter 4, its major disadvantage is its assignment rule, which is the same as K -means and results in only convex clusters. In this chapter, we introduce a new algorithm based on K -modes, which essentially relaxes the discrete cluster indicator matrix to continuous variables and imposes a smoothness penalty on the assignments. We name this algorithm *Laplacian K -modes*. It naturally combines three powerful ideas in clustering: the explicit use of assignment variables (as in K -means); the estimation of cluster centroids which are modes of each cluster's density estimate (as in mean-shift); and the regularizing effect of the graph Laplacian, which encourages similar assignments for nearby points (as in spectral clustering). The optimization algorithm alternates an assignment step, which is a convex quadratic program, and a mean-shift step, which separates for each cluster centroid. The algorithm finds meaningful density estimates for each cluster, even with challenging problems where the clusters have manifold structure, are highly nonconvex or in high dimension. It also provides centroids that are valid patterns, truly representative of their cluster (unlike K -means), and an out-of-sample mapping that predicts soft assignments for a new point.

Table 5.1: Comparison of properties of different clustering algorithms.

	K -means	K -medoids	Mean-shift	Spectral clustering	K -modes	Laplacian K -modes
Centroids	invalid	“valid”	“valid”	N/A	valid	valid
Nonconv. clust.	no	depends	yes	yes	no	yes
Density	no	no	yes	no	yes	yes
Assignment	hard	hard	hard	hard	hard	soft
Cost/iteration	KND	KN^2D	N^2D	$N^2 \sim N^3$	KND	KND

5.1 Introduction

The K -modes algorithm (Chapter 4) combines the idea of clustering through binary assignment variables with the idea that high-density points are representative of a cluster. Each centroid found by the K -modes algorithm is the mode of a kde defined by data points in each cluster. As a result, the centroids average out noise or idiosyncrasies that exist in individual data points and are representative of their cluster and neighborhood. This can be seen from the K -modes centroid for the rotated digit-1 problem in Figure 4.1. K -modes was also shown to have nice properties such as being more robust to mis-specification of the bandwidth and to outliers, and enjoying an efficient optimization procedure.

One important disadvantage of K -modes is that it uses the same assignment rule as K -means (each point is assigned to its closest centroid in Euclidean distance), so it can only find convex clusters (a Voronoi tessellation). Therefore, like K -means, it cannot handle clusters with nonconvex shapes or manifold structure, unlike mean-shift or spectral clustering (Shi and Malik, 2000).

The main contribution of this chapter is to solve this issue, while keeping the nice properties that K -modes does have. The key idea is to modify the K -modes objective function such that the assignment rule becomes much more flexible. We then give an alternating optimization procedure to find the assignments and the modes. The resulting *Laplacian K -modes* algorithm is able to produce for each cluster a nonparametric density and a mode as valid representative (like K -modes), to separate nonconvex shaped clusters (like mean-shift and spectral clustering), and to give soft assignment of

data points to each cluster. Yet, all of these merits are achieved at a reasonable computational cost, and the algorithm works well with high-dimensional data. Table 5.1 compares Laplacian K -modes with other popular clustering algorithms.

5.2 Algorithm

5.2.1 The Laplacian K -Modes Algorithm

We change the assignment rule of K -modes to handle more complex shaped clusters based on two ideas: (1) the observation that *nearby data points should have similar assignments*; and (2) the use of *soft assignments*, which allows more flexibility in the clusters and simplifies the optimization. We first build a graph (e.g. k -nearest-neighbor graph) on the dataset, and let w_{mn} be an affinity (e.g. binary, heat kernel) between \mathbf{x}_m and \mathbf{x}_n . We then add to the K -modes objective function a Laplacian smoothing term $\frac{\lambda}{2} \sum_{m=1}^N \sum_{n=1}^N w_{mn} \|\mathbf{z}_m - \mathbf{z}_n\|^2$ to be minimized, where $\mathbf{z}_n = [z_{n1}, \dots, z_{nK}]^T$ is the assignment vector of \mathbf{x}_n , $n = 1, \dots, N$, to each of the K clusters, and $\lambda \geq 0$ is a trade-off parameter. The assignments are now continuous variables, but constrained to be positive and sum to 1. Thus, z_{nk} can be considered as the probability of assigning \mathbf{x}_n to cluster k (soft assignment). Thus, the *Laplacian K -modes* objective function is:

$$\begin{aligned} \min_{\mathbf{Z}, \mathbf{C}} \quad & \frac{\lambda}{2} \sum_{m=1}^N \sum_{n=1}^N w_{mn} \|\mathbf{z}_m - \mathbf{z}_n\|^2 - \sum_{n=1}^N \sum_{k=1}^K z_{nk} G \left(\left\| \frac{\mathbf{x}_n - \mathbf{c}_k}{\sigma} \right\|^2 \right) \quad (5.1) \\ \text{s.t.} \quad & \sum_{k=1}^K z_{nk} = 1, \quad n = 1, \dots, N, \\ & z_{nk} \geq 0, \quad n = 1, \dots, N, \quad k = 1, \dots, K. \end{aligned}$$

We can rewrite this objective in matrix form:

$$\begin{aligned} \min_{\mathbf{Z}, \mathbf{C}} \quad & \lambda \operatorname{tr}(\mathbf{Z}^T \mathbf{L} \mathbf{Z}) - \operatorname{tr}(\mathbf{B}^T \mathbf{Z}) \quad (5.2) \\ \text{s.t.} \quad & \mathbf{Z} \mathbf{1}_K = \mathbf{1}_N, \quad \mathbf{Z} \geq \mathbf{0} \end{aligned}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the graph Laplacian for the affinity matrix $\mathbf{W} = (w_{nm})$ and degree matrix $\mathbf{D} = \text{diag}(\sum_{n=1}^N w_{mn})$, $\mathbf{B} = (b_{nk})$ is an $N \times K$ matrix containing data-centroid affinities $b_{nk} = G(\|(\mathbf{x}_n - \mathbf{c}_k)/\sigma\|^2)$, $n = 1, \dots, N$, $k = 1, \dots, K$, $\mathbf{1}_K$ is a K dimensional vector of 1s and \geq means elementwise comparison. Other variations of the graph Laplacian can also be used (e.g. the normalized Laplacian), see von Luxburg (2007). The constraint on \mathbf{Z} shows it is a stochastic matrix. We can obtain a hard clustering if desired by assigning each point to the cluster with highest assignment value.

Special Cases of the Hyperparameters (λ, σ) In Laplacian K -modes, in addition to K there are two user parameters: λ controls the smoothness of the assignment, and σ controls the smoothness of the kde defined on each cluster. Consider first the case of $\lambda = 0$, where Laplacian K -modes becomes the original K -modes algorithm. Carreira-Perpiñán and Wang (2013a) already noted that the K -modes algorithm has two interesting limit cases: it becomes K -means when $\sigma \rightarrow \infty$, and a form of K -medoids when $\sigma \rightarrow 0$, since the centroids are driven towards data points. In both cases the assignments are hard (1-out-of- K coding). The case when $\lambda \rightarrow \infty$ makes the first term in (5.1) dominant and forces all connected points to have identical assignments, which is not interesting for the purpose of clustering. Therefore, the most interesting behavior of the algorithm is for intermediate λ . Finally, another interesting special case of Laplacian K -modes corresponds to $\lambda > 0$ and $\sigma \rightarrow \infty$, which we call *Laplacian K -means*, and which seems to be new as well.

5.2.2 Optimization Procedure for Laplacian K -modes

To solve (5.1), we use alternating optimization over \mathbf{C} and \mathbf{Z} , which takes advantage of the problem's structure.

C-step For fixed \mathbf{Z} , we are only concerned with the second term of (5.1) which is the K -modes objective. Therefore, our step over \mathbf{C} is identical to that of K -modes: it decouples over clusters and we apply mean-shift to solve for each \mathbf{c}_k separately. The

cost of this step is $\mathcal{O}(KND)$.

Z-step Unlike in K -modes, our \mathbf{Z} -step no longer decouples, which means we have to solve for NK variables all together. Since the graph Laplacian \mathbf{L} is positive semidefinite, the problem over \mathbf{Z} is a convex quadratic program (QP). While we could apply a standard QP algorithm, such as an interior point method, we provide here an algorithm that is very simple (no parameters to set), efficient and that scales well to real problems where the number of points N or the number of clusters K is very large. The solution is based on the gradient proximal algorithm used by Beck and Teboulle (2009). Their general framework solves convex problems of the form $\min_{\mathbf{x}} f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$, where g is convex and has Lipschitz continuous gradient (with constant L), and h is convex but not necessarily differentiable. The gradient proximal algorithm iteratively updates the variables by first taking a gradient step of the first function and then projecting it with the second function, i.e., $\mathbf{x}_{\tau+1} = \arg \min_{\mathbf{y}} \frac{L}{2} \|\mathbf{y} - (\mathbf{x}_{\tau} - \frac{1}{L} \nabla g(\mathbf{x}_{\tau}))\|^2 + h(\mathbf{y})$. It can be proven that the algorithm converges in objective function value with rate $\mathcal{O}(1/T)$ (where T is the iteration counter) with a *constant stepsize* $\frac{1}{L}$, and using Nesterov’s acceleration scheme improves the rate to $\mathcal{O}(1/T^2)$ (see Appendix B for details).

To apply this framework to our \mathbf{Z} -step, we make the identification that g is our smooth quadratic objective function, which has continuous gradient with $L = 2\lambda M$ being the (smallest) Lipschitz constant where M is the largest eigenvalue of \mathbf{L} , and h is the indicator function of the probability simplex. Consequently, our proximal step is computing the Euclidean projection of the gradient step onto the probability simplex. Note that computing the Euclidean projection onto the K dimensional simplex is itself a quadratic program. Fortunately, there exists an efficient algorithm which computes the exact projection with $\mathcal{O}(K \log K)$ time complexity (Duchi et al., 2008, also see Appendix A for details).

We provide the accelerated gradient projection algorithm for our \mathbf{Z} -step in Algorithm 1. Notice the graph Laplacian is sparse and its largest eigenvalue M can be obtained efficiently (e.g. by power iterations). Therefore the constant stepsize s can be easily determined right after constructing graph Laplacian. Compared to a pure gradient projection

Algorithm 1 Accelerated gradient projection for the \mathbf{Z} step.

Input: Initial $\mathbf{Z}_0 \in \mathbf{R}^{N \times K}$, $s = \frac{1}{2\lambda M}$ where M is the largest eigenvalue of the graph Laplacian \mathbf{L} .

- 1: Set $\mathbf{Y}_1 = \mathbf{Z}_0$, $t_1 = 1$, $\tau = 1$.
- 2: **repeat**
- 3: Compute gradient at \mathbf{Y}_τ : $\mathbf{G}_\tau = 2\lambda\mathbf{L}\mathbf{Y}_\tau - \mathbf{B}$
- 4: $\mathbf{Z}_\tau =$ simplex projection of each row of $\mathbf{Y}_\tau - s\mathbf{G}_\tau$
- 5: $t_{\tau+1} = (1 + \sqrt{1 + 4t_\tau^2})/2$
- 6: $\mathbf{Y}_{\tau+1} = \mathbf{Z}_\tau + (\frac{t_\tau - 1}{t_{\tau+1}})(\mathbf{Z}_\tau - \mathbf{Z}_{\tau-1})$
- 7: $\tau = \tau + 1$
- 8: **until** convergence

Output: \mathbf{Z}_τ is the solution of the \mathbf{Z} -step.

algorithm, the additional computational effort of the acceleration scheme in maintaining an auxiliary sequence \mathbf{Y} (lines 5–6 of Algorithm 1) is minimal, and we clearly observe an improved convergence behavior in practice. Each iteration of Algorithm 1 costs $\mathcal{O}(NK\rho + NK \log K)$, where ρ is the neighborhood size in constructing \mathbf{L} (or the number of nonzero entries in each row). The first term accounts for computing the gradient and the second term accounts for projecting each row of \mathbf{Z} onto the probability simplex. Notice that, although it is solving a large QP, the cost per iteration of our \mathbf{Z} -step is independent of the input dimensionality D , and is even less costly than the \mathbf{C} -step which has time complexity $\mathcal{O}(KND)$. Despite its sublinear convergence rate, the algorithm has a clear advantage in its simplicity: it does not require any line search or costly matrix operation, and is extremely easy to implement. We note that an alternative approach for the \mathbf{Z} -step is the Alternating Direction Method of Multipliers (ADMM, Boyd et al., 2011), which is applied in our recent work of a simple assignment model for the same type of optimization problem (Carreira-Perpiñán and Wang, 2013b).

Convergence Properties In the \mathbf{C} -step, each mean-shift update increases the density of the cluster kde (or leaves it unchanged) and its convergence rate to a mode is linear in general (Carreira-Perpiñán, 2007). Roughly speaking, achieving an approximate solution of error ϵ in this step takes $\log(1/\epsilon)$ iterations. In the \mathbf{Z} -step, the accelerated gradient projection converges theoretically at $\mathcal{O}(1/T^2)$ rate where T is iteration counter. Roughly speaking, achieving an approximate solution of error ϵ in this step takes $1/\sqrt{\epsilon}$

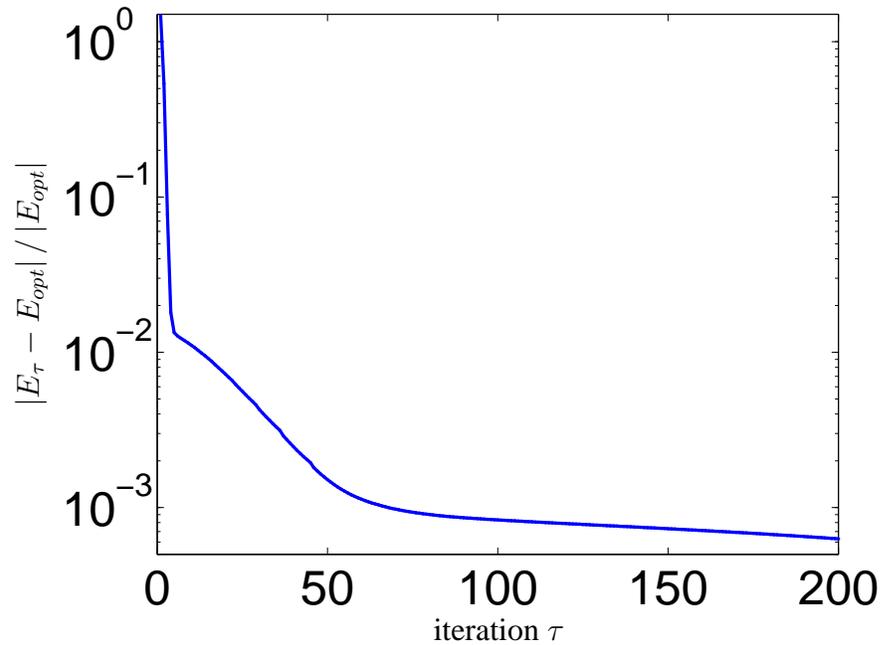


Figure 5.1: Learning curve of Laplacian K -modes on the synthetic 2-moons problem ($N = 1000$, $D = 2$, $K = 2$). We show the relative errors in objective function value ($|E_\tau - E_{opt}| / |E_{opt}|$) over iterations (τ) of our alternating optimization scheme for user parameters ($\lambda = 1$, $\sigma = 0.2$).

iterations, although gradient projection seems to perform much better than its theoretical guarantee in practice (Beck and Teboulle, 2009). We could further improve the convergence rate of gradient projection to be linear, by adding a quadratic regularization on \mathbf{Z} to the objective function such that it becomes strongly convex (see Appendix B). We alternate the \mathbf{C} and \mathbf{Z} steps until a convergence criterion is satisfied (e.g. the change to the variables is below some threshold). In an efficient implementation, both steps should be inexact (e.g. each could run for a fixed, small number of iterations). Note that the \mathbf{Z} -step algorithm is feasible, so exiting it early produces valid assignments.

It is to be understood that, even though the per iteration cost of Laplacian K -modes has the same complexity as K -modes and K -means in terms of problem size (Table 5.1), Laplacian K -modes may need many more iterations to converge to an accurate solution (in fact K -means is guaranteed to converge to an exact local minimum in finite steps). To illustrate this point, we show in Figure 5.1 the learning curve of Laplacian K -modes

on the synthetic example of two moons (dataset is shown in Figure 5.3) for some fixed user parameter.

Homotopy Algorithm As with K -means and K -modes, the Laplacian K -modes objective function has local optima, which are caused by the nonlinear, kde term. One strategy to find a good optimum consists of first finding a good optimum for K -means and then run a homotopy algorithm initialized there. We can construct a homotopy by varying continuously λ from 0 and σ from ∞ , which corresponds to K -means, to their target values (λ^*, σ^*) . In practice, we follow this path approximately, by running some iterations of the fixed- (λ, σ) Laplacian K -modes algorithm for each value of (λ, σ) . In practice, as is well known with homotopy techniques, this tends to find better optima than starting directly at the target value (λ^*, σ^*) . A good optimum for K -means can be obtained by picking the best of several random restarts, or by using the K -means++ initialization strategy, which has approximation guarantees (Arthur and Vassilvitskii, 2007).

5.2.3 Out-of-sample Problem

We now consider the out-of-sample problem, that is, given an unseen test point $\mathbf{x} \in \mathbb{R}^D$, we wish to find a meaningful assignment $\mathbf{z}(\mathbf{x})$ to the clusters found during training. A natural and efficient way to do this is to solve a problem of the same form as (5.1) with a dataset consisting of the original training set augmented with \mathbf{x} , but keeping \mathbf{Z} and \mathbf{C} fixed to the values obtained during training (this avoids having to solve for all points again). After dropping constant terms, this is equivalent to the following problem:

$$\begin{aligned} \min_{\mathbf{z}} \quad & \lambda \sum_{n=1}^N w_n \|\mathbf{z} - \mathbf{z}_n\|^2 - \sum_{k=1}^K z_k G\left(\left\|\frac{\mathbf{x}_n - \mathbf{c}_k}{\sigma}\right\|^2\right) \\ \text{s.t.} \quad & z_k \geq 0, \quad k = 1, \dots, K, \quad \sum_{k=1}^K z_k = 1 \end{aligned}$$

where w_n is the affinity between test point \mathbf{x} and training point \mathbf{x}_n . After some algebra, the above optimization problem further reduces to the following quadratic program:

$$\min_{\mathbf{z}} \quad \frac{1}{2} \|\mathbf{z} - (\bar{\mathbf{z}} + \gamma \mathbf{q})\|^2 \quad \text{s.t.} \quad \mathbf{z}^T \mathbf{1}_K = 1, \quad \mathbf{z} \geq \mathbf{0} \quad (5.3)$$

where the expressions for $\bar{\mathbf{z}}$, $\mathbf{q} = [q_1, \dots, q_K]^T$ and γ are as follows:

$$\begin{aligned} \bar{\mathbf{z}} &= \sum_{n=1}^N \frac{w_n}{\sum_{n'=1}^N w_{n'}} \mathbf{z}_n, \\ q_k &= \frac{G(\|(\mathbf{x} - \mathbf{c}_k)/\sigma\|^2)}{\sum_{k'=1}^K G(\|(\mathbf{x} - \mathbf{c}_{k'})/\sigma\|^2)}, \\ \gamma &= \frac{\sum_{k=1}^K G(\|(\mathbf{x} - \mathbf{c}_k)/\sigma\|^2)}{2\lambda \sum_{n=1}^N w_n}. \end{aligned}$$

Thus, the out-of-sample solution is the projection of the K -dimensional vector $\bar{\mathbf{z}} + \gamma \mathbf{q}$ onto the probability simplex. The computational cost is $\mathcal{O}(ND)$, dominated by the cost of $\bar{\mathbf{z}}$, since the simplex projection costs $\mathcal{O}(K \log K)$.

The solution has an intuitive interpretation, consisting of the linear combination of two terms, each a valid assignment vector (having positive elements that sum to 1). The Laplacian term, $\bar{\mathbf{z}}$, is the weighted average of the neighboring training points' assignments, and results in nonconvex clusters. The kde term, \mathbf{q} , assigns a point based on its distances (posterior probabilities) to the centroids, and results in convex clusters. These two distinct assignment rules are combined using a weight γ to give the final assignment. Essentially, \mathbf{x} is assigned to cluster k with high probability if its nearby points are assigned to it (\bar{z}_k is large) or if it is close to \mathbf{c}_k (q_k is large). Although the out-of-sample mapping is defined variationally, it is just as useful as a closed-form expression: computationally it does not require an iterative procedure, and the interpretation above makes its meaning clear. This interpretation also illuminates the meaning of the Laplacian in the training objective (5.1). In fact, iterating the out-of-sample mapping sequentially over the training points gives another (slower) way to solve the \mathbf{Z} -step, i.e., alternating optimization over $\mathbf{z}_1, \dots, \mathbf{z}_N$.

5.3 A brief review of related work

We have discussed centroids-based algorithms (K -means, K -modes, mean-shift) at length in Chapter 4. Here we review other closely related work of Laplacian K -modes in the clustering context.

Obtaining hard assignments by optimizing over a discrete cluster indicator matrix is usually difficult, because interesting objective functions are typically NP-hard. Spectral clustering algorithms (Shi and Malik, 2000; Yu and Shi, 2003) avoid this difficulty by first approximating the solution using eigenvectors of the normalized graph Laplacian. Formally, spectral clustering solves the following optimization problem (\mathbf{D} and \mathbf{W} are defined as in Section 5.2)

$$\begin{aligned} \min_{\mathbf{Z}} \quad & \text{tr}(\mathbf{Z}^T \mathbf{W} \mathbf{Z}) \\ \text{s.t.} \quad & \mathbf{Z}^T \mathbf{D} \mathbf{Z} = \mathbf{I}. \end{aligned} \tag{5.4}$$

And the global optimum of this non-convex problem is obtained by the K leading eigenvectors of $\mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}$. However, since the eigenvectors do not readily provide valid assignments, these algorithms need to run another clustering algorithm (usually K -means) on the eigenvectors to obtain actual partitions of the data—a post-processing step that may have multiple local optima. In Laplacian K -modes, we relax \mathbf{Z} to be a stochastic matrix, so our \mathbf{Z} -step results from a convex QP and provides soft assignments of points to clusters, which may also be used as posterior probabilities.

Laplacian smoothing has also been used in combination with nonnegative matrix factorization (NMF) for clustering (Cai et al., 2011). NMF learns a decomposition of the input data matrix where both basis and coefficients are nonnegative, and tends to produce part-based representation of the data (Lee and Seung, 1999). Cai et al. (2011) add to the NMF objective function a Laplacian smoothing term regarding the coefficient matrix, so that data points that are close in input space are encouraged to have a similar representation

using the common basis set. Formally, it solves the following optimization problem

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V}} \quad & \|\mathbf{X} - \mathbf{UV}\|^2 + \text{tr}(\mathbf{V}\mathbf{L}\mathbf{V}^T) \\ \text{s.t.} \quad & \mathbf{U} \geq \mathbf{0}, \quad \mathbf{V} \geq \mathbf{0}, \end{aligned} \quad (5.5)$$

where $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_K] \in \mathbb{R}^{D \times K}$ are the basis/dictionary of the dataset and $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_N] \in \mathbb{R}^{K \times N}$ are the coefficients for each data point. Note the Laplacian smoothing term $\text{tr}(\mathbf{V}^T\mathbf{L}\mathbf{V})$ imposes smoothness on the coefficients/codes. Similar to spectral clustering, the coefficients \mathbf{V} provides the representation of the dataset using learnt basis and does not give clustering assignment either. Thus K -means is then applied to \mathbf{V} obtain a final partition of the data.

There has been recent work in clustering that directly optimizes over a stochastic assignment matrix. Arora et al. (2011) optimize over a stochastic matrix \mathbf{Z} such that \mathbf{ZZ}^T best approximates a rescaled similarity matrix. Formally, it solves the following optimization problem

$$\begin{aligned} \min_{c, \mathbf{Z}} \quad & \|c\mathbf{W} - \mathbf{ZZ}^T\|^2 \\ \text{s.t.} \quad & c > 0, \quad \mathbf{Z}\mathbf{1}_K = \mathbf{1}_N, \quad \mathbf{Z} \geq \mathbf{0}. \end{aligned} \quad (5.6)$$

It is easy to see that the optimization problem has multiple solutions which are related by rotations about the normal to the probability simplex (e.g., permutations of the columns of \mathbf{Z} which change the cluster labels). The authors propose to exploits the geometry of the problem using rotation-based algorithm, which is straightforward for up to $K = 4$ clusters and requires optimization procedure for computing projection onto probability simplex for more clusters.

Yang and Oja (2012) use the idea of AnchorGraphs (Liu et al., 2010) to approximate the affinities between data points through a two-step Data-Cluster-Data (DCD) random walk. Assume uniform prior distribution $P(i) = 1/N$ over the data points, the random

walk probabilities from cluster \mathbf{c}_k to data point \mathbf{x}_i can be written as

$$P(i|k) = \frac{P(k|i)P(i)}{\sum_{n=1}^N P(k|n)P(n)} = \frac{z_{ik}}{\sum_{n=1}^N z_{nk}}.$$

Then consider then the two-step random walks from \mathbf{x}_i to \mathbf{x}_j via the augmented cluster nodes $\{\mathbf{c}_k\}_{k=1}^K$:

$$P(i|j) = \sum_{k=1}^K P(i|k)P(k|j) = \sum_{k=1}^K \frac{z_{ik}z_{jk}}{\sum_{n=1}^N z_{nk}}.$$

The authors then minimize the generalized KL divergence between a given sparse affinity matrix \mathbf{W} and the affinities resulted from DCD random walks over the stochastic matrix \mathbf{Z} :

$$\begin{aligned} \min_{\mathbf{Z}} \quad & \sum_{i=1}^N \sum_{j=1}^N \left(P(i|j) \log \frac{P(i|j)}{w_{ij}} - P(i|j) + w_{ij} \right) \quad (5.7) \\ \text{s.t.} \quad & \mathbf{Z}\mathbf{1}_K = \mathbf{1}_N, \quad \mathbf{Z} \geq \mathbf{0}. \end{aligned}$$

It is obvious that the above approaches are related to Laplacian K -modes in that they all optimize certain objective over the assignment probabilities. Laplacian K -modes has a very simple quadratic problem over the assignments \mathbf{Z} , whereas the objective function of Yang and Oja (2012) is heavily nonlinear. Thanks to the kde term in the objective function, Laplacian K -modes does not have the issue of rotational equivalence of Arora et al. (2011), and makes use of the efficient projection onto the probability simplex to deal with any number of clusters. Furthermore, Laplacian K -modes provides density estimate of the clusters and prototypical centroids while the above algorithms do not.

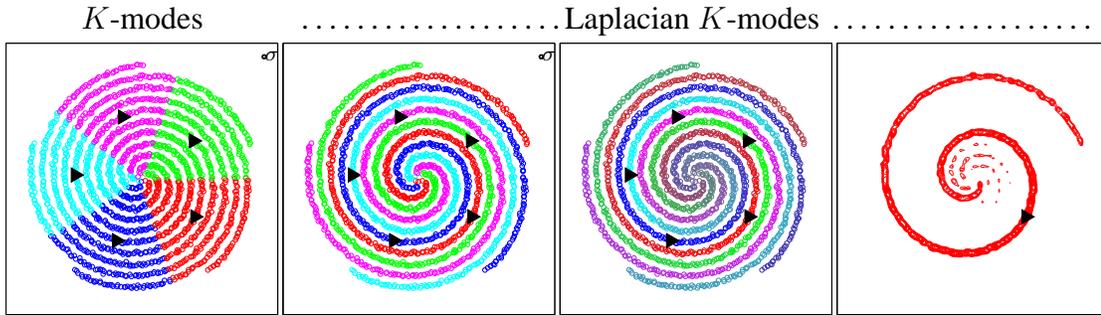


Figure 5.2: Synthetic dataset of 5-spirals. From left to right: K -modes clustering ($\lambda = 0$, $\sigma = 0.2$, the circle at the top right corner has a radius of σ); Laplacian K -modes clustering ($\lambda = 100$, $\sigma = 0.2$); Laplacian K -modes assignment probabilities; contours of the kde of the “red” cluster.

5.4 Experimental results

5.4.1 Illustrative Synthetic Examples

Spirals We first demonstrate the power of Laplacian smoothing. The 2D dataset in Figure 5.2 consist of 5 spirals where each spiral contains 400 points (denoted by \circ). The natural way of partitioning this dataset into $K = 5$ groups is to assign points of each spiral into a separate cluster. Due to the nonconvex shape of the spirals, the ideal result can not be possibly achieved by K -modes (plot 1, we color each point differently according to the cluster it is assigned to) or K -means (achieves a similar result to K -modes which is not shown here), even though the K -modes centroids (denoted by \blacktriangleright) are lying on each spiral and are valid representatives of the dataset. We then build 5-nearest neighbor graph on this dataset using heat kernel weighting, and run Laplacian K -modes using K -means result as initialization. We achieve perfect separation of the spirals and one centroid for each spiral in few steps of our alternating optimization scheme, as shown in plot 2. We show the assignment probabilities \mathbf{Z} in plot 3, where each data point \mathbf{x}_n is colored using a mixture of the 5 clusters’ colors with its assignment probability z_n being the mixing coefficient. We show the contours of the kde defined on the “red” cluster in plot 4, which is localized to the cluster and represents its shape well. It is obvious that running mean-shift on this dataset with the same σ will result in a large

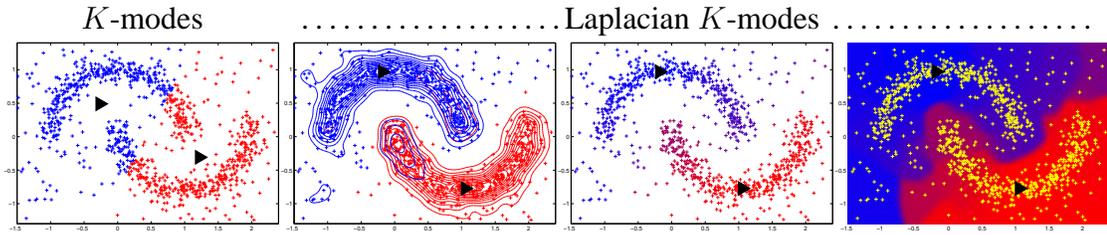


Figure 5.3: Synthetic dataset of 2-moons. We denote data points by $+$ and centroids by \blacktriangleright . We run Laplacian K -modes in homotopy and show results at final parameter value ($\lambda = 1$ and $\sigma = 0.1$). From left to right: K -modes clustering ($\lambda = 0$, $\sigma \rightarrow \infty$); Laplacian K -modes clustering and contours of kde of each cluster; Laplacian K -modes assignment probabilities, using the same coloring scheme as in 5-spirals; out-of-sample mapping in input space, colored in the same way as assignment probabilities of plot 3 (training points are now plotted in yellow).

number of modes and therefore clusters. In contrast, the number of modes is fixed in Laplacian K -modes and the algorithm will track one of the major modes in each cluster.

It is interesting to notice that because the kernel width σ we use is quite small, only a small proportion of data points are close enough to centroids to have nonzero affinity. This implies that the \mathbf{B} matrix in (5.2) is quite sparse. Nonetheless, we achieve good assignment probabilities using the graph Laplacian, which propagates the sparse “label” information in \mathbf{B} throughout the graph. This also partly explains the success of Laplacian smoothing in spectral clustering (Shi and Malik, 2000) and semi-supervised learning algorithms (Belkin et al., 2006; Zhu et al., 2003).

Noisy Two Moons We demonstrate the out-of-sample mapping of Laplacian K -modes on the “two-moons” dataset in Figure 5.3. The dataset has two nonconvex, interleaved clusters (each has 400 points) and we set add massive outliers (200 points) around them. The “moons” cannot be perfectly separated by either K -means (results shown in plot 1) or K -modes, since both define Voronoi tessellations. This problem is also difficult for hierarchical clustering because, as is well known, its major problem is that it creates connections between different clusters as the merging occurs. We build 5-nearest neighbor graph on this dataset using heat kernel weighting, and run Laplacian K -modes from K -means initialization. We run the homotopy version and reduce σ from 5 to 0.1

in 10 steps while fixing $\lambda = 1$. The obtained hard partition, along with the two centroids and kdes for each cluster at $\sigma = 0.1$ are given in plot 2. With the existence of heavy noise/outliers, the “inliers” are still perfectly separated, the modes lie in high density area and we obtain good density estimate for each cluster. We show the assignment probabilities \mathbf{Z} in plot 3, colored using the same scheme as in the spirals example. The assignment is certain near the centroids (purer color) and obscure at the boundaries and outliers (mixed color). Finally, the out-of-sample mapping in input space is shown in plot 4, where we compute out-of-sample assignment for a fine grid and color each grid point using the same coloring scheme as in plot 3. We see clearly that the assignment rule is very different from the hard assignment of K -means. The mapping at each point combines the average assignment of nearby training points and the assignment from centroids, and has complicated, flexible shape.

Figure-Ground Segmentation We consider the problem of segmenting an occluder from a textured background in a grayscale image. This problem has been shown to be difficult for spectral clustering (Carreira-Perpiñán and Zemel, 2005; Chennubhotla and Jepson, 2003), because of the intensity gradients between the occluder and the background (and within the background itself), which cause many graph edges to connect them, see the example in Figure 5.4. We formalize it as a clustering problem and partition the pixels into $K = 5$ clusters. We use for each pixel its 2D location and intensity value as features, and build a graph where each pixel is connected to the eight nearby pixels, with edge weighted using a heat kernel of width σ (same value is used as Gaussian kernel width for Laplacian K -modes). The goal is to have one of the clusters extract the occluder, which can then be separated from the background. To measure the performance, we choose the cluster that overlap the most with occluder as positive prediction (the rest pixels are considered as background/negative prediction) and compute classification error rate. As we can see from Figure 5.4, normalized cut (Yu and Shi, 2003) performs well for a narrow range of σ , while Laplacian K -modes (with fixed $\lambda = 0.1$) has much more stable performance when using the same graph. This is due to the fact that our algorithm has different, more flexible assignment rule compared to spectral clustering.

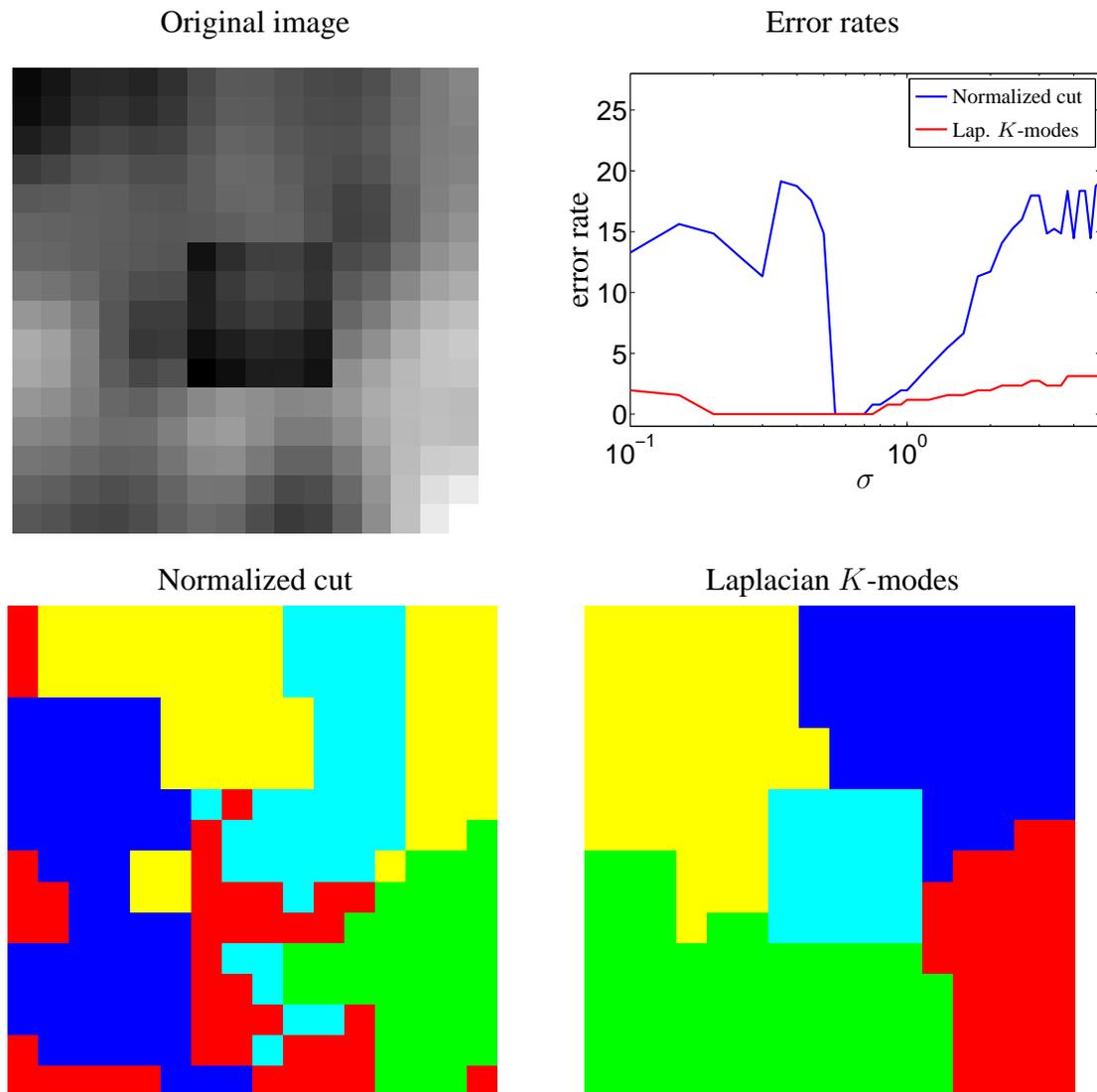


Figure 5.4: Occluder segmentation result. Top: original image and error rates over range of σ . Bottom: segmentation of normalized cut and Laplacian K -modes at $\sigma = 0.2$.



Figure 5.5: Clustering results on USPS data with Laplacian K -modes with $K = 10$, $\lambda = 0.12$ and homotopy for σ . Each row corresponds to a cluster $k = 1, \dots, K = 10$. The leftmost image shows the centroid c_k and the right 20 images are the 20 nearest neighbors to it within cluster k . The right panel shows the histogram of class labels (color-coded) for the neighbors.

USPS Digits We now cluster the USPS digits subset used in Section 4.5. Starting from the same k -means result in Figure 4.5, we run the homotopy version of Laplacian K -modes and gradually decrease σ from 10 to 1 in 100 steps (the same values used by K -modes), while fixing $\lambda = 0.12$. A 5-nearest-neighbor graph is built on this dataset to compute the Laplacian. We show the centroids and their neighborhood found by Laplacian K -modes in Figure 5.5. We see that, similar to K -modes, the centroids are valid patterns, and the neighborhood of each centroid are homogeneous. Notice these centroids represent more classes than those of K -modes (c_{10} is a prototypical 2). But still, the centroids can not uncover all the 10 classes. An explanation is that the sample size of this dataset is relatively small and the variations within each class is not very smooth, and it is therefore hard to build a graph that more or less connects images of the same class and separate different classes. A potential solution is to use features that are more invariant to within class variations rather than pixel values.

5.4.2 Clustering Analysis

We report clustering statistics in datasets with known pattern class labels (which the algorithms did not use): (1) MNIST, which contains 28×28 grayscale handwritten digit

Table 5.2: Statistics (size, dimensionality, number of classes) of the three real world datasets.

dataset	N	D	K
MNIST	2000	784	10
COIL-20	1440	1024	20
TDT2	9394	36771	30

Table 5.3: Clustering accuracy (%) on three datasets. N/A means our GMS code ran out of memory.

dataset	K -means	K -modes	GMS	NCut	GNMF	DCD	Laplacian K -modes
MNIST	58.2	59.2	15.9	65.5	66.2	69.4	70.5
COIL-20	66.5	67.2	27.2	79.0	75.3	71.5	81.0 (81.5)
TDT2	68.9	70.0	N/A	88.4	88.6	55.1	91.4

images (we randomly sample 200 of each digit); (2) COIL-20, which contains 32×32 grayscale images of 20 objects viewed from varying angles; (3) the NIST Topic Detection and Tracking (TDT2) corpus, which contains on-topic documents of different semantic categories (documents appearing in more than one category are removed and only the largest 30 categories are kept). Statistics of the datasets are collected in table 5.2. Datasets (2) and (3) are the same as used by Cai et al. (2011), and we also use the same features: pixel values for (1) and (2), and TFIDF for (3).

We compare the following algorithms: K -means, initialized randomly; K -modes, a special case of Laplacian K -modes with $\lambda = 0$; Gaussian mean-shift (GMS), we search for σ that produces exactly K modes; Normalized cut (NCut), one typical spectral clustering algorithm, and we use the implementation of Yu and Shi (2003); Graph regularized NMF (GNMF) proposed by Cai et al. (2011); Data-Cluster-Data random walk (DCD) proposed by Yang and Oja (2012); and Laplacian K -modes, initialized from K -means.

Several algorithms uses graph Laplacian: for NCut, GNMF, and Laplacian K -modes, we build 5-nearest-neighbor graph and use binary weighting scheme for computing graph Laplacian (same as in Cai et al., 2011); for DCD, we find that it achieve better performance using a graph built with larger neighborhood size, so we let DCD select optimal size in $\{5, 10, 20, 30\}$. We run each algorithm with 20 random restarts, letting them use respective optimal hyper-parameter (if there is any) based on grid search, and report

Table 5.4: Normalized Mutual Information (%) on three datasets.

dataset	K -means	K -modes	GMS	NCut	GNMF	DCD	Laplacian K -modes
MNIST	53.3	53.6	6.51	66.9	64.9	65.6	68.8
COIL-20	75.3	75.9	38.9	88.0	87.5	77.6	87.3 (88.0)
TDT2	75.3	75.8	N/A	83.7	83.7	68.6	88.8

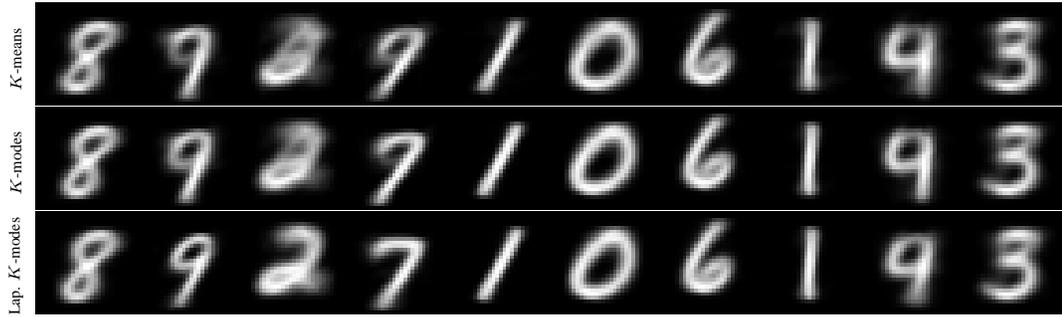


Figure 5.6: Centroids found by different algorithms on MNIST. First row: K -means ($\lambda = 0, \sigma \rightarrow \infty$, ACC: 55.2%, NMI: 50.2%). Second row: K -modes ($\lambda = 0, \sigma = 0.35$, ACC: 56.0%, NMI: 50.6%). Third row: Laplacian K -modes ($\lambda = 0.07, \sigma = 0.35$, ACC: 70.5%, NMI: 68.8%).

the best performance from different random restarts. Performance evaluations using two criteria—accuracy (ACC) and normalized mutual information (NMI)—are given in tables 5.3 and 5.4, respectively. It is clear that algorithms using Laplacian smoothing are in general superior than algorithms not using it, demonstrating the importance of graph Laplacian in separating nonconvex and manifold clusters. GMS performs poorly for the reasons described earlier. On all datasets, Laplacian K -modes achieves the best or close to best performance under both criteria (we find in practice there exist wide range of hyper-parameters with which our algorithm gives very competitive performance). We are able to further improve our performance on COIL-20 using the homotopy technique described earlier: we fix λ at 0.01, and decrease σ from 0.45 to 0.1 gradually in 7 steps, initializing the algorithm for current σ value from solution for the previous σ value. This improved result is shown in parenthesis in tables 5.3 and 5.4.

Another key advantage of Laplacian K -modes is that the centroids are interpretable patterns of the dataset. We show the centroids (each as a image) found by centroids-based algorithms (using optimal hyper-parameter) on MNIST in Figure 5.6 and COIL-20 in

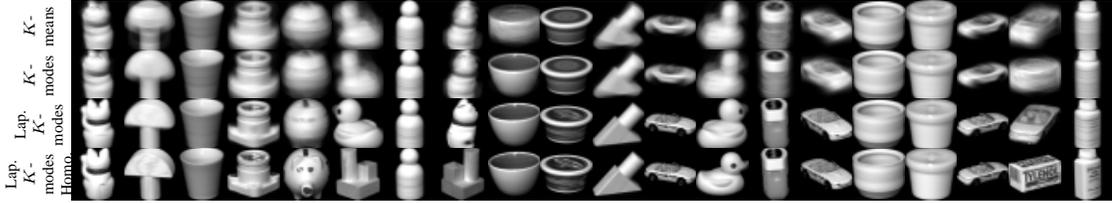


Figure 5.7: Centroids found by different algorithms on COIL-20. First row: K -means ($\lambda = 0$, $\sigma \rightarrow \infty$, ACC: 64.8%, NMI: 73.5%). Second row: K -modes ($\lambda = 0$, $\sigma=0.3$, ACC: 65.5%, NMI: 73.0%). Third row: Laplacian K -modes ($\lambda = 0.01$, $\sigma = 0.1$, ACC: 73.2%, NMI: 83.8%). Fourth row: Laplacian K -modes with homotopy in σ ($\lambda = 0.01$, ACC: 81.5%, NMI: 88.0%, see text).

Figure 5.7, all using the K -means initialization. For such high dimensional problems with small K , GMS tends to have majority of the centroids associated with very few points that are outliers with unusual patterns, and we do not show them here. Not surprisingly, some K -means centroids are blurry images consisting of an average of digits/objects of different identity and style. This implies some centroids lie between different branches of data manifolds, thus in low density area and not prototypical. K -modes centroids have cleaner shapes, but the identities of the centroids somewhat overlap. This is because K -modes concerns about kdes only, and multiple modes might move to the same manifold which happen to have higher density. Laplacian K -modes centroids not only have prototypical shapes, but also covers more digit/object identities (such centroids should help obtain better accuracy).

Applying our algorithm at an intermediate σ achieves just the right amount of smoothing. This is clearly seen from the centroids obtained on MNIST. It allows the centroids to look like valid digit images, but at the same time to average out noise, unusual strokes or other idiosyncrasies of the dataset images (while not averaging digits of different identities or different styles, as K -means does). This yields centroids that are more representative even than individual images of the dataset.

5.5 Conclusion

Our Laplacian K -modes algorithm enjoys some of the best properties of a range of clustering algorithms. It is nonparametric and allows the user to work with a kernel density estimate that produces exactly K clusters (as in K -means and K -modes), even in high dimension (unlike in mean-shift), and which can be nonconvex (as in mean-shift and spectral clustering). It also finds centroids that are valid patterns and lie in high-density areas, are representative of their cluster and neighborhood, yet they average out noise or idiosyncrasies that exist in individual data points. Computationally, our current alternating optimization scheme is simple, efficient and scales well. Experiments demonstrate the superior performance of Laplacian K -modes compared to well-known clustering algorithms.

There are some obvious extensions to the algorithm that might improve the performance. Our Laplacian smoothing term could use more carefully constructed graphs (Zelnik-Manor and Perona, 2005) or better normalized Laplacian (Zass and Shashua, 2007) in place of the usual Laplacian. We could also use localized kernel width σ (Vladymyrov and Carreira-Perpiñán, 2013) for each data point to obtain better kde. Moreover, we would like to investigate about the automatic choice of hyper-parameters and design homotopy procedure in parameter λ .

Chapter 6

Concluding remarks

We have shown in this thesis some successful examples of learning the manifold structure of data using the mean-shift algorithms. Here we give some concluding remarks and concerns for future research (or what is not fully investigated in this thesis regarding the topic of manifold learning with mean-shift).

- Popular applications of mean-shift algorithms are usually in relatively low dimensions (Carreira-Perpiñán, 2006a,b, 2008; Comaniciu and Meer, 2002; Comaniciu et al., 2003). This is because, in high dimensions, one may need significant amount of data to obtain a good kernel density estimate. An intuitive reasoning is through the kernel bandwidth selection: since data points are sparse in high dimensional Euclidean spaces, a small bandwidth at each point includes only the point itself and cause the density to be a sum of δ functions, while a big bandwidth includes all points and locality is lost, thus the range of “good” bandwidth may be very narrow. Exploiting the manifold structure can overcome the curse of dimensionality to some extent, as we have shown empirically in this thesis. Yet one may still wonder what is the limit of this approach, e.g., what is the sample complexity for the learning algorithms in this thesis to work well, or when can we even believe the local tangent space estimated in a neighborhood using PCA is a good approximate of the manifold structure? Recent theoretical research may shed some light on such problems (Canas et al., 2012).

- There has been work on generalizing the mean-shift procedure for mode finding in Euclidean space to certain Riemannian manifolds (Cetingül and Vidal, 2009; Subbarao and Meer, 2006, 2009) (similar generalization exist for K -means, Banerjee et al., 2005). Such algorithms involve procedures of moving on the tangent plane and mapping the resulting vector back to the manifold, so as to guarantee to operate “on” the manifold of interest, although clustering of manifold can be achieved via dimension reduction and clustering in the latent space as well (Goh and Vidal, 2008). It would be interesting to establish a connection between these approaches with the MBMS algorithms in this thesis, which also constrain the mean-shift motion with regard to the manifold structure.
- Random projection has become increasingly popular for machine learning (Baraniuk and Wakin, 2009; Candes and Tao, 2006; Dasgupta and Freund, 2009; Li and Hastie, 2008; Wakin et al., 2006). The power of random projection can be characterized by the famous Johnson-Lindenstrauss lemma (Dasgupta and Gupta, 1999; Johnson and Lindenstrauss, 1984). Roughly speaking, the lemma implies that one can reduce the dimensionality of a dataset to $\mathcal{O}(\log N)$ where N is the dataset size with certain random projections, without distorting the distance between any pair of points too much. Baraniuk et al. (2006) show that the same type of random projections also satisfy the *restricted isometry property* requirement on the sensing operator for compressed sensing (Donoho, 2006). Therefore, random projection is a very efficient dimensionality reduction technique as it requires little or no training, and it is effective as it maintains most of the useful information within the dataset, e.g., it approximately preserves the pairwise distances between data points, and allows for exact recovery of sparse signals. As we have seen, the time complexity of mean-shift is linear in the input dimensionality and thus mean-shift is costly for very high dimensional dataset (e.g., documents represented by TFIDF features), exacerbating the slow speed due to its convergence rate. We expect random projection to be useful for reducing the time complexity of mean-shift, by first projecting very high dimensional dataset into moderate dimensions, without sacrificing much the clustering accuracy.

- The revived interest on mean-shift can be attributed to its simplicity and wide applications, mainly in computer vision. Therefore, another future direction is to apply the newly developed mean-shift algorithms to real-world applications, while incorporating various domain knowledge.

Appendix A

Projection onto the probability simplex

We provide an elementary proof of a simple, efficient algorithm for computing the Euclidean projection of a point onto the probability simplex. This algorithm is used in Laplacian K -modes clustering in Chapter 5.

A.1 Problem

Consider the problem of computing the Euclidean projection of a point $\mathbf{y} = [y_1, \dots, y_D]^T \in \mathbb{R}^D$ onto the probability simplex, which is defined by the following optimization problem:

$$\min_{\mathbf{x} \in \mathbb{R}^D} \frac{1}{2} \|\mathbf{x} - \mathbf{y}\|^2 \tag{A.1a}$$

$$\text{s.t. } \mathbf{x}^T \mathbf{1} = 1 \tag{A.1b}$$

$$\mathbf{x} \geq \mathbf{0}. \tag{A.1c}$$

This is a quadratic program and the objective function is strictly convex, so there is a unique solution which we denote by $\mathbf{x} = [x_1, \dots, x_D]^T$ with a slight abuse of notation.

A.2 Algorithm

The following $\mathcal{O}(D \log D)$ algorithm finds the solution \mathbf{x} to the problem:

Algorithm 2 Euclidean projection of a vector onto the probability simplex.

Input: $\mathbf{y} \in \mathbb{R}^D$

Sort \mathbf{y} into \mathbf{u} : $u_1 \geq u_2 \geq \dots \geq u_D$

Find $\rho = \max\{1 \leq j \leq D: u_j + \frac{1}{j}(1 - \sum_{i=1}^j u_i) > 0\}$

Define $\lambda = \frac{1}{\rho}(1 - \sum_{i=1}^{\rho} u_i)$

Output: \mathbf{x} s.t. $x_i = \max\{y_i + \lambda, 0\}, i = 1, \dots, D$.

The complexity of the algorithm is dominated by the cost of sorting the components of \mathbf{y} . The algorithm is not iterative and identifies the active set exactly after at most D steps. It can be easily implemented (see section A.4).

The algorithm has the following geometric interpretation. The solution can be written as $x_i = \max\{y_i + \lambda, 0\}, i = 1, \dots, D$, where λ is chosen such that $\sum_{i=1}^D x_i = 1$. Place the values y_1, \dots, y_D as points on the X axis. Then the solution is given by a rigid shift of the points such that the points to the right of the Y axis sum to 1.

The pseudocode above appears in Duchi et al. (2008), although earlier papers (Brucker, 1984; Pardalos and Koor, 1990) solved the problem in greater generality¹.

Other algorithms The problem (A.1) can be solved in many other ways, for example by particularizing QP algorithms such as active-set methods, gradient-projection methods or interior-point methods. It can also be solved by alternating projection onto the two constraints in a finite number of steps (Michelot, 1986). Another way (Boyd and Vandenberghe, 2004, Exercise 4.1, solution available at <http://see.stanford.edu/materials/lsocoe364b/hw4sol.pdf>) is to construct a Lagrangian formulation by dualizing the equality constraint and then solve a 1D nonsmooth optimization over the Lagrange multiplier. Algorithm 2 has the advantage of being very simple,

¹<http://www.cs.berkeley.edu/~jduchi/projects/DuchiShSiCh08.html>

not iterative, and identifying exactly the active set at the solution after at most D steps (each of cost $O(1)$) after sorting.

A.3 A simple proof

A proof of correctness of Algorithm 2 can be found in Shalev-Shwartz and Singer (2006) and Chen and Ye (2011), but we offer a simpler proof which involves only the KKT theorem.

We apply the standard KKT conditions for the problem (Nocedal and Wright, 2006). The Lagrangian of the problem is

$$\mathcal{L}(\mathbf{x}, \lambda, \boldsymbol{\beta}) = \frac{1}{2} \|\mathbf{x} - \mathbf{y}\|^2 - \lambda(\mathbf{x}^T \mathbf{1} - 1) - \boldsymbol{\beta}^T \mathbf{x}$$

where λ and $\boldsymbol{\beta} = [\beta_1, \dots, \beta_D]^T$ are the Lagrange multipliers for the equality and inequality constraints, respectively. At the optimal solution \mathbf{x} the following KKT conditions hold:

$$x_i - y_i - \lambda - \beta_i = 0, \quad i = 1, \dots, D \quad (\text{A.2a})$$

$$x_i \geq 0, \quad i = 1, \dots, D \quad (\text{A.2b})$$

$$\beta_i \geq 0, \quad i = 1, \dots, D \quad (\text{A.2c})$$

$$x_i \beta_i = 0, \quad i = 1, \dots, D \quad (\text{A.2d})$$

$$\sum_{i=1}^D x_i = 1. \quad (\text{A.2e})$$

From the complementarity condition (A.2d), it is clear that if $x_i > 0$, we must have $\beta_i = 0$ and $x_i = y_i + \lambda > 0$; if $x_i = 0$, we must have $\beta_i \geq 0$ and $x_i = y_i + \lambda + \beta_i = 0$, whence $y_i + \lambda = -\beta_i \leq 0$. Obviously, the components of the optimal solution \mathbf{x} that are zeros correspond to the smaller components of \mathbf{y} . Without loss of generality, we assume

the components of \mathbf{y} are sorted and \mathbf{x} uses the same ordering, i.e.,

$$\begin{aligned} y_1 &\geq \cdots \geq y_\rho \geq y_{\rho+1} \geq \cdots \geq y_D, \\ x_1 &\geq \cdots \geq x_\rho > x_{\rho+1} = \cdots = x_D, \end{aligned}$$

and that $x_1 \geq \cdots \geq x_\rho > 0$, $x_{\rho+1} = \cdots = x_D = 0$. In other words, ρ is the number of positive components in the solution \mathbf{x} . Now we apply the last condition and have

$$1 = \sum_{i=1}^D x_i = \sum_{i=1}^{\rho} x_i = \sum_{i=1}^{\rho} (y_i + \lambda)$$

which gives $\lambda = \frac{1}{\rho}(1 - \sum_{i=1}^{\rho} y_i)$. Hence ρ is the key to the solution. Once we know ρ (there are only D possible values of it), we can compute λ , and the optimal solution is obtained by just adding λ to each component of \mathbf{y} and thresholding as in the end of Algorithm 2. (It is easy to check that this solution indeed satisfies all KKT conditions.) In the algorithm, we carry out the tests for $j = 1, \dots, D$ if $t_j = y_j + \frac{1}{j}(1 - \sum_{i=1}^j y_i) > 0$. We now prove that the number of times this test turns out positive is exactly ρ . The following theorem is essentially Lemma 3 of Shalev-Shwartz and Singer (2006).

Theorem 1. Let ρ be the number of positive components in the solution \mathbf{x} , then

$$\rho = \max\{1 \leq j \leq D: y_j + \frac{1}{j}(1 - \sum_{i=1}^j y_i) > 0\}.$$

Proof. Recall from the KKT conditions (A.2) that $\lambda\rho = 1 - \sum_{i=1}^{\rho} y_i$, $y_i + \lambda > 0$ for $i = 1, \dots, \rho$ and $y_i + \lambda \leq 0$ for $i = \rho + 1, \dots, D$. In the sequel, we show that for $j = 1, \dots, D$, the test will continue to be positive until $j = \rho$ and then stay non-positive afterwards, i.e., $y_j + \frac{1}{j}(1 - \sum_{i=1}^j y_i) > 0$ for $j \leq \rho$, and $y_j + \frac{1}{j}(1 - \sum_{i=1}^j y_i) \leq 0$ for $j > \rho$.

(i) For $j = \rho$, we have

$$y_\rho + \frac{1}{\rho} \left(1 - \sum_{i=1}^{\rho} y_i \right) = y_\rho + \lambda = x_\rho > 0.$$

(ii) For $j < \rho$, we have

$$\begin{aligned} y_j + \frac{1}{j} \left(1 - \sum_{i=1}^j y_i \right) &= \frac{1}{j} \left(jy_j + 1 - \sum_{i=1}^j y_i \right) = \frac{1}{j} \left(jy_j + \sum_{i=j+1}^{\rho} y_i + 1 - \sum_{i=1}^{\rho} y_i \right) \\ &= \frac{1}{j} \left(jy_j + \sum_{i=j+1}^{\rho} y_i + \rho\lambda \right) = \frac{1}{j} \left(j(y_j + \lambda) + \sum_{i=j+1}^{\rho} (y_i + \lambda) \right). \end{aligned}$$

Since $y_i + \lambda > 0$ for $i = j, \dots, \rho$, we have $y_j + \frac{1}{j}(1 - \sum_{i=1}^j y_i) > 0$.

(iii) For $j > \rho$, we have

$$\begin{aligned} y_j + \frac{1}{j} \left(1 - \sum_{i=1}^j y_i \right) &= \frac{1}{j} \left(jy_j + 1 - \sum_{i=1}^j y_i \right) = \frac{1}{j} \left(jy_j + 1 - \sum_{i=1}^{\rho} y_i - \sum_{i=\rho+1}^j y_i \right) \\ &= \frac{1}{j} \left(jy_j + \rho\lambda - \sum_{i=\rho+1}^j y_i \right) = \frac{1}{j} \left(\rho(y_j + \lambda) + \sum_{i=\rho+1}^j (y_j - y_i) \right). \end{aligned}$$

Notice $y_j + \lambda \leq 0$ for $j > \rho$, and $y_j \leq y_i$ for $j \geq i$ since \mathbf{y} is sorted, therefore $y_j + \frac{1}{j}(1 - \sum_{i=1}^j y_i) < 0$.

□

Remarks

1. We denote $\lambda_j = \frac{1}{j}(1 - \sum_{i=1}^j y_i)$. At the j -th test, λ_j can be considered as a guess of the true λ (indeed, $\lambda_\rho = \lambda$). If we use this guess to compute a tentative solution $\bar{\mathbf{x}}$ where $\bar{x}_i = \max\{y_i + \lambda_j, 0\}$, then it is easy to see that $\bar{x}_i > 0$ for $i = 1, \dots, j$, and $\sum_{i=1}^j \bar{x}_i = 1$. In other words, the first j components of $\bar{\mathbf{x}}$ are positive and sum to 1. If we find $\bar{x}_{j+1} = 0$ (or $y_{j+1} + \lambda_j \leq 0$), then we know we have found the optimal solution and $j = \rho$ because $\bar{\mathbf{x}}$ satisfies all KKT conditions.
2. To extend the algorithm to a simplex with a different scale, i.e., $\mathbf{x}^T \mathbf{1} = a$ for $a > 0$, replace the $1 - \sum u_i$ terms with $a - \sum u_i$ in Algorithm 2.

A.4 Matlab code

The following Matlab code implements algorithm 2.

```
function X = SimplexProj(Y)

[N,D] = size(Y);
X = sort(Y,2,'descend');
Xtmp = (cumsum(X,2)-1)*diag(sparse(1./(1:D)));
X = max(bsxfun(@minus,Y,Xtmp(sub2ind([N,D],...
(1:N)',sum(X>Xtmp,2))))),0);
```

Appendix B

Convergence rates of the gradient projection algorithms

In this appendix, we provide proofs of the convergence properties of the gradient projection algorithm. Namely, the convergence rate is sublinear for weakly convex functions and linear for strongly convex functions. Moreover, we show that Nesterov's acceleration scheme can be used to improve the convergence speed significantly in both cases.

B.1 Basic algorithm

There has been great recent interest in optimizing convex function of the composite form $f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$ in the field of machine learning, where g is a (usually differentiable) fitting term, and h is a (usually non-smooth) regularization term which gives inductive bias towards desirable models (e.g. sparse solution is preferred when using a ℓ_1 regularization). Many algorithms (Becker et al., 2011; Cai et al., 2010; Duchi et al., 2010; Ji and Ye, 2009; Lan, 2012; Nesterov, 2007) exploit the fact that several interesting, commonly used regularizers h leads to simple and efficient projection step (see Xiao, 2010, Section 2 for a short list of such regularizers), and thus it is wise to treat g and h separately instead of treating their sum as a single convex function. Beck and

Algorithm 3 Gradient projection algorithm.

Input: L =Lipschitz constant of ∇g , some constant $\nu > 0$ (inverse stepsize).

- 1: **repeat**
- 2: $\mathbf{x}_{\tau+1} = p_\nu(\mathbf{x}_\tau)$,
- 3: $\tau = \tau + 1$,
- 4: **until** convergence.

Output: \mathbf{x}_τ is the solution.

Teboulle (2009) took this approach and incorporated Nesterov’s acceleration scheme, and demonstrated the superior performance of accelerated gradient projection (FISTA) on large scale ℓ_1 problem, which makes the algorithm very popular at the time.

The original gradient projection algorithm dates back to Rockafellar (1976), hence the proof for the convergence properties is certainly not new. Here I simply follow the steps of Beck and Teboulle (2009) and fill in all the missing details to give a complete elementary proof and to make this note self-contained. The gradient projection algorithm for solving problems of the form $f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$ is sketched in Algorithm 3, where the step p_ν is defined as

$$p_\nu(\mathbf{x}) = \arg \min_{\mathbf{y}} \frac{\nu}{2} \left\| \mathbf{y} - \left(\mathbf{x} - \frac{1}{\nu} \nabla g(\mathbf{x}) \right) \right\|^2 + h(\mathbf{y}). \quad (\text{B.1})$$

B.2 Weakly convex functions

For now, the only assumptions about the convex function g are that it is continuously differentiable, and its gradient ∇g is Lipschitz continuous with constant $L > 0$. In this section, we call such functions *weakly* convex functions (to be differentiated from the *strongly* convex functions introduced later). In order to prove the convergence properties of the algorithm, we first need several results about general convex functions, which appear in many papers and textbooks (e.g. Bertsekas, 1999; Nesterov, 2004). The first one concerns a quadratic upper bound of a smooth function derived from its Lipschitz constant.

Lemma B.2.1. *Let $g : \mathbb{R}^n \rightarrow \mathbb{R}$ be a continuous differentiable function and ∇g is*

Lipschitz continuous with constant $L(g)$. Then, for any $L \geq L(g)$, we have

$$g(\mathbf{x}) \leq g(\mathbf{y}) + \langle \nabla g(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{L}{2} \|\mathbf{x} - \mathbf{y}\|^2 \text{ for every } \mathbf{x}, \mathbf{y} \in \mathbb{R}^n. \quad (\text{B.2})$$

Proof. Consider first the case in which g is a function of single variable x . According to the assumption, we have $|g'(1) - g'(0)| < L(g)$. Moreover,

$$\begin{aligned} g(1) &= g(0) + \int_0^1 g'(x) dx \\ &= g(0) + \int_0^1 g'(0) dx + \int_0^1 (g'(x) - g'(0)) dx \\ &\leq g(0) + g'(0) + \int_0^1 |g'(x) - g'(0)| dx \\ &\leq g(0) + g'(0) + L(g) \int_0^1 x dx \\ &= g(0) + g'(0) + \frac{1}{2} L(g). \end{aligned}$$

Now consider the case in which g is a multi-variate function. It is obvious that $\phi(t) = g(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))$ is a uni-variate function, and $\phi'(t) = \langle \nabla g(\mathbf{x} + t(\mathbf{y} - \mathbf{x})), \mathbf{y} - \mathbf{x} \rangle$. Additionally, $\phi'(t)$ has a Lipschitz constant of $L(\phi) \leq L(g) \|\mathbf{y} - \mathbf{x}\|^2$. This is because for any t_1 and t_2 , we have

$$\begin{aligned} |\phi'(t_1) - \phi'(t_2)| &= |\langle \nabla g(\mathbf{x} + t_1(\mathbf{y} - \mathbf{x})) - \nabla g(\mathbf{x} + t_2(\mathbf{y} - \mathbf{x})), \mathbf{y} - \mathbf{x} \rangle| \\ &\leq \|\nabla g(\mathbf{x} + t_1(\mathbf{y} - \mathbf{x})) - \nabla g(\mathbf{x} + t_2(\mathbf{y} - \mathbf{x}))\| \|\mathbf{y} - \mathbf{x}\| \\ &\leq L(g) \|\mathbf{y} - \mathbf{x}\|^2 |t_1 - t_2|. \end{aligned}$$

Thus we can apply the previous result for uni-variate case to $\phi(t)$, and obtain

$$\begin{aligned} g(\mathbf{y}) &= \phi(1) \leq \phi(0) + \phi'(0) + \frac{1}{2} L(\phi) \\ &\leq g(\mathbf{x}) + \langle \nabla g(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L}{2} \|\mathbf{x} - \mathbf{y}\|^2. \end{aligned}$$

□

This lemma also leads to the following useful inequalities.

Corollary B.2.2. *Let g and L satisfy the assumptions in Lemma B.2.1. Then the following inequalities hold for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$:*

$$g(\mathbf{y}) \geq g(\mathbf{x}) + \langle \nabla g(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{1}{2L} \|\nabla g(\mathbf{x}) - \nabla g(\mathbf{y})\|^2, \quad (\text{B.3})$$

$$\langle \nabla g(\mathbf{x}) - \nabla g(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \geq \frac{1}{L} \|\nabla g(\mathbf{x}) - \nabla g(\mathbf{y})\|^2. \quad (\text{B.4})$$

Proof. Consider the function $\phi(\mathbf{y}) = g(\mathbf{y}) - \langle \nabla g(\mathbf{x}), \mathbf{y} \rangle$. Then ϕ also has Lipschitz continuous gradient of constant L and its optimal point is $\mathbf{y}^* = \mathbf{x}$. Therefore, in view of Lemma B.2.1, we have

$$\begin{aligned} \phi(\mathbf{x}) = \phi(\mathbf{y}^*) &\leq \phi(\mathbf{y} - \frac{1}{L} \nabla \phi(\mathbf{y})) \leq \phi(\mathbf{y}) + \langle \nabla \phi(\mathbf{y}), -\frac{1}{L} \nabla \phi(\mathbf{y}) \rangle + \frac{L}{2} \left\| \frac{1}{L} \nabla \phi(\mathbf{y}) \right\|^2 \\ &= \phi(\mathbf{y}) - \frac{1}{2L} \|\nabla \phi(\mathbf{y})\|^2. \end{aligned}$$

And we get (B.3) since $\nabla \phi(\mathbf{y}) = \nabla g(\mathbf{y}) - \nabla g(\mathbf{x})$. We invoke (B.3) again with \mathbf{x} and \mathbf{y} interchanged, and obtain

$$g(\mathbf{x}) \geq g(\mathbf{y}) + \langle \nabla g(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{1}{2L} \|\nabla g(\mathbf{x}) - \nabla g(\mathbf{y})\|^2.$$

Add this inequality and (B.3) gives (B.4). □

The next lemma concerns the optimality condition at the projection

$$p_L(\mathbf{x}) = \arg \min_{\mathbf{y}} \frac{L}{2} \left\| \mathbf{y} - \left(\mathbf{x} - \frac{1}{L} \nabla g(\mathbf{x}) \right) \right\|^2 + h(\mathbf{y}). \quad (\text{B.5})$$

Lemma B.2.3. *For any $\mathbf{x} \in \mathbb{R}^n$ and $L > 0$, we have $\mathbf{z} = p_L(\mathbf{x})$ if and only if there exist $\gamma(\mathbf{x}) \in \partial h(\mathbf{z})$, the subdifferential of h at \mathbf{z} , such that*

$$\nabla g(\mathbf{x}) + L(\mathbf{z} - \mathbf{x}) + \gamma(\mathbf{x}) = 0. \quad (\text{B.6})$$

Proof. First, notice that the projection step solves a strictly convex objective function and hence the projection is unique. Since h is not necessarily differentiable, we have to

invoke the (necessary and sufficient) optimality condition for non-differentiable convex function. That is, vector $\mathbf{0}$ belongs to the subdifferential at the optimum, i.e.,

$$\mathbf{0} \in L(\mathbf{z} - (\mathbf{x} - \frac{1}{L}\nabla g(\mathbf{x}))) + \partial h(\mathbf{z}),$$

by the calculus of subdifferential. This means there exist $\gamma(\mathbf{x}) \in \partial h(\mathbf{z})$ (the choice of particular subgradient depends on \mathbf{x}) such that,

$$\nabla g(\mathbf{x}) + L(\mathbf{z} - \mathbf{x}) + \gamma(\mathbf{x}) = \mathbf{0}.$$

□

Remark B.2.4. The projection step (B.5) can be equivalently written as

$$p_L(\mathbf{x}) = \arg \min_{\mathbf{y}} g(\mathbf{x}) + \langle \nabla g(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L}{2} \|\mathbf{y} - \mathbf{x}\|^2 + h(\mathbf{y}). \quad (\text{B.7})$$

In view of Lemma B.2.1 and (B.7), we can see that the gradient projection algorithm is in fact a majorization-minimization algorithm, since the right hand side of (B.7) is an upper bound of f .

Remark B.2.5. When h is the indicator function of some convex set C , i.e.,

$$h(\mathbf{x}) = \begin{cases} 0 & : \mathbf{x} \in C, \\ \infty & : \mathbf{x} \notin C, \end{cases}$$

the subdifferential at $\mathbf{x} \in C$ is the normal cone of C at \mathbf{x} (the set of vectors \mathbf{t} such that $\langle \mathbf{t}, \mathbf{y} - \mathbf{x} \rangle \leq 0$ for all $\mathbf{y} \in C$). In this case, the projection step is indeed computing the Euclidean projection of the gradient step onto the convex constraint set:

$$\begin{aligned} p_L(\mathbf{x}) &= \arg \min_{\mathbf{y}} \frac{1}{2} \left\| \mathbf{y} - \left(\mathbf{x} - \frac{1}{L} \nabla g(\mathbf{x}) \right) \right\|^2 \\ \text{s.t. } &\mathbf{x} \in C. \end{aligned}$$

With the previous two lemmas, we are now ready to prove the most important result, which says essentially that every gradient projection step improves the original objective

function $f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$ usually by a non-zero amount.

Lemma B.2.6. *For any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $L \geq L(g)$, we have*

$$f(\mathbf{x}) - f(p_L(\mathbf{y})) \geq \frac{L}{2} \|p_L(\mathbf{y}) - \mathbf{y}\|^2 + L\langle p_L(\mathbf{y}) - \mathbf{y}, \mathbf{y} - \mathbf{x} \rangle. \quad (\text{B.8})$$

Proof. Denote $\mathbf{z} = p_L(\mathbf{y})$. By Lemma B.2.1, we have

$$g(\mathbf{z}) \leq g(\mathbf{y}) + \langle \nabla g(\mathbf{y}), \mathbf{z} - \mathbf{y} \rangle + \frac{L}{2} \|\mathbf{z} - \mathbf{y}\|^2. \quad (\text{B.9})$$

It is noteworthy that we have used the second order upper bound at reference point \mathbf{y} instead of \mathbf{x} , because it will greatly simplify the rest derivation.

By the convexity of h , we have

$$h(\mathbf{z}) \leq h(\mathbf{x}) - \langle \gamma(\mathbf{y}), \mathbf{x} - \mathbf{z} \rangle,$$

where we have chosen the specific subgradient $\gamma(\mathbf{y})$ defined in Lemma B.2.3. Summing the above two inequalities and using (B.6) gives

$$\begin{aligned} f(\mathbf{z}) &= g(\mathbf{z}) + h(\mathbf{z}) \leq g(\mathbf{y}) + \langle \nabla g(\mathbf{y}), \mathbf{z} - \mathbf{y} \rangle + \frac{L}{2} \|\mathbf{z} - \mathbf{y}\|^2 + h(\mathbf{x}) - \langle \gamma(\mathbf{y}), \mathbf{x} - \mathbf{z} \rangle, \\ &= g(\mathbf{y}) + \langle \nabla g(\mathbf{y}), \mathbf{z} - \mathbf{y} \rangle + \frac{L}{2} \|\mathbf{z} - \mathbf{y}\|^2 + h(\mathbf{x}) + \langle \nabla g(\mathbf{y}) + L(\mathbf{z} - \mathbf{y}), \mathbf{x} - \mathbf{z} \rangle \\ &= g(\mathbf{y}) + \langle \nabla g(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + h(\mathbf{x}) + \frac{L}{2} \|\mathbf{z} - \mathbf{y}\|^2 + L\langle \mathbf{z} - \mathbf{y}, \mathbf{x} - \mathbf{z} \rangle. \end{aligned}$$

Therefore,

$$\begin{aligned} f(\mathbf{x}) - f(\mathbf{z}) &\geq g(\mathbf{x}) - g(\mathbf{y}) - \langle \nabla g(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle - \frac{L}{2} \|\mathbf{z} - \mathbf{y}\|^2 + L\langle \mathbf{z} - \mathbf{y}, \mathbf{z} - \mathbf{x} \rangle \\ &\geq 0 - \frac{L}{2} \|\mathbf{z} - \mathbf{y}\|^2 + L\langle \mathbf{z} - \mathbf{y}, \mathbf{z} - \mathbf{y} \rangle + L\langle \mathbf{z} - \mathbf{y}, \mathbf{y} - \mathbf{x} \rangle \quad (\text{B.10}) \\ &= \frac{L}{2} \|\mathbf{z} - \mathbf{y}\|^2 + L\langle \mathbf{z} - \mathbf{y}, \mathbf{y} - \mathbf{x} \rangle, \end{aligned}$$

where we have used the convexity of g in (B.10). \square

Remark B.2.7. Setting $\mathbf{y} = \mathbf{x}$ in Lemma B.2.6 gives

$$f(\mathbf{x}) - f(p_L(\mathbf{x})) \geq \frac{L}{2} \|p_L(\mathbf{x}) - \mathbf{x}\|^2, \quad (\text{B.11})$$

meaning that each gradient projection step decreases the objective function by at least $\frac{L}{2} \|p_L(\mathbf{x}) - \mathbf{x}\|^2$.

We are now fully equipped to prove the convergence rate of gradient projection algorithm.

Theorem B.2.8 (Sublinear convergence rate for weakly convex functions). *Let g be continuously differentiable and ∇g is Lipschitz continuous with constant $L(g)$. For any $L \geq L(g)$, let $\mathbf{x}_{\tau+1} = p_L(\mathbf{x}_\tau)$, $\tau = 0, 1, \dots$, be the sequence generated by the gradient projection algorithm, and \mathbf{x}^* be a minimum of f . Then for any $T \geq 1$, we have*

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \frac{L \|\mathbf{x}_0 - \mathbf{x}^*\|^2}{2T}. \quad (\text{B.12})$$

Proof. Invoking Lemma B.2.6 with $\mathbf{x} = \mathbf{x}^*$ and $\mathbf{y} = \mathbf{x}_\tau$, we obtain

$$\begin{aligned} \frac{2}{L}(f(\mathbf{x}^*) - f(\mathbf{x}_{\tau+1})) &\geq \|\mathbf{x}_{\tau+1} - \mathbf{x}_\tau\|^2 + 2\langle \mathbf{x}_{\tau+1} - \mathbf{x}_\tau, \mathbf{x}_\tau - \mathbf{x}^* \rangle \\ &= \|\mathbf{x}_{\tau+1} - \mathbf{x}^*\|^2 - \|\mathbf{x}_\tau - \mathbf{x}^*\|^2. \end{aligned}$$

Note that $f(\mathbf{x}^*) - f(\mathbf{x}_{\tau+1}) \leq 0$ by definition of \mathbf{x}^* , so the distance between the sequence and a minimum $\|\mathbf{x}_\tau - \mathbf{x}^*\|$ is also non-increasing. Summing the above inequality over $\tau = 0, \dots, T-1$ gives

$$\frac{2}{L} \left(T f(\mathbf{x}^*) - \sum_{\tau=0}^{T-1} f(\mathbf{x}_{\tau+1}) \right) \geq \|\mathbf{x}^* - \mathbf{x}_T\|^2 - \|\mathbf{x}^* - \mathbf{x}_0\|^2. \quad (\text{B.13})$$

In view of (B.11), we have

$$\frac{2}{L}(f(\mathbf{x}_\tau) - f(\mathbf{x}_{\tau+1})) \geq \|\mathbf{x}_\tau - \mathbf{x}_{\tau+1}\|^2.$$

Multiplying this inequality by τ and summing over $\tau = 0, \dots, T-1$, we obtain

$$\frac{2}{L} \sum_{\tau=0}^{T-1} (\tau f(\mathbf{x}_\tau) - (\tau+1)f(\mathbf{x}_{\tau+1}) + f(\mathbf{x}_{\tau+1})) \geq \sum_{\tau=0}^{T-1} \tau \|\mathbf{x}_\tau - \mathbf{x}_{\tau+1}\|^2,$$

which simplifies to

$$\frac{2}{L} \left(-Tf(\mathbf{x}_T) + \sum_{\tau=0}^{T-1} f(\mathbf{x}_{\tau+1}) \right) \geq \sum_{\tau=0}^{T-1} \tau \|\mathbf{x}_\tau - \mathbf{x}_{\tau+1}\|^2. \quad (\text{B.14})$$

Adding (B.13) and (B.14), we have

$$\frac{2T}{L} (f(\mathbf{x}^*) - f(\mathbf{x}_T)) \geq \|\mathbf{x}^* - \mathbf{x}_T\|^2 + \sum_{\tau=0}^{T-1} \tau \|\mathbf{x}_\tau - \mathbf{x}_{\tau+1}\|^2 - \|\mathbf{x}^* - \mathbf{x}_0\|^2 \geq -\|\mathbf{x}^* - \mathbf{x}_0\|^2$$

and hence it follows that

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \frac{L \|\mathbf{x}_0 - \mathbf{x}^*\|^2}{2T}.$$

□

B.3 Strongly convex functions

The sublinear rate from previous section can be greatly improved if f has additional structures. In this section, we consider the case in which g is μ -strongly convex with some parameter $\mu > 0$, i.e.,

$$g(\mathbf{y}) \geq g(\mathbf{x}) + \langle \nabla g(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\mu}{2} \|\mathbf{y} - \mathbf{x}\|^2, \text{ for every } \mathbf{x}, \mathbf{y} \in \mathbb{R}^n. \quad (\text{B.15})$$

In other words, function g is greater than its linear approximation by a quadratic function. Notice this assumption is stronger than the strictly convex condition, because it follows trivially that if $\mathbf{x} \neq \mathbf{y}$, we have

$$g(\mathbf{y}) > g(\mathbf{x}) + \langle \nabla g(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle, \quad (\text{B.16})$$

which defines strictly convexity. Weakly convex function can be considered as having $\mu = 0$. For a twice continuously differentiable function, we can set μ and L to be the smallest and largest eigenvalue of the Hessian matrix, respectively. An immediate consequence of the above assumption is that f is also μ -strongly convex, and is thus strictly convex and has an unique global optimum.

The next lemma shows an important property of the strongly convex function. It is a strengthened version of Corollary B.2.2.

Lemma B.3.1. *Let g be a strongly convex function of parameter μ and ∇g is Lipschitz continuous with constant L . Then for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we have*

$$\langle \nabla g(\mathbf{x}) - \nabla g(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \leq \frac{\mu L}{\mu + L} \|\mathbf{x} - \mathbf{y}\|^2 + \frac{1}{\mu + L} \|\nabla g(\mathbf{x}) - \nabla g(\mathbf{y})\|^2. \quad (\text{B.17})$$

Proof. Consider $\phi(\mathbf{x}) = g(\mathbf{x}) - \frac{1}{2}\mu \|\mathbf{x}\|^2$. Note that $\nabla \phi(\mathbf{x}) = \nabla g(\mathbf{x}) - \mu \mathbf{x}$. This function is convex, because for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ we have

$$\phi(\mathbf{y}) - \phi(\mathbf{x}) - \langle \nabla \phi(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle = g(\mathbf{y}) - g(\mathbf{x}) - \langle \nabla g(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle - \frac{\mu}{2} \|\mathbf{y} - \mathbf{x}\|^2 \geq 0,$$

by (B.15). Similarly, we can see that

$$\phi(\mathbf{y}) - \phi(\mathbf{x}) - \langle \nabla \phi(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \frac{1}{2}(L - \mu) \|\mathbf{x} - \mathbf{y}\|^2.$$

Therefore we can apply Corollary B.2.2 to ϕ and obtain

$$\langle \nabla \phi(\mathbf{x}) - \nabla \phi(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \leq \frac{1}{L - \mu} \|\nabla \phi(\mathbf{x}) - \nabla \phi(\mathbf{y})\|$$

and this inequality can be rewritten as (B.17). □

We will show that for strongly convex functions, the gradient projection algorithm can achieve linear convergence rate, which is theoretically much better than the sublinear rate since it implies the error decreases at a geometric rate at every iteration. This result is *not* given in Beck and Teboulle (2009) since they are not concerned with strongly

convex functions. The proof given below is based on an online note of Benjamin Recht¹ which, however, places stronger assumption on g (it assumes that g has second order derivatives which is not required here).

We will need the following lemma which concerns the nonexpansive property of the projection step.

Lemma B.3.2. (*Nonexpansiveness of projection*) For any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $L > 0$, we have

$$\|p_L(\mathbf{x}) - p_L(\mathbf{y})\| \leq \left\| \left(\mathbf{x} - \frac{1}{L} \nabla g(\mathbf{x}) \right) - \left(\mathbf{y} - \frac{1}{L} \nabla g(\mathbf{y}) \right) \right\|. \quad (\text{B.18})$$

Proof. We apply Lemma B.2.3 to \mathbf{x} and \mathbf{y} , and obtain

$$\begin{aligned} \mathbf{x} - \frac{1}{L} \nabla g(\mathbf{x}) &= p_L(\mathbf{x}) + \frac{1}{L} \gamma(\mathbf{x}), \\ \mathbf{y} - \frac{1}{L} \nabla g(\mathbf{y}) &= p_L(\mathbf{y}) + \frac{1}{L} \gamma(\mathbf{y}). \end{aligned}$$

As a result,

$$\begin{aligned} \left\| \left(\mathbf{x} - \frac{1}{L} \nabla g(\mathbf{x}) \right) - \left(\mathbf{y} - \frac{1}{L} \nabla g(\mathbf{y}) \right) \right\|^2 &= \left\| \frac{1}{L} (\gamma(\mathbf{x}) - \gamma(\mathbf{y})) + (p_L(\mathbf{x}) - p_L(\mathbf{y})) \right\|^2 \\ &= \frac{1}{L^2} \|\gamma(\mathbf{x}) - \gamma(\mathbf{y})\|^2 + \frac{2}{L} \langle \gamma(\mathbf{x}) - \gamma(\mathbf{y}), p_L(\mathbf{x}) - p_L(\mathbf{y}) \rangle + \|p_L(\mathbf{x}) - p_L(\mathbf{y})\|^2 \end{aligned} \quad (\text{B.19})$$

Notice $\gamma(\mathbf{x})$ and $\gamma(\mathbf{y})$ are subgradients of h at $p_L(\mathbf{x})$ and $p_L(\mathbf{y})$ respectively. By definition of subgradient, we have

$$\begin{aligned} h(p_L(\mathbf{y})) - h(p_L(\mathbf{x})) &\geq \langle \gamma(\mathbf{x}), p_L(\mathbf{y}) - p_L(\mathbf{x}) \rangle, \\ h(p_L(\mathbf{x})) - h(p_L(\mathbf{y})) &\geq \langle \gamma(\mathbf{y}), p_L(\mathbf{x}) - p_L(\mathbf{y}) \rangle. \end{aligned}$$

Adding these two inequalities gives

$$\langle \gamma(\mathbf{x}) - \gamma(\mathbf{y}), p_L(\mathbf{x}) - p_L(\mathbf{y}) \rangle \geq 0.$$

¹<http://pages.cs.wisc.edu/~brecht/cs726docs/ProjectedGradientMethods.pdf>

In view of this inequality, it follows from (B.19) that

$$\left\| \left(\mathbf{x} - \frac{1}{L} \nabla g(\mathbf{x}) \right) - \left(\mathbf{y} - \frac{1}{L} \nabla g(\mathbf{y}) \right) \right\| \geq \|p_L(\mathbf{x}) - p_L(\mathbf{y})\|.$$

□

Theorem B.3.3 (Linear convergence rate for strongly convex functions). *Let g be strongly convex with parameter μ and ∇g is Lipschitz continuous with constant L . Let $\mathbf{x}_{\tau+1} = p_{\frac{L+\mu}{2}}(\mathbf{x}_\tau)$, $\tau = 0, 1, \dots$, be the sequence generated by the gradient projection algorithm, and \mathbf{x}^* is the minimum of f . Then for any $T \geq 1$, we have*

$$\|\mathbf{x}_T - \mathbf{x}^*\| \leq \left(\frac{\kappa - 1}{\kappa + 1} \right)^T \|\mathbf{x}_0 - \mathbf{x}^*\|, \quad (\text{B.20})$$

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \frac{L}{2} \left(\frac{\kappa - 1}{\kappa + 1} \right)^{2T} \|\mathbf{x}_0 - \mathbf{x}^*\|^2, \quad (\text{B.21})$$

where $\kappa = L/\mu$.

Proof. The proof technique here is somewhat different from Theorem B.2.8, and the main reason is that we are now using $\nu \leq L$ and thus Lemma B.2.6 does not apply.

We first assume that the projection uses an arbitrary inverse stepsize $\nu > 0$. We assert that $\mathbf{x}^* = p_\nu(\mathbf{x}^*)$, i.e., the global optimum of f is a fixed point under the projection. This is because the optimality of \mathbf{x}^* implies that

$$\mathbf{0} = \nabla g(\mathbf{x}^*) + \gamma,$$

where γ is some subgradient of h at \mathbf{x}^* . This equality can be written as

$$\mathbf{0} = \nabla g(\mathbf{x}^*) + \nu(\mathbf{x}^* - \mathbf{x}^*) + \gamma.$$

In view of Lemma B.2.3, we see that $\mathbf{x}^* = p_\nu(\mathbf{x}^*)$.

We now make use of the nonexpansive property of projection, and obtain

$$\begin{aligned} \|\mathbf{x}_{\tau+1} - \mathbf{x}^*\| &= \|p_\nu(\mathbf{x}_\tau) - p_\nu(\mathbf{x}^*)\| \leq \left\| \left(\mathbf{x}_\tau - \frac{1}{\nu} \nabla g(\mathbf{x}_\tau) \right) - \left(\mathbf{x}^* - \frac{1}{\nu} \nabla g(\mathbf{x}^*) \right) \right\| \\ &= \|\mathbf{x}_\tau - \mathbf{x}^*\|^2 - \frac{2}{\nu} \langle \mathbf{x}_\tau - \mathbf{x}^*, \nabla g(\mathbf{x}_\tau) - \nabla g(\mathbf{x}^*) \rangle + \frac{1}{\nu^2} \|\nabla g(\mathbf{x}_\tau) - \nabla g(\mathbf{x}^*)\|^2. \end{aligned}$$

In view of Lemma B.3.1 and the Lipschitz continuity of ∇g , it follows that

$$\|\mathbf{x}_{\tau+1} - \mathbf{x}^*\|^2 \leq \left(1 - \frac{2\mu L}{\nu(\mu + L)}\right) \|\mathbf{x}_\tau - \mathbf{x}^*\|^2 + \frac{1}{\nu} \left(\frac{1}{\nu} - \frac{2}{\mu + L}\right) \|\nabla g(\mathbf{x}_\tau) - \nabla g(\mathbf{x}^*)\|^2.$$

It is easy to see that the second term vanishes at $\nu = \frac{L+\mu}{2}$ and we are left with $\|\mathbf{x}_{\tau+1} - \mathbf{x}^*\| \leq \left(\frac{\kappa-1}{\kappa+1}\right) \|\mathbf{x}_\tau - \mathbf{x}^*\|$. We apply this relation recursively for $\tau = T-1, \dots, 0$ and obtain (B.20).

To prove convergence in function value, we simply note that

$$f(\mathbf{x}_T) \leq f(\mathbf{x}^*) + \frac{L}{2} \|\mathbf{x}_T - \mathbf{x}^*\|^2$$

(we have used Lemma B.2.1 and the fact that $\mathbf{0}$ is in the subdifferential of $f(\mathbf{x}^*)$). \square

Remark B.3.4. In the case of our Laplacian K -modes model (Chapter 5), μ and L are relatively easy to set, since they correspond to the extrema of eigenvalues of the sparse Laplacian matrix. When L is not known, we could apply backtracking line search to find an estimate of it and still guarantee convergence of the algorithm (see Beck and Teboulle, 2009 for the procedure), and the proof of this approach is only slightly different from the one presented here. On the other hand, it is straight-forward and natural to have strongly convex objective function in machine learning problems, by adding regularizations.

Remark B.3.5. Despite its theoretically better convergence rate in Theorem B.3.3, the practical performance of the algorithm may not be satisfying, if $\kappa = \frac{L}{\mu}$ is large and the geometric rate $\left(\frac{\kappa-1}{\kappa+1}\right)^2$ is close to 1. In some sense, κ measures the difficulty of the convex problem for first order methods, and thus is sometimes referred to as the problem's *condition number*.

Algorithm 4 Accelerated gradient projection algorithm for weakly convex g .

Input: L =Lipschitz constant of ∇g .

- 1: Set $\mathbf{y}_1 = \mathbf{x}_0, t_1 = 1$.
- 2: **repeat**
- 3: $\mathbf{x}_\tau = p_L(\mathbf{y}_\tau)$,
- 4: $t_{\tau+1} = \frac{1+\sqrt{1+4t_\tau^2}}{2}$,
- 5: $\mathbf{y}_{\tau+1} = \mathbf{x}_\tau + \left(\frac{t_\tau-1}{t_{\tau+1}}\right)(\mathbf{x}_\tau - \mathbf{x}_{\tau-1})$,
- 6: $\tau = \tau + 1$,
- 7: **until** convergence.

Output: \mathbf{x}_τ is the solution.

B.4 Acceleration scheme

The accelerated gradient projection algorithm is based on the work of Nesterov in early 1980s, and has been a hot topic recently to speedup first order methods (Beck and Teboulle, 2009; Hu et al., 2009; Ji and Ye, 2009; Nesterov, 2005, 2007; Rakhlin et al., 2012). The idea of the algorithm is to maintain an auxiliary sequence, taking into account the information from previous steps. The updating formula for this sequence is somewhat similar to that of conjugate gradient (Nocedal and Wright, 2006) and the momentum technique (Bishop, 1995) typically used in neural network training.

B.4.1 Weakly convex function

Nesterov’s original proof for the acceleration scheme is based on the notion of *estimate sequence* (Nesterov, 2004). For weakly convex functions, we follow the proof of Beck and Teboulle (2009) instead, which is formally simpler. For the ease of discussion, we abstract the accelerated gradient projection algorithm in Algorithm 4.

Note the difference between Algorithm 4 and Algorithm 3. Instead of projecting the previous estimate \mathbf{x}_τ , we are now projecting \mathbf{y}_τ , which is a specific combination of previous two estimates \mathbf{x}_τ and $\mathbf{x}_{\tau-1}$.

Analogous to Lemma B.2.6, we first derive an estimate of progress for the projection step. The difference, however, is that we now have to consider two consecutive steps

together.

Lemma B.4.1. *The sequences $\{\mathbf{x}_\tau\}_{\tau=0}^\infty$ and $\{\mathbf{y}_\tau\}_{\tau=1}^\infty$ generated via Algorithm 4 satisfy for every $\tau \geq 1$*

$$\frac{2}{L}(t_\tau^2 v_\tau - t_{\tau+1}^2 v_{\tau+1}) \geq \|\mathbf{u}_{\tau+1}\|^2 - \|\mathbf{u}_\tau\|^2, \quad (\text{B.22})$$

where $v_\tau = f(\mathbf{x}_\tau) - f(\mathbf{x}^*)$, $\mathbf{u}_\tau = t_\tau \mathbf{x}_\tau - (t_\tau - 1)\mathbf{x}_{\tau-1} - \mathbf{x}^*$.

Proof. We first apply Lemma B.2.6 at $(\mathbf{x} = \mathbf{x}_\tau, \mathbf{y} = \mathbf{y}_{\tau+1})$ and likewise $(\mathbf{x} = \mathbf{x}^*, \mathbf{y} = \mathbf{y}_{\tau+1})$, and get

$$\begin{aligned} \frac{2}{L}(v_\tau - v_{\tau+1}) &\geq \|\mathbf{x}_{\tau+1} - \mathbf{y}_{\tau+1}\|^2 + 2\langle \mathbf{x}_{\tau+1} - \mathbf{y}_{\tau+1}, \mathbf{y}_{\tau+1} - \mathbf{x}_\tau \rangle, \\ \frac{2}{L}(0 - v_{\tau+1}) &\geq \|\mathbf{x}_{\tau+1} - \mathbf{y}_{\tau+1}\|^2 + 2\langle \mathbf{x}_{\tau+1} - \mathbf{y}_{\tau+1}, \mathbf{y}_{\tau+1} - \mathbf{x}^* \rangle, \end{aligned}$$

where we have used the fact that $\mathbf{x}_{\tau+1} = p_L(\mathbf{y}_{\tau+1})$. Multiplying the first inequality above by $(t_{\tau+1} - 1)$ and adding it to the second inequality gives

$$\begin{aligned} \frac{2}{L}((t_{\tau+1} - 1)v_\tau - t_{\tau+1}v_{\tau+1}) &\geq t_{\tau+1} \|\mathbf{x}_{\tau+1} - \mathbf{y}_{\tau+1}\|^2 \\ &\quad + 2\langle \mathbf{x}_{\tau+1} - \mathbf{y}_{\tau+1}, t_{\tau+1}\mathbf{y}_{\tau+1} - (t_{\tau+1} - 1)\mathbf{x}_\tau - \mathbf{x}^* \rangle. \end{aligned}$$

Multiplying the last inequality by $t_{\tau+1}$ and using the relation $t_\tau^2 = t_{\tau+1}^2 - t_{\tau+1}$, we obtain

$$\begin{aligned} \frac{2}{L}(t_\tau^2 v_\tau - t_{\tau+1}^2 v_{\tau+1}) &\geq \|t_{\tau+1}(\mathbf{x}_{\tau+1} - \mathbf{y}_{\tau+1})\|^2 \\ &\quad + 2t_{\tau+1}\langle \mathbf{x}_{\tau+1} - \mathbf{y}_{\tau+1}, t_{\tau+1}\mathbf{y}_{\tau+1} - (t_{\tau+1} - 1)\mathbf{x}_\tau - \mathbf{x}^* \rangle. \end{aligned}$$

Applying the usual Pythagoras relation

$$\|\mathbf{b} - \mathbf{a}\|^2 + 2\langle \mathbf{b} - \mathbf{a}, \mathbf{a} - \mathbf{c} \rangle = \|\mathbf{b} - \mathbf{c}\|^2 - \|\mathbf{a} - \mathbf{c}\|^2$$

to the right-hand side of the last inequality with

$$\mathbf{a} = t_{\tau+1}\mathbf{y}_{\tau+1}, \quad \mathbf{b} = t_{\tau+1}\mathbf{x}_{\tau+1}, \quad \mathbf{c} = (t_{\tau+1} - 1)\mathbf{x}_\tau + \mathbf{x}^*,$$

we thus get

$$\begin{aligned} \frac{2}{L}(t_\tau^2 v_\tau - t_{\tau+1}^2 v_{\tau+1}) &\geq \|t_{\tau+1} \mathbf{x}_{\tau+1} - (t_{\tau+1} - 1) \mathbf{x}_\tau - \mathbf{x}^*\|^2 \\ &\quad - \|t_{\tau+1} \mathbf{y}_{\tau+1} - (t_{\tau+1} - 1) \mathbf{x}_\tau - \mathbf{x}^*\|^2. \end{aligned}$$

Therefore, with $\mathbf{y}_{\tau+1}$ and \mathbf{u}_τ defined by

$$t_{\tau+1} \mathbf{y}_{\tau+1} = t_{\tau+1} \mathbf{x}_\tau + (t_\tau - 1)(\mathbf{x}_\tau - \mathbf{x}_{\tau-1}), \quad \text{and} \quad \mathbf{u}_\tau = t_\tau \mathbf{x}_\tau - (t_\tau - 1) \mathbf{x}_{\tau-1} - \mathbf{x}^*,$$

it follows that

$$\frac{2}{L}(t_\tau^2 v_\tau - t_{\tau+1}^2 v_{\tau+1}) \geq \|\mathbf{u}_{\tau+1}\|^2 - \|\mathbf{u}_\tau\|^2.$$

□

Remark B.4.2. The convergence of the algorithm now depends on the asymptotic behavior of the $\{t_\tau\}_{\tau=0}^\infty$ sequence. Using the recursive definition of the sequence, it is easy to prove by induction that $t_\tau \geq \frac{\tau+1}{2}$.

Theorem B.4.3 (Improved sublinear convergence rate for weakly convex functions).

Let g be continuously differentiable and ∇g is Lipschitz continuous with constant L . Let $\{\mathbf{x}_\tau\}_{\tau=0}^\infty$ be generated via the accelerated gradient projection algorithm. Then for any $T \geq 1$, we have

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \frac{2L \|\mathbf{x}_0 - \mathbf{x}^*\|^2}{(T+1)^2}. \quad (\text{B.23})$$

Proof. Lemma B.4.1 shows that the sequence of $\{\frac{2}{L}t_\tau^2 v_\tau + \|\mathbf{u}_\tau\|^2\}$ is non-increasing over iterations. Now we consider the case of $\tau = 1$. Since $t_1 = 1$, we have

$$\frac{2}{L}t_1^2 v_1 = \frac{2}{L}(f(\mathbf{x}_1) - f(\mathbf{x}^*)) \quad \text{and} \quad \|\mathbf{u}_1\|^2 = \|\mathbf{x}_1 - \mathbf{x}^*\|^2. \quad (\text{B.24})$$

Moreover, in view of Lemma B.2.6, we have

$$f(\mathbf{x}^*) - f(p_L(\mathbf{y}_1)) \geq \frac{L}{2} \|p_L(\mathbf{y}_1) - \mathbf{y}_1\|^2 + L \langle \mathbf{y}_1 - \mathbf{x}^*, p_L(\mathbf{y}_1) - \mathbf{y}_1 \rangle.$$

Thus,

$$\begin{aligned}
f(\mathbf{x}^*) - f(\mathbf{x}_1) &= f(\mathbf{x}^*) - f(p_L(\mathbf{y}_1)) \\
&\geq \frac{L}{2} \|p_L(\mathbf{y}_1) - \mathbf{y}_1\|^2 + L\langle \mathbf{y}_1 - \mathbf{x}^*, p_L(\mathbf{y}_1) - \mathbf{y}_1 \rangle \\
&= \frac{L}{2} \|\mathbf{x}_1 - \mathbf{y}_1\|^2 + L\langle \mathbf{y}_1 - \mathbf{x}^*, \mathbf{x}_1 - \mathbf{y}_1 \rangle \\
&= \frac{L}{2} (\|\mathbf{x}_1 - \mathbf{x}^*\|^2 - \|\mathbf{y}_1 - \mathbf{x}^*\|^2).
\end{aligned}$$

Consequently, combining this inequality with (B.24) gives

$$\frac{2}{L} t_1^2 v_1 + \|\mathbf{u}_1\|^2 \leq -\frac{2}{L} \cdot \frac{L}{2} (\|\mathbf{x}_1 - \mathbf{x}^*\|^2 - \|\mathbf{y}_1 - \mathbf{x}^*\|^2) + \|\mathbf{x}_1 - \mathbf{x}^*\|^2 = \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

(we have used the fact that $\mathbf{y}_1 = \mathbf{x}_0$ in the last step). Using the non-increasing property of the $\{\frac{2}{L} t_\tau^2 v_\tau + \|\mathbf{u}_\tau\|^2\}$ sequence, we have

$$\frac{2}{L} t_T^2 v_T \leq \frac{2}{L} t_T^2 v_T + \|\mathbf{u}_T\|^2 \leq \frac{2}{L} t_1^2 v_1 + \|\mathbf{u}_1\|^2 = \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

Therefore, we obtain the desired inequality by noting that $t_T \geq \frac{T+1}{2}$. \square

Remark B.4.4. The above sublinear convergence rate is considered *optimal* for general non-smooth convex functions in the sense of Nemirovski and Yudin (1983) (consider the case of $g \equiv 0$ and we are left with the non-smooth part h).

B.4.2 Strongly convex functions

This section concerns the convergence rate of accelerated gradient projection algorithm for strongly convex functions. The procedure is given in Algorithm 5. Notice it is different from Algorithm 4 in providing the stepsize sequence $\{\alpha_\tau\}_{\tau=0}^\infty$, and the update formula for the auxiliary sequence $\{\mathbf{y}_\tau\}_{\tau=1}^\infty$.

For strongly convex functions, Lemma B.2.6 can be strengthened.

Lemma B.4.5. *Let g be μ -strongly convex and ∇g is Lipschitz continuous with constant*

Algorithm 5 Accelerated gradient projection algorithm for strongly convex g .

Input: L =Lipschitz constant of ∇g , μ =strongly convex parameter of g .

- 1: Set $\mathbf{y}_1 = \mathbf{x}_0$.
- 2: **repeat**
- 3: $\mathbf{x}_\tau = p_L(\mathbf{y}_\tau)$,
- 4: Solve $\alpha_\tau^2 = (1 - \alpha_\tau)\alpha_{\tau-1}^2 + \frac{\mu}{L}\alpha_\tau$ for α_τ .
- 5: $\mathbf{y}_{\tau+1} = \mathbf{x}_\tau + \frac{\alpha_{\tau-1}(1-\alpha_{\tau-1})}{\alpha_{\tau-1}^2 + \alpha_\tau}(\mathbf{x}_\tau - \mathbf{x}_{\tau-1})$,
- 6: $\tau = \tau + 1$,
- 7: **until** convergence.

Output: \mathbf{x}_τ is the solution.

L. Then for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we have

$$f(\mathbf{x}) - f(p_L(\mathbf{y})) \geq \frac{\mu}{2} \|\mathbf{x} - \mathbf{y}\|^2 + \frac{L}{2} \|p_L(\mathbf{y}) - \mathbf{y}\|^2 + L\langle p_L(\mathbf{y}) - \mathbf{y}, \mathbf{y} - \mathbf{x} \rangle. \quad (\text{B.25})$$

Proof. We simply note that we can now replace $g(\mathbf{x}) - g(\mathbf{y}) - \langle \nabla g(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \geq 0$ with $g(\mathbf{x}) - g(\mathbf{y}) - \langle \nabla g(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \geq \frac{\mu}{2} \|\mathbf{x} - \mathbf{y}\|^2$ in (B.10). \square

Theorem B.4.6 (Improved linear convergence rate for strongly convex functions). *Let g be μ -strongly convex and ∇g is Lipschitz continuous with constant L . Let $\{\mathbf{x}_\tau\}_{\tau=0}^\infty$ be the solution estimated by Algorithm 5. Then for any $T \geq 1$, we have*

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq L \left(1 - \sqrt{\frac{\mu}{L}}\right)^T \|\mathbf{x}^* - \mathbf{x}_0\|^2.$$

Proof. We construct the following sequence of *quadratic* functions:

$$\begin{aligned} \phi_0(\mathbf{x}) &= f(\mathbf{x}_0) + \frac{\xi_0}{2} \|\mathbf{x} - \mathbf{x}_0\|^2, \\ \phi_{\tau+1}(\mathbf{x}) &= (1 - \alpha_\tau)\phi_\tau(\mathbf{x}) + \alpha_\tau[f(p_L(\mathbf{y}_{\tau+1})) + \frac{L}{2} \|p_L(\mathbf{y}_{\tau+1}) - \mathbf{y}_{\tau+1}\|^2 \\ &\quad + L\langle p_L(\mathbf{y}_{\tau+1}) - \mathbf{y}_{\tau+1}, \mathbf{y}_{\tau+1} - \mathbf{x} \rangle + \frac{\mu}{2} \|\mathbf{x} - \mathbf{y}_{\tau+1}\|^2] \quad \text{for } \tau = 0, 1, \dots, \end{aligned}$$

where $\alpha_\tau \in (0, 1)$ for $\tau = 1, 2, \dots$. We can write these functions equivalently in the

form of $\phi_\tau(\mathbf{x}) = \frac{\xi_\tau}{2} \|\mathbf{x} - \mathbf{v}_\tau\|^2 + \phi_\tau^*$ where $\phi_\tau^* = \min_{\mathbf{x}} \phi_\tau(\mathbf{x})$ as:

$$\xi_0 \geq 0, \quad \mathbf{v}_0 = \mathbf{x}_0, \quad \phi_0^* = f(\mathbf{x}_0), \quad (\text{B.26})$$

$$\xi_{\tau+1} = (1 - \alpha_\tau)\xi_\tau + \alpha_\tau\mu, \quad (\text{B.27})$$

$$\mathbf{v}_{\tau+1} = \frac{(1 - \alpha_\tau)\xi_\tau\mathbf{v}_\tau + \alpha_\tau(L(p_L(\mathbf{y}_{\tau+1}) - \mathbf{y}_{\tau+1}) + \mu\mathbf{y}_{\tau+1})}{\xi_{\tau+1}}, \quad (\text{B.28})$$

$$\begin{aligned} \phi_{\tau+1}^* &= (1 - \alpha_\tau)\phi_\tau^* + \alpha_\tau \left[f(p_L(\mathbf{y}_{\tau+1})) + \frac{L}{2} \|\mathbf{y}_{\tau+1} - p_L(\mathbf{y}_{\tau+1})\|^2 \right] \\ &\quad - \frac{\alpha_\tau^2 L^2}{2\xi_{\tau+1}} \|\mathbf{y}_{\tau+1} - p_L(\mathbf{y}_{\tau+1})\|^2 \\ &\quad + \frac{\alpha_\tau(1 - \alpha_\tau)\xi_\tau}{\xi_{\tau+1}} \left[\frac{\mu}{2} \|\mathbf{y}_{\tau+1} - \mathbf{v}_\tau\|^2 + L\langle \mathbf{y}_{\tau+1} - p_L(\mathbf{y}_{\tau+1}), \mathbf{v}_\tau - \mathbf{y}_{\tau+1} \rangle \right] \end{aligned} \quad (\text{B.29})$$

We now prove some properties of this sequence recursively.

Property 1. For $\tau = 0, 1, \dots$, we have $f(\mathbf{x}_\tau) \leq \phi_\tau^*$.

The case for $\tau = 0$ is trivial. Now assume $f(\mathbf{x}_\tau) \leq \phi_\tau^*$ holds. Invoking Lemma B.4.5 with $(\mathbf{x} = \mathbf{x}_\tau, \mathbf{y} = \mathbf{y}_{\tau+1})$ gives

$$f(\mathbf{x}_\tau) \geq f(\mathbf{x}_{\tau+1}) + L\langle \mathbf{y}_{\tau+1} - \mathbf{x}_{\tau+1}, \mathbf{x}_\tau - \mathbf{y}_{\tau+1} \rangle + \frac{L}{2} \|\mathbf{y}_{\tau+1} - \mathbf{x}_{\tau+1}\|^2 + \frac{\mu}{2} \|\mathbf{x}_\tau - \mathbf{y}_{\tau+1}\|^2.$$

Then in view of (B.29), we have

$$\begin{aligned} \phi_{\tau+1}^* &\geq (1 - \alpha_\tau)f(\mathbf{x}_{\tau+1}) + (1 - \alpha_\tau)L\langle \mathbf{y}_{\tau+1} - \mathbf{x}_{\tau+1}, \mathbf{x}_\tau - \mathbf{y}_{\tau+1} \rangle \\ &\quad + \frac{(1 - \alpha_\tau)L}{2} \|\mathbf{y}_{\tau+1} - \mathbf{x}_{\tau+1}\|^2 + \frac{(1 - \alpha_\tau)\mu}{2} \|\mathbf{x}_\tau - \mathbf{y}_{\tau+1}\|^2 \\ &\quad + \alpha_\tau \left[f(\mathbf{x}_{\tau+1}) + \frac{L}{2} \|\mathbf{y}_{\tau+1} - \mathbf{x}_{\tau+1}\|^2 \right] - \frac{\alpha_\tau^2 L^2}{2\xi_{\tau+1}} \|\mathbf{y}_{\tau+1} - \mathbf{x}_{\tau+1}\|^2 \\ &\quad + \frac{\alpha_\tau(1 - \alpha_\tau)\xi_\tau}{\xi_{\tau+1}} \left[\frac{\mu}{2} \|\mathbf{y}_{\tau+1} - \mathbf{v}_\tau\|^2 + L\langle \mathbf{y}_{\tau+1} - \mathbf{x}_{\tau+1}, \mathbf{v}_\tau - \mathbf{y}_{\tau+1} \rangle \right] \\ &\geq f(\mathbf{x}_{\tau+1}) + \left(\frac{L}{2} - \frac{\alpha_\tau^2 L^2}{2\xi_{\tau+1}} \right) \|\mathbf{y}_{\tau+1} - \mathbf{x}_{\tau+1}\|^2 \\ &\quad + (1 - \alpha_\tau)L\langle \mathbf{y}_{\tau+1} - \mathbf{x}_{\tau+1}, \mathbf{x}_\tau - \mathbf{y}_{\tau+1} \rangle + \frac{\alpha_\tau\xi_\tau}{\xi_{\tau+1}} \langle \mathbf{v}_\tau - \mathbf{y}_{\tau+1} \rangle, \end{aligned} \quad (\text{B.30})$$

where we have dropped squared terms of $\|\mathbf{x}_\tau - \mathbf{y}_{\tau+1}\|$ and $\|\mathbf{v}_\tau - \mathbf{y}_{\tau+1}\|$ to obtain the last inequality. Thus, we can choose

$$\xi_{\tau+1} = (1 - \alpha_\tau)\xi_\tau + \alpha_\tau\mu = L\alpha_\tau^2, \quad (\text{B.31})$$

$$\mathbf{y}_{\tau+1} = \frac{1}{\xi_\tau + \alpha_\tau\mu}(\alpha_\tau\xi_\tau\mathbf{v}_\tau + \xi_{\tau+1}\mathbf{x}_\tau), \quad (\text{B.32})$$

to ensure the second term and the third terms vanish in (B.30) and we are left with $\phi_{\tau+1}^* \geq f(\mathbf{x}_{\tau+1})$. We can ensure (B.31) recursively as follows: we set

$$\xi_0 \geq \mu, \quad \text{and} \quad \xi_1 = (1 - \alpha_0)\xi_0 + \alpha_0\mu = L\alpha_0^2,$$

and update α_τ such that for $\tau = 1, 2, \dots$,

$$\xi_{\tau+1} = (1 - \alpha_\tau)\xi_\tau + \alpha_\tau\mu = (1 - \alpha_\tau)L\alpha_{\tau-1}^2 + \alpha_\tau\mu = L\alpha_\tau^2,$$

or equivalently

$$\alpha_\tau^2 = (1 - \alpha_\tau)\alpha_{\tau-1}^2 + \frac{\mu}{L}\alpha_\tau.$$

By substituting (B.32) into (B.28), we get

$$\begin{aligned} \mathbf{v}_{\tau+1} &= \frac{1}{\xi_{\tau+1}} \left[\frac{1 - \alpha_\tau}{\alpha_\tau} (-\xi_{\tau+1}\mathbf{x}_\tau + (\xi_\tau + \alpha_\tau\mu)\mathbf{y}_{\tau+1}) + \alpha_\tau(L(\mathbf{x}_{\tau+1} - \mathbf{y}_{\tau+1}) + \mu\mathbf{y}_{\tau+1}) \right] \\ &= \frac{\alpha_\tau - 1}{\alpha_\tau} \mathbf{x}_\tau + \frac{\alpha_\tau L}{\xi_{\tau+1}} \mathbf{x}_{\tau+1} + \left(\frac{1}{\alpha_\tau} - \frac{\alpha_\tau L}{\xi_{\tau+1}} \right) \mathbf{y}_{\tau+1} \\ &= \mathbf{x}_\tau + \frac{1}{\alpha_\tau} (\mathbf{x}_{\tau+1} - \mathbf{x}_\tau), \end{aligned}$$

where we have used $\xi_{\tau+1} = L\alpha_\tau^2$ in the last step. By substituting this relation into (B.32), we have

$$\mathbf{y}_{\tau+1} = \mathbf{x}_\tau + \frac{\alpha_{\tau-1}(1 - \alpha_{\tau-1})}{\alpha_{\tau-1}^2 + \alpha_\tau} (\mathbf{x}_\tau - \mathbf{x}_{\tau-1}).$$

So far, we have explained the construction of the sequence $\{\mathbf{x}_\tau\}_{\tau=0}^\infty$ and $\{\mathbf{y}_\tau\}_{\tau=1}^\infty$ (the sequence $\{\xi_\tau\}_{\tau=0}^\infty$ and $\{\mathbf{v}_\tau\}_{\tau=0}^\infty$ have been effectively eliminated) and established the property that $f(\mathbf{x}_\tau) \leq \phi_\tau^*$ for $\tau = 1, 2, \dots$. We now show the error bound provided by

sequence $\phi(\mathbf{x})$.

Property 2. Define $\{\lambda_\tau\}_{\tau=0}^\infty$ recursively as $\lambda_0 = 1$, and $\lambda_{\tau+1} = (1 - \alpha_\tau)\lambda_\tau$ for $\tau = 0, 1, \dots$. Then we have

$$\phi_\tau(\mathbf{x}) \leq (1 - \lambda_\tau)f(\mathbf{x}) + \lambda_\tau\phi_0(\mathbf{x}), \quad \text{for } \tau = 0, 1, \dots$$

To see this, we just need to notice the recursive definition of $\phi_\tau(\mathbf{x})$ satisfies

$$\begin{aligned} \phi_{\tau+1}(\mathbf{x}) &\leq (1 - \alpha_\tau)\phi_\tau(\mathbf{x}) + \alpha_\tau f(\mathbf{x}) \\ &\leq (1 - \alpha_\tau)[(1 - \lambda_\tau)f(\mathbf{x}) + \lambda_\tau\phi_0(\mathbf{x})] + \alpha_\tau f(\mathbf{x}) \\ &= (1 - (1 - \alpha_\tau)\lambda_\tau)f(\mathbf{x}) + (1 - \alpha_\tau)\lambda_\tau\phi_0(\mathbf{x}). \end{aligned}$$

This property is the defining property of *estimate sequence*. Clearly, we have $\lambda_T = \prod_{\tau=0}^{T-1} \alpha_\tau \rightarrow 0$ as $T \rightarrow \infty$.

Property 3. For $\tau = 0, 1, \dots$, we have $f(\mathbf{x}_\tau) - f(\mathbf{x}^*) \leq \lambda_\tau[\phi_0(\mathbf{x}^*) - f(\mathbf{x}^*)] \rightarrow 0$.

This is a straight-forward consequence of Property 1 and Property 2:

$$f(\mathbf{x}_\tau) \leq \phi_\tau^* = \min_{\mathbf{x}} \phi_\tau(\mathbf{x}) \leq \min_{\mathbf{x}} [(1 - \lambda_\tau)f(\mathbf{x}) + \lambda_\tau\phi_0(\mathbf{x})] \leq (1 - \lambda_\tau)f(\mathbf{x}^*) + \lambda_\tau\phi_0(\mathbf{x}^*).$$

With the above properties, we are able to estimate the convergence speed of $f(\mathbf{x}_\tau)$ using the convergence speed of the $\{\lambda_\tau\}$ sequence.

Property 4. For $\tau = 0, 1, \dots$, we have $\lambda_\tau \leq (1 - \sqrt{\frac{\mu}{L}})^\tau$.

Notice we have set $\xi_0 \geq \mu$. Assume $\xi_\tau \geq \mu$, then by induction

$$\xi_{\tau+1} = L\alpha_\tau^2 = (1 - \alpha_\tau)\xi_\tau + \alpha_\tau\mu \geq \mu.$$

Therefore we have $\xi_\tau \geq \mu$ and $\alpha_\tau \geq \sqrt{\frac{\mu}{L}}$ for $\tau = 0, 1, \dots$

We can now prove the desired convergence rate. Using Lemma B.2.1 at $(\mathbf{x} = \mathbf{x}_0, \mathbf{y} = \mathbf{x}^*)$, we have

$$f(\mathbf{x}_0) \leq f(\mathbf{x}^*) + \frac{L}{2} \|\mathbf{x}^* - \mathbf{x}_0\|^2.$$

Therefore,

$$\begin{aligned} \phi_0(\mathbf{x}^*) &= f(\mathbf{x}_0) + \frac{L}{2} \|\mathbf{x}^* - \mathbf{x}_0\|^2 \\ &\leq f(\mathbf{x}^*) + \frac{L}{2} \|\mathbf{x}^* - \mathbf{x}_0\|^2 + \frac{L}{2} \|\mathbf{x}^* - \mathbf{x}_0\|^2 \\ &= f(\mathbf{x}^*) + L \|\mathbf{x}^* - \mathbf{x}_0\|^2, \end{aligned}$$

or equivalently $\phi_0(\mathbf{x}^*) - f(\mathbf{x}^*) \leq L \|\mathbf{x}^* - \mathbf{x}_0\|^2$. Using Property 3 and 4, we obtain

$$f(\mathbf{x}_\tau) - f(\mathbf{x}^*) \leq L \left(1 - \sqrt{\frac{\mu}{L}}\right)^\tau \|\mathbf{x}^* - \mathbf{x}_0\|^2.$$

□

Remark B.4.7. The above convergence rate has indeed improved over Theorem B.3.3 because the geometric rate in which the error decreases is now $(1 - 1/\sqrt{\kappa})$, bounded further away from 1.

Remark B.4.8. For weakly convex functions, i.e., $\mu = 0$, we can derive from (B.31) that

$$\xi_{\tau+1} = (1 - \alpha_\tau)\xi_\tau = (1 - \alpha_\tau)L\alpha_{\tau-1}^2 = L\alpha_\tau^2,$$

or equivalently

$$\frac{1}{\alpha_\tau} \left(\frac{1}{\alpha_\tau} - 1\right) = \left(\frac{1}{\alpha_{\tau-1}}\right)^2.$$

In other words, we recover exactly the same update formula for t_τ in Algorithm 4 by making the identification $t_\tau = \frac{1}{\alpha_\tau}$. In this sense, the procedure of Beck and Teboulle (2009) (Algorithm 4) is a simplification of Nesterov's work on estimate sequence, their theoretical contribution is in giving an elementary proof of the algorithm.

Remark B.4.9. A remarkably simple stepsize rule for Algorithm 5 is obtained by choosing $\xi_0 = \mu$ and $\alpha_0 = \sqrt{\frac{\mu}{L}}$. It is easy to see that this leads to constant stepsize $\alpha_\tau = \sqrt{\frac{\mu}{L}}$ for $\tau = 0, 1, \dots$

Bibliography

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In Bernhard Schölkopf, John Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 19, pages 41–48. MIT Press, Cambridge, MA, 2007.

Raman Arora, Maya Gupta, Amol Kapila, and Maryam Fazel. Clustering by left-stochastic matrix factorization. In Lise Getoor and Tobias Scheffer, editors, *Proc. of the 28th Int. Conf. Machine Learning (ICML 2011)*, pages 761–768, Bellevue, WA, June 28 – July 2 2011.

David Arthur and Sergei Vassilvitskii. k -means++: The advantages of careful seeding. In *Proc. of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA 2007)*, pages 1027–1035, New Orleans, LA, January 7–9 2007.

Laura Balzano, Robert Nowak, and Benjamin Recht. Online identification and tracking of subspaces from highly incomplete information. arXiv:1006.4046 [cs.IT], June 2010.

Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382, September 2005.

Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. The Johnson-Lindenstrauss lemma meets compressed sensing. 2006.

Richard G. Baraniuk and Michael B. Wakin. Random projections of smooth manifolds.

- Foundations of Computational Mathematics*, 9(1):51–77, February 2009.
- Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1):183–202, 2009.
- S. Becker, J. Bobin, and E.J. Candès. NESTA: A fast and accurate first-order method for sparse recovery. *SIAM J. Imaging Sciences*, 4(1):1–39, 2011.
- Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 585–591. MIT Press, Cambridge, MA, 2002.
- Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, June 2003.
- Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, November 2006.
- Robert Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Proc. of the 7th IEEE Int. Conf. Data Mining (ICDM'07)*, pages 43–52, Omaha, NE, October 28–31 2007.
- Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Nashua, NH, second edition, 1999.
- Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, Oxford, 1995.
- Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Series in Information Science and Statistics. Springer-Verlag, Berlin, 2006.
- Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, U.K., 2004.
- Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed

- optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.
- Matthew Brand. Charting a manifold. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 985–992. MIT Press, Cambridge, MA, 2003.
- Peter Brucker. An $O(n)$ algorithm for quadratic knapsack problems. *Operations Research Letters*, 3(3):163–166, 1984.
- Jörg Bruske and Gerald Sommer. Intrinsic dimensionality estimation with optimally topology preserving maps. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(5):572–575, May 1998.
- A. M. Buchanan and A. W. Fitzgibbon. Damped Newton algorithms for matrix factorization with missing data. In *Proc. of the 2005 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'05)*, pages 316–322, San Diego, CA, June 20–25 2005.
- Deng Cai, Xiaofei He, Jiawei Han, and Thomas S. Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(8):1548–1560, August 2011.
- Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.
- Francesco Camastra and Alessandro Vinciarelli. Estimating the intrinsic dimension of data with a fractal-based method. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(10):1405–1408, October 2002.
- Guillermo Canas, Tomaso Poggio, and Lorenzo Rosasco. Learning manifolds with k-means and k-flats. In P. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 25, pages 2474–2482. MIT Press, Cambridge, MA, 2012.

- Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, December 2009.
- Emmanuel J. Candès and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Information Theory*, 52(12):5406–5425, 2006.
- Emmanuel J. Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Information Theory*, 56(5):2053–2080, April 2010.
- Miguel Á. Carreira-Perpiñán. Mode-finding for mixtures of Gaussian distributions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1318–1323, November 2000.
- Miguel Á. Carreira-Perpiñán. Acceleration strategies for Gaussian mean-shift image segmentation. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *Proc. of the 2006 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'06)*, pages 1160–1167, New York, NY, June 17–22 2006a.
- Miguel Á. Carreira-Perpiñán. Fast nonparametric clustering with Gaussian blurring mean-shift. In William W. Cohen and Andrew Moore, editors, *Proc. of the 23rd Int. Conf. Machine Learning (ICML'06)*, pages 153–160, Pittsburgh, PA, June 25–29 2006b.
- Miguel Á. Carreira-Perpiñán. Gaussian mean shift is an EM algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(5):767–776, May 2007.
- Miguel Á. Carreira-Perpiñán. Generalised blurring mean-shift algorithms for nonparametric clustering. In *Proc. of the 2008 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'08)*, pages 1–8, Anchorage, AK, June 23–28 2008.
- Miguel Á. Carreira-Perpiñán and Zhengdong Lu. The Laplacian Eigenmaps Latent Variable Model. In Marina Meilă and Xiaotong Shen, editors, *Proc. of the 11th Int. Work-*

- shop on Artificial Intelligence and Statistics (AISTATS 2007)*, pages 59–66, San Juan, Puerto Rico, March 21–24 2007.
- Miguel Á. Carreira-Perpiñán and Zhengdong Lu. Dimensionality reduction by unsupervised regression. In *Proc. of the 2008 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'08)*, Anchorage, AK, June 23–28 2008.
- Miguel Á. Carreira-Perpiñán and Zhengdong Lu. Manifold learning and missing data recovery through unsupervised regression. In *Proc. of the 12th IEEE Int. Conf. Data Mining (ICDM 2011)*, pages 1014–1019, Vancouver, BC, December 11–14 2011.
- Miguel Á. Carreira-Perpiñán and Weiran Wang. The k -modes algorithm for clustering. arXiv:1304.6478 [cs.LG], April 23 2013a.
- Miguel Á. Carreira-Perpiñán and Weiran Wang. A simple assignment model with Laplacian smoothing. Unpublished manuscript, 2013b.
- Miguel Á. Carreira-Perpiñán and Richard S. Zemel. Proximity graphs for clustering and manifold learning. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 225–232. MIT Press, Cambridge, MA, 2005.
- Hasan Ertan Cetingül and René Vidal. Intrinsic mean shift for clustering on Stiefel and Grassmann manifolds. In *Proc. of the 2009 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'09)*, pages 1896–1902, Miami, FL, June 20–26 2009.
- Anil Chaturvedi, Paul E. Green, and J. Douglas Carroll. k -modes clustering. *Journal of Classification*, 18(1):35–55, January 2001.
- Minhua Chen, Jorge Silva, John Paisley, Chunping Wang, David Dunson, and Lawrence Carin. Compressive sensing on manifolds using a nonparametric mixture of factor analyzers: Algorithm and performance bounds. *IEEE Trans. Signal Processing*, 58(12):6140–6155, December 2010.

- Yunmei Chen and Xiaojing Ye. Projection onto a simplex. arXiv:1101.6081, February 10 2011.
- Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(8):790–799, August 1995.
- Chakra Chennubhotla and Allan Jepson. EigenCuts: Half-lives of EigenFlows for spectral clustering. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 705–712. MIT Press, Cambridge, MA, 2003.
- A. L. Chistov and D. Yu. Grigoriev. Complexity of quantifier elimination in the theory of algebraically closed fields. In *Proceedings of the 11th Mathematical Foundations of Computer Science*, volume 176 of *Lecture Notes in Computer Science*, pages 17–31, 1984.
- Fan R. K. Chung. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, Providence, RI, 1997.
- Ulrich Clarenz, Udo Diewald, and Martin Rumpf. Processing textured surfaces via anisotropic geometric diffusion. *IEEE Trans. Image Processing*, 13(2):248–261, February 2004.
- R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proc. Natl. Acad. Sci. USA*, 102(21):7426–7431, May 24 2005.
- Robert T. Collins. Mean-shift blob tracking through scale space. In *Proc. of the 2003 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'03)*, pages 234–240, Madison, Wisconsin, June 16–22 2003.
- Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

- Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(5):564–577, May 2003.
- Jose A. Costa and Alfred O. Hero. Geodesic entropic graphs for dimension and entropy estimation in manifold learning. *IEEE Trans. Signal Processing*, 52(8):2210–2221, August 2004.
- Sanjoy Dasgupta and Yoav Freund. Random projection trees for vector quantization. *IEEE Trans. Information Theory*, 55(7):3229–3242, July 2009.
- Sanjoy Dasgupta and Anupam Gupta. An elementary proof of the Johnson-Lindenstrauss lemma. Technical Report TR-99-006, International Computer Science Institute (ICSI), 1999.
- V. de Silva and Joshua B. Tenenbaum. Global versus local approaches to nonlinear dimensionality reduction. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 721–728. MIT Press, Cambridge, MA, 2003.
- David Edward DeMers. *Learning to Invert Many-to-One Mappings*. PhD thesis, University of California, San Diego, 1993.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society, B*, 39(1):1–38, 1977.
- Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In Warren Waggenspack, editor, *Proc. of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1999)*, pages 317–324, Los Angeles, CA, August 8–13 1999.
- Tamal K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*, volume 23 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, 2007.

- David L. Donoho. Compressed sensing. *IEEE Trans. Information Theory*, 52(4):1289–1306, April 2006.
- David L. Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. Natl. Acad. Sci. USA*, 100(10):5591–5596, May 13 2003.
- John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the ℓ_1 -ball for learning in high dimensions. In Andrew McCallum and Sam Roweis, editors, *Proc. of the 25th Int. Conf. Machine Learning (ICML'08)*, pages 272–279, Helsinki, Finland, July 5–9 2008.
- John Duchi, Shai Shalev-shwartz, Yoram Singer, and Ambuj Tewari. Composite objective mirror descent. In Adam Tauman Kalai and Mehryar Mohri, editors, *Proc. of the 23th Annual Conference on Learning Theory (COLT'10)*, pages 14–26, Haifa, Israel, June 27–29 2010.
- Richard Durbin and David Willshaw. An analogue approach to the traveling salesman problem using an elastic net method. *Nature*, 326(6114):689–691, April 16 1987.
- Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *Proc. of the 2009 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'09)*, pages 2790–2797, Miami, FL, June 20–26 2009.
- Ehsan Elhamifar and Rene Vidal. Sparse manifold clustering and embedding. In J. Shawe-Taylor, R. S. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 24, pages 55–63. MIT Press, Cambridge, MA, 2011.
- Patrick Etyngier, Florent Ségonne, and Renaud Keriven. Shape priors using manifold learning techniques. In *Proc. 11th Int. Conf. Computer Vision (ICCV'07)*, pages 1–8, Rio de Janeiro, Brazil, October 14–21 2007.
- Keinosuke Fukunaga and Larry D. Hostetler. The estimation of the gradient of a density

- function, with application in pattern recognition. *IEEE Trans. Information Theory*, IT-21(1):32–40, January 1975.
- Keinosuke Fukunaga and David R. Olsen. An algorithm for finding intrinsic dimensionality of data. *IEEE Trans. Computers*, C-20(2):176–183, February 1971.
- Zoubin Ghahramani and Michael I. Jordan. Learning from incomplete data. Technical Report 108, MIT Center for Biological and Computational Learning, 1994.
- Albert Gifi. *Nonlinear Multivariate Analysis*. John Wiley & Sons, 1990.
- Alvina Goh and René Vidal. Clustering and dimensionality reduction on Riemannian manifolds. In *Proc. of the 2008 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'08)*, Anchorage, AK, June 23–28 2008.
- Dian Gong, Fei Sha, and Gerard Medioni. Locally linear denoising on image manifolds. In Yee Whye Teh and Mike Titterton, editors, *Proc. of the 13th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2010)*, pages 265–272, Chia Laguna, Sardinia, Italy, March 21–24 2010.
- Peter Grassberger and Itamar Procaccia. Measuring the strangeness of strange attractors. *Physica D*, 9(1–2):189–208, October 1983.
- Trevor J. Hastie, Robert J. Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning—Data Mining, Inference and Prediction*. Springer Series in Statistics. Springer-Verlag, second edition, 2009.
- Matthias Hein and Markus Maier. Manifold denoising. In Bernhard Schölkopf, John Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 19, pages 561–568. MIT Press, Cambridge, MA, 2007.
- G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 28 2006.
- Geoffrey Hinton and Sam T. Roweis. Stochastic neighbor embedding. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Informa-*

- tion Processing Systems (NIPS)*, volume 15, pages 857–864. MIT Press, Cambridge, MA, 2003.
- Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k -center problem. *Math. Oper. Res.*, 10(2):180–184, May 1985.
- Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, U.K., 1986.
- Chonghai Hu, James Kwok, and Weike Pan. Accelerated gradient methods for stochastic optimization and online learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 22, pages 781–789. MIT Press, Cambridge, MA, 2009.
- Zhexue Huang. Extensions to the k -means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(2):283–304, September 1998.
- Prateek Jain, Raghu Meka, and Inderjit S. Dhillon. Guaranteed rank minimization via singular value projection. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 23, pages 937–945. MIT Press, Cambridge, MA, 2010.
- Shuiwang Ji and Jieping Ye. An accelerated gradient method for trace norm minimization. In Léon Bottou and Michael Littman, editors, *Proc. of the 26th Int. Conf. Machine Learning (ICML'09)*, pages 457–464, Montreal, Canada, June 14–18 2009.
- William Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in Modern Analysis and Probability*, pages 189–206, New Haven, CT, 1984. American Mathematical Society.
- Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 1990.
- Balázs Kégl. Intrinsic dimension estimation using packing numbers. In Suzanna Becker,

- Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 697–704. MIT Press, Cambridge, MA, 2003.
- Raghuveer H. Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from a few entries. *IEEE Trans. Information Theory*, 56(6):2980–2998, 2010.
- Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proc. of the 14th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (SIGKDD 2008)*, pages 426–434, Las Vegas, NV, August 24–27 2008.
- Guanghui Lan. An optimal method for stochastic composite optimization. *Math. Prog.*, 133(1–2):365–397, June 2012.
- Carsten Lange and Konrad Polthier. Anisotropic smoothing of point sets. *Computer Aided Geometric Design*, 22(7):680–692, October 2005.
- Neil D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 16. MIT Press, Cambridge, MA, 2004.
- Neil D. Lawrence and Raquel Urtasun. Non-linear matrix factorization with Gaussian processes. In Léon Bottou and Michael Littman, editors, *Proc. of the 26th Int. Conf. Machine Learning (ICML'09)*, pages 601–608, Montreal, Canada, June 14–18 2009.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, November 1998.
- Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 21 1999.
- David Levin. Mesh-independent surface interpolation. In Guido Brunnett, Bernd

- Hamann, Heinrich Müller, and Lars Linsen, editors, *Geometric Modeling for Scientific Visualization*, pages 37–49. Springer-Verlag, 2003.
- Elizaveta Levina and Peter J. Bickel. Maximum likelihood estimation of intrinsic dimension. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 777–784. MIT Press, Cambridge, MA, 2005.
- Adrian S. Lewis and Jérôme Malick. Alternating projections on manifolds. *Math. Oper. Res.*, 33(1):February, 216–234 2008.
- Ping Li and Trevor Hastie. A unified near-optimal estimator for dimension reduction in l_α ($0 < \alpha \leq 2$) using stable random projections. In John C. Platt, Daphne Koller, Yoram Singer, and Sam Roweis, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 20. MIT Press, Cambridge, MA, 2008.
- Wei Liu, Junfeng He, and Shih-Fu Chang. Large graph construction for scalable semi-supervised learning. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proc. of the 27th Int. Conf. Machine Learning (ICML 2010)*, pages 679–686, Haifa, Israel, June 21–25 2010.
- Zhang Liu and Lieven Vandenberghe. Interior-point method for nuclear norm approximation with application to system identification. *SIAM J. Matrix Anal. and Apps.*, 31(3):1235–1256, 2009.
- Raghu Meka, Prateek Jain, Constantine Caramanis, and Inderjit S. Dhillon. Rank minimization via online learning. In Andrew McCallum and Sam Roweis, editors, *Proc. of the 25th Int. Conf. Machine Learning (ICML'08)*, pages 656–663, Helsinki, Finland, July 5–9 2008.
- Adam Meyerson, Liadan O’Callaghan, and Serge Plotkin. A k -median algorithm with running time independent of data size. *Machine Learning*, 56(1–3):61–87, July–August 2004.

- C. Michelot. A finite algorithm for finding the projection of a point onto the canonical simplex of \mathbb{R}^n . *J. Optimization Theory and Applications*, 50(1):195–200, 1986.
- A. S. Nemirovski and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley & Sons, 1983.
- Y. Nesterov. *Introductory Lectures on Convex Optimization. A Basic Course*. Number 87 in Applied Optimization. Springer-Verlag, 2004.
- Yu. Nesterov. Smooth minimization of non-smooth functions. *Math. Prog.*, 103(1): 127–152, May 2005.
- Yurii Nesterov. Gradient methods for minimizing composite objective function. Technical report, CORE, UCL, September 2007.
- M. E. J. Newman. *Networks: An Introduction*. Oxford University Press, 2010.
- A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 849–856. MIT Press, Cambridge, MA, 2002.
- Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, second edition, 2006.
- Panos M. Pardalos and Naina Kovero. An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Math. Prog.*, 46(1-3):321–328, 1990.
- JinHyeong Park, Zhenyue Zhang, Hongyuan Zha, and Rangachar Kasturi. Local smoothing for manifold learning. In *Proc. of the 2004 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'04)*, pages 452–459, Washington, DC, June 27 – July 2 2004.

- Mark Pauly, Leif P. Kobbelt, and Markus Gross. Point-based multiscale surface representation. *ACM Trans. Graphics*, 25(2):177–193, April 2006.
- Yigang Peng, Arvind Ganesh, John Wright, Wenli Xu, , and Yi Ma. RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images. In *Proc. of the 2010 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'10)*, pages 763–770, San Francisco, CA, June 13–18 2010.
- Pietro Perona and Jitendra Malik. Scale-space and edge-detection using anisotropic diffusion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(7):629–639, July 1990.
- Karl W. Pettis, Thomas A. Bailey, Anil K. Jain, and Richard C. Dubes. Intrinsic dimensionality estimator from near-neighbor information. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1(1):25–37, January 1979.
- Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In John Langford and Joelle Pineau, editors, *Proc. of the 29th Int. Conf. Machine Learning (ICML 2012)*, pages 449–456, Edinburgh, Scotland, June 26 – July 1 2012.
- R. Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control and Optim.*, 14(5):877–898, 1976.
- Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 22 2000.
- Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In John C. Platt, Daphne Koller, Yoram Singer, and Sam Roweis, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 20, pages 1257–1264. MIT Press, Cambridge, MA, 2008a.
- Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In Andrew McCallum and Sam Roweis, editors,

- Proc. of the 25th Int. Conf. Machine Learning (ICML'08)*, pages 880–887, Helsinki, Finland, July 5–9 2008b.
- Tapio Schneider. Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values. *Journal of Climate*, 14(5): 853–871, March 2001.
- Matthias Scholz, Fatma Kaplan, Charles L. Guy, Joachim Kopka, and Joachim Selbig. Non-linear PCA: A missing data approach. *Bioinformatics*, 21(20):3887–3895, October 15 2005.
- Shai Shalev-Shwartz and Yoram Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, 7:1567–1599, 2006.
- Lior Shapira, Shai Avidan, and Ariel Shamir. Mode-detection via median-shift. In *Proc. 12th Int. Conf. Computer Vision (ICCV'09)*, pages 1909–1916, Kyoto, Japan, September 29 – October 2 2009.
- Yaser Ajmal Sheikh, Erum Arif Khan, and Takeo Kanade. Mode-seeking via medoid-shifts. In *Proc. 11th Int. Conf. Computer Vision (ICCV'07)*, Rio de Janeiro, Brazil, October 14–21 2007.
- Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.
- Alexander J. Smola, Sebastian Mika, Bernhard Schölkopf, and Robert C. Williamson. Regularized principal manifolds. *Journal of Machine Learning Research*, 1:179–209, June 2001.
- Mahdi Soltanolkotabi, Ehsan Elhamifar, and Emmanuel J. Candes. Robust subspace clustering. January 2013.
- Nathan Srebro, Jason D. M. Rennie, and Tommi S. Jaakkola. Maximum-margin matrix factorization. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 1329–1336. MIT Press, Cambridge, MA, 2005.

- Raghav Subbarao and Peter Meer. Nonlinear mean shift for clustering over analytic manifolds. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *Proc. of the 2006 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'06)*, pages 1168–1175, New York, NY, June 17–22 2006.
- Raghav Subbarao and Peter Meer. Nonlinear mean shift over Riemannian manifolds. *Int. J. Computer Vision*, 84(1):1–20, August 2009.
- Gabriel Taubin. A signal processing approach to fair surface design. In Susan G. Mair and Robert Cook, editors, *Proc. of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1995)*, pages 351–358, Los Angeles, CA, August 6–11 1995.
- Gabriel Taubin, Tong Zhang, and Gene Golub. Optimal surface smoothing as filter design. In Bernard F. Buxton and Roberto Cipolla, editors, *Proc. 4th European Conf. Computer Vision (ECCV'96)*, volume 1, pages 283–292, Cambridge, UK, April 15–18 1996.
- Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 22 2000.
- Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, B*, 58(1):267–288, 1996.
- Michael E. Tipping and Christopher M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, February 1999.
- Ranjith Unnikrishnan and Martial Hebert. Denoising manifold and non-manifold point clouds. In *Proc. of the 18th British Machine Vision Conference (BMVC 2007)*, Warwick, UK, September 10–13 2007.
- René Vidal. A tutorial on subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, March 2011.
- Pascal Vincent and Yoshua Bengio. Manifold Parzen windows. In Suzanna Becker, Se-

- bastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 849–856. MIT Press, Cambridge, MA, 2003.
- Max Vladymyrov and Miguel Á. Carreira-Perpiñán. Entropic affinities: Properties and efficient numerical computation. In *Proc. of the 230th Int. Conf. Machine Learning (ICML 2013)*, Atlanta, GA, June 16–21 2013.
- Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4): 395–416, December 2007.
- Michael Wakin, Marco Duarte, Shriram Sarvotham, Dror Baron, and Richard Baraniuk. Recovery of jointly sparse signals from few random projections. In Yair Weiss, Bernhard Schölkopf, and John Platt, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 18. MIT Press, Cambridge, MA, 2006.
- M. P. Wand and M. C. Jones. *Kernel Smoothing*. Number 60 in Monographs on Statistics and Applied Probability. Chapman & Hall, London, New York, 1994.
- Weiran Wang and Miguel Á. Carreira-Perpiñán. Manifold blurring mean shift algorithms for manifold denoising. In *Proc. of the 2010 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'10)*, pages 1759–1766, San Francisco, CA, June 13–18 2010.
- Weiran Wang, Miguel Á. Carreira-Perpiñán, and Zhengdong Lu. A denoising view of matrix completion. In J. Shawe-Taylor, R. S. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 24, pages 334–342. MIT Press, Cambridge, MA, 2011.
- Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, October 2010.
- Zhirong Yang and Erkki Oja. Clustering by low-rank doubly stochastic matrix decomposition. In John Langford and Joelle Pineau, editors, *Proc. of the 29th Int. Conf.*

- Machine Learning (ICML 2012)*, pages 831–838, Edinburgh, Scotland, June 26 – July 1 2012.
- Stella X. Yu and Jianbo Shi. Multiclass spectral clustering. In *Proc. 9th Int. Conf. Computer Vision (ICCV'03)*, pages 313–319, Nice, France, October 14–17 2003.
- Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, B*, 68(1):49–67, 2006.
- Xiaotong Yuan, Bao-Gang Hu, and Ran He. Agglomerative mean-shift clustering. *IEEE Trans. Knowledge and Data Engineering*, 24(2):209–219, February 2010.
- Ron Zass and Amnon Shashua. Doubly stochastic normalization for spectral clustering. In Bernhard Schölkopf, John Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 19, pages 1569–1576. MIT Press, Cambridge, MA, 2007.
- Lih Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 1601–1608. MIT Press, Cambridge, MA, 2005.
- Zhenyue Zhang and Hongyuan Zha. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *SIAM J. Sci. Comput.*, 26(1):313–338, 2004.
- Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In Tom Fawcett and Nina Mishra, editors, *Proc. of the 20th Int. Conf. Machine Learning (ICML'03)*, pages 912–919, Washington, DC, August 21–24 2003.