School of Electrical Engineering, Computing and Mathematical Science

Affinity Learning on Graphs with Diffusion Processes

Qilin Li
0000-0001-6584-8879

This thesis is presented for the degree of
Doctor of Philosophy
of
Curtin University

March 2020

Declaration

I hereby declare that this submission is my own work that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other or diploma of the university or other institutes of higher learning, except where due acknowledgment has been made in the text. I have obtained permission from the copyright owners, where necessary, to use any third-party copyright material reproduced in the thesis, or to use any of my own published work in which the copyright is held by another party.

Signature: _____ Date: _____

Abstract

In this thesis, we propose machine learning algorithms utilizing diffusion processes to learn the pairwise affinity between data samples, which is referred to as affinity learning. We show that context-aware affinity can be learned using a diffusion process that is similar to the Markov Random Walk. By walking from one node to another on an edge-weighted graph, the diffusion process propagates pairwise relationships with respect to the local graph structure. The graph edges absorb neighbor affinities and re-evaluate their own affinity values in such an iterative process, resulting in accurate and smooth pairwise affinities. Similar ideas are also embedded to graph convolutional networks, in which a graph smoothness loss is proposed to produce better feature representation, given the input space is not reliable. These proposed algorithms improve performance for various machine learning tasks, such as data cluster analysis, dimensionality reduction, and semi-supervised classification. We believe they are also useful for many other downstream learning tasks.

We focus on high-dimensional visual data in computer vision. To avoid the curse of dimensionality, we hold the manifold assumption, which states that high-dimensional data lie on low-dimensional manifolds or subspaces. Such underlying manifolds can be represented by graph structures, in which graph nodes represent data points, and edges between these nodes encode pairwise similarities. Considering the central role of similarity learning in various machine learning tasks, we conduct in-depth research on learning accurate pairwise similarity in the high-dimensional space. The primary technique used throughout the whole thesis is the diffusion process, and its key idea is to aggregate contextual information such as neighbors. The diffusion process is used to propagate pairwise similarity on graphs to learn manifold-aware similarities. It can be interpreted as a random walk on the graph. While the walkers keep walking from one node to another, the similarities between these nodes are refined after every visit accordingly. We use this technique and many of its variants to build various learning algorithms adapted to different problem settings.

We first propose a diffusion-based sparse subspace clustering (DSSC) algorithm that segments high-dimensional data into different groups. Our main contribution is on learning accurate similarity in the unsupervised setting. This is of great importance, as the pairwise

similarity is the key to identify the pattern of interest due to the lack of supervisory signal. We show that the diffusion process used with sparse representation models can significantly improve the quality of the learned similarities. Next, motivated by the scalability issue of DSSC, we propose a diffusion-based variational Nyström (DVN) algorithm to address the general large-scale spectral problem. DVN rephrases the classic Nyström method with the idea of diffusion and solves the built-in eigenvector problem with an approximate landmark solution. We demonstrate that DVN achieves the best tradeoff between accuracy and efficiency.

We then shift our focus to semi-supervised learning, which aims at learning from both labeled and unlabeled data. We emphasize the importance of affinity learning in this setting, and show affinity learning can significantly improve the classification accuracy, especially for graph-based semi-supervised learning. A novel semi-supervised diffusion process, namely self-reinforced diffusion (SRD), is proposed to take advantage of both the available label information and data distribution to supervise the process of diffusion. SRD is later on extended to another diffusion process, named as alternating diffusion process (ADP), that integrates affinity learning and label propagation into one unified framework, in which affinities and class labels can be learned simultaneously. While SRD improves the accuracy by making use of the given partial labels, ADP can produce further improvements with marginal computational overheads, by incorporating intermediate estimated labels as feedbacks to guide the diffusion.

Last but not least, we study representation learning on graphs with graph convolutional networks (GCNs). GCNs are a new type of neural network which aims at generalizing convolutional operations on graphical data. It shows a huge potential of learning good representations similar to convolutional neural networks (CNNs). We establish connections between GCNs and the diffusion process, and show that the layer-wise propagation in GCNs is similar to the iterative operation in the diffusion process. We then propose a regularization loss that can be derived from the diffusion process to deal with the overfitting problem of existing GCNs. Extensive experiments show that it can improve state-of-the-art GCNs by simply incorporating the proposed loss function.

Acknowledgments

I would like to express my deepest gratitude to the following people for their great guidance and support during my PhD study. This thesis could not have been possible without them.

- First and foremost, I wish to express my sincere appreciation to my supervisor Associate Professor Wanquan Liu and co-supervisor Professor Ling Li, who provided me with constant guidance, support, and encouragement. It is whole-heartedly appreciated that your great advice has proven monumental towards the success of this study. It has been a wonderful and invaluable journey to be your students, and I will always be.

- I am grateful to Curtin University for providing financial support for my PhD study through Research Training Program (RTP) scholarship.

- I would like to thank all my fellow research students and researchers (Antoni Liang, Nadith Pathirage, Yan Ren, Yi Zhang, Xin Zhang, Jingjing Liu, Jun Chang, Lu Tan, Ruhua Wang, Xianchao Xiu) for many discussions and inspirations.

- I owe my greatest gratitude to my parents Mr. Hongbo Li and Mrs. Haiyan Wu, who have always guided and supported me in my life. Thanks for believing in me all the time. It has been my source of strength along the way.

- I would also like to thank our administrative officer Mary Mulligan, Ether Yew, and Cindy Wong, for your assistance on the paper works.

- Last but not least, a special thanks goes to my wife Mrs. Xiangmeng Tang, who always stood beside me, encouraged me, and provided me with all the support I needed. Thanks for your constant companion, which has made this long journey so enjoyable and memorable.

Publications

This is a list of works that have been published over the course of the author's PhD Degree in chronological order:

- Liu, J., Liu, W., **Li, Q.**, Ma, S., & Chen, G. (2016). Evaluation of k-svd with different embedded sparse representation algorithms. *In 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 426-432.

- **Li, Q.**, Liu, W., Li, L., & Wang, R. (2017). Towards Large Scale Spectral Problems via Diffusion Process. *In 2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 1-7.

- **Li, Q.**, Liu, W., & Li, L. (2018). Affinity learning via a diffusion process for subspace clustering. *Pattern Recognition*, 84, 39-50.

- Ren, Y., **Li, Q.**, Liu, W., Li, L., & Guan, W. (2019). Semantics characterization for eye shapes based on directional triangle-area curve clustering. *Multimedia Tools and Applications*, 1-34.

- **Li, Q.**, Liu, W., & Li, L. (2019). Self-Reinforced Diffusion for Graph-based Semi-Supervised Learning. *Pattern Recognition Letters*, 125, 439-445.

- **Li, Q.**, An, S., Li, L., & Liu, W. (2019). Semi-supervised Learning on Graph with an Alternating Diffusion Process. *arXiv preprint arXiv:1902.06105.*

- **Li, Q.**, Liu, W., & Li, L. (2020). Regularizing Semi-supervised Graph Convolutional Networks with a Manifold Smoothness Loss. *arXiv preprint arXiv:2002.07031.*

Attribution Statement

Chapters 3 to 7 of this thesis are based on works that have been published with joint-authorship. We hereby make an authorship attribution statement to clarify the contribution of individual authors.

Chapter 3 is based on the publication:

- **Li, Q.**, Liu, W., & Li, L. (2018). Affinity learning via a diffusion process for subspace clustering. *Pattern Recognition*, 84, 39-50.

	Conception and Design	**Data Acquisition and Manipulation**	**Programming**	**Experiments**	**Interpretation and Discussion**	**Manuscript Writing and Revision**	**Total Contribution**
Co-author 1 **(Qilin Li)**	✓	✓	✓	✓	✓	✓	60%
Co-author 1 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 2 (Wanquan Liu)	✓	×	×	×	✓	✓	20%
Co-author 2 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 3 (Ling Li)	✓	×	×	×	✓	✓	20%
Co-author 3 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							

Chapter 4 is based on the publication:

- **Li, Q.**, Liu, W., Li, L., & Wang, R. (2017). Towards Large Scale Spectral Problems via Diffusion Process. *In 2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 1-7.

	Conception and Design	Data Acquisition and Manipulation	Programming	Experiments	Interpretation and Discussion	Manuscript Writing and Revision	Total Contribution
Co-author 1 **(Qilin Li)**	✓	✓	✓	✓	✓	✓	60%
Co-author 1 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 2 (Wanquan Liu)	✓	×	×	×	✓	✓	15%
Co-author 2 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 3 (Ling Li)	✓	×	×	×	✓	✓	15%
Co-author 3 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 4 (Ruhua Wang)	×	×	×	✓	✓	×	10%
Co-author 4 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							

Chapter 5 is based on the publication:

- **Li, Q.**, Liu, W., & Li, L. (2019). Self-Reinforced Diffusion for Graph-based Semi-Supervised Learning. *Pattern Recognition Letters*, 125, 439-445.

	Conception and Design	Data Acquisition and Manipulation	Programming	Experiments	Interpretation and Discussion	Manuscript Writing and Revision	Total Contribution
Co-author 1 **(Qilin Li)**	✓	✓	✓	✓	✓	✓	60%
Co-author 1 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 2 (Wanquan Liu)	✓	×	×	×	✓	✓	20%
Co-author 2 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 3 (Ling Li)	✓	×	×	×	✓	✓	20%
Co-author 3 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							

Chapter 6 is based on the publication:

- **Li, Q.**, An, S., Li, L., & Liu, W. (2019). Semi-supervised Learning on Graph with an Alternating Diffusion Process. *arXiv preprint arXiv:1902.06105*.

	Conception and Design	Data Acquisition and Manipulation	Programming	Experiments	Interpretation and Discussion	Manuscript Writing and Revision	Total Contribution
Co-author 1 **(Qilin Li)**	✓	✓	✓	✓	✓	✓	70%
Co-author 1 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 2 (Senjian An)	✓	×	×	×	✓	×	10%
Co-author 2 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 3 (Ling Li)	✓	×	×	×	✓	✓	10%
Co-author 3 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 4 (Wanquan Liu)	✓	×	×	×	✓	✓	10%
Co-author 4 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							

Chapter 7 is based on the publication:

- **Li, Q.**, Liu, W., & Li, L. (2020). Regularizing Semi-supervised Graph Convolutional Networks with a Manifold Smoothness Loss. *arXiv preprint arXiv:2002.07031*.

	Conception and Design	Data Acquisition and Manipulation	Programming	Experiments	Interpretation and Discussion	Manuscript Writing and Revision	Total Contribution
Co-author 1 **(Qilin Li)**	✓	✓	✓	✓	✓	✓	60%
Co-author 1 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 2 (Wanquan Liu)	✓	×	×	×	✓	✓	20%
Co-author 2 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							
Co-author 3 (Ling Li)	✓	×	×	×	✓	✓	20%
Co-author 3 Acknowledgement: I acknowledge that these represent my contribution to the above research output Signed:							

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning refers to the scientific study of analytical models and algorithms that computer systems use to perform a specific task without being explicitly programmed. It is seen as a subset of artificial intelligence (AI) based on the idea that statistical models can learn from data, identify patterns in them, and make predictions or decisions, with minimal human intervention.

The term of machine learning was first coined in 1959 when machine learning research dealt mostly with pattern classification [107]. Bayesian methods were also introduced for probabilistic inference in this field around the same time [126]. Machine learning was seen as a promising path to AI then. However, the research focus of AI later shifted to the logic and knowledge-based expert systems, which basically dominated the AI field by the 1980s. The statistical nature of machine learning was out of favor, which causes a rift between AI and machine learning. By the 1990s, after the knowledge-based expert systems have been outperformed by the data-driven approaches, machine learning started to flourish again. The representatives were support vector machines (SVMs) in the 1990s, kernel methods in the 2000s, and deep learning methods in the 2010s.

Despite the various stages and approaches, machine learning can be broadly categorized into supervised learning, unsupervised learning, and semi-supervised learning. From the terms, we can easily see that the main difference is the availability of the supervisory signal given by humans.

Supervised learning algorithms build statistical models based on labeled training data. A training sample consists of an input object (typically a feature vector) and the desired output

value (the so-called supervisory signal). A supervised model analyzes the input training data and learns an inferred function that maps the input to its desired output. This function can then be used to make predictions for unseen test data. Supervised learning mainly includes classification and regression. Classification models are used when the output set is limited and discrete, while regression refers to problems that contain continuous output values.

Unsupervised learning algorithms, on contrary, take a set of data that contains only the object as input and find underlying structures and manifolds of the data. Clustering is the most widely used unsupervised learning method. It refers to the problem of identifying natural groups within the data so that samples in the same groups or clusters are more similar than those in different clusters. Due to the lack of supervisory labels, clustering is somewhat an ambiguous process that users may have no idea what the groups represent even after clustering. It has to be manually analyzed to figure out the unexplored patterns of each group. Exactly because of this reason, clustering is extremely useful in data mining where the goal is to identify previously unknown patterns and knowledge in the data, while supervised learning such as classification can only categorize or recognize several existing patterns or classes.

Semi-supervised learning (SSL) algorithms make use of both labeled and unlabeled data (typically a small amount of labeled data with a large amount of unlabeled data). It falls between the supervised learning, where all the data are labeled, and unsupervised learning, where none of the data is labeled. SSL is of interest due to the observation that the performance of supervised learning models can be improved by injecting certain amounts of unlabeled data. It is also useful for many real-world applications, in which the labeled data is very expensive to acquire while the acquisition of unlabeled data is relatively easy. On the other hand, SSL is more similar to the learning strategy of humans who learn from the supervision of others as well as from their own observations. Fig.1-1 shows an overview of various fields of machine learning and their possible real-world applications.

Despite the various learning paradigms, a common critical issue is similarity or affinity learning when they are applied for pattern recognition. The goal of pattern recognition is to identify data patterns that are represented by a group of similar objects, for example, a class in classification or a cluster in clustering. The pattern recognition problem would be effectively solved if the similarities can be precisely defined given a pair of samples. It is easy to define similarity for low-dimensional data, in which a simple Euclidean distance or cosine

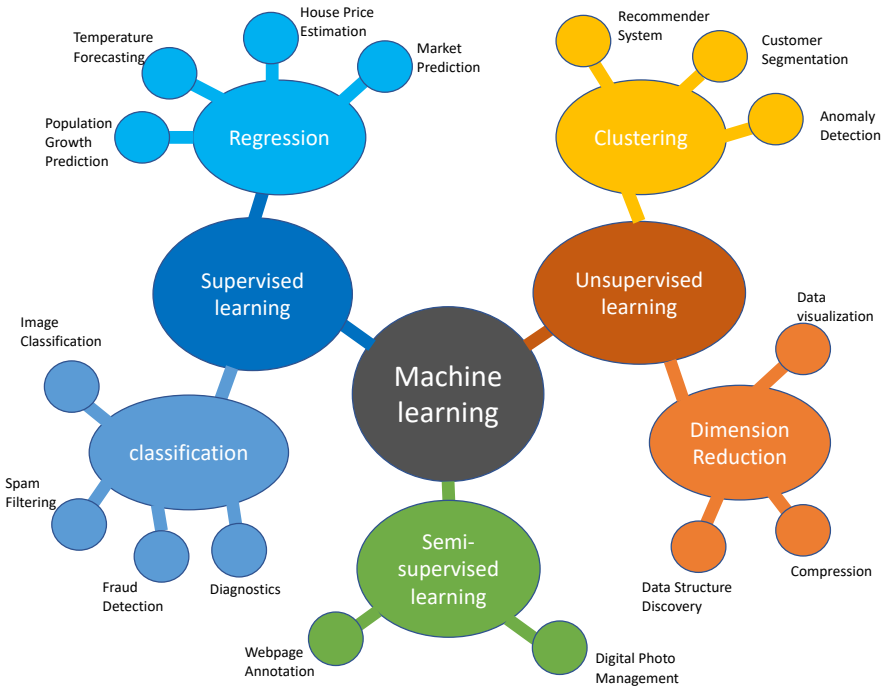

Figure 1-1: An overview of the field of machine learning: various types of learning and their corresponding applications.

similarity could work. But it is difficult to measure the similarity in high-dimensional space (usually hundreds or thousands dimensions), due to the "curse of dimensionality" [13]. When the data dimensionality increases, the volume of its ambient space increases so fast that the available data become sparse. This sparsity is problematic for similarity measurement, as all samples in the sparse space are dissimilar in many ways, which weakens the distinction between intra-class and inter-class similarities.

In this thesis, we focus on pattern recognition for computer vision problems, in which the data is usually high-dimensional, *e.g.*, high-resolution images. The manifold assumption is used to deal with the high-dimensionality problem, and we use the node-edge graph structure to approximate the underlying data manifold. Graph nodes represent data points, while edges are weighted by their pairwise affinity or similarity. This relaxation helps us reduce the problem to learn accurate pairwise affinity, assuming a reliable set of features is provided for the data. Once an accurate graph is constructed, various machine learning tasks can be performed on it, *e.g.*, clustering, semi-supervised classification. On the other hand, if the given feature is not good enough for the task, we propose to learn specific representations using a data-driven, task-driven approach.

1.1 Research Objectives

The main goal of this research is to learn pairwise affinities in the high-dimensional data space so that it can be utilized to improve the solution for various machine learning tasks. After conducting an in-depth literature review in this direction, we identified several research gaps and drawbacks of the existing approaches. The following research objectives are hence proposed to address them respectively.

- **Unsupervised learning of context-aware affinity**. Despite the significance of affinity measurement, the oversimplified Gaussian kernel and its variants are still widely used. There are mainly two issues with the Gaussian kernel. 1) It is usually based on the Euclidean distance, which easily fails to capture the underlying data structure in high-dimensional space. Under the manifold assumption, high-dimensional data generally lie on low-dimensional subspaces or manifolds. Euclidean distance cannot sufficiently approximate the geodesic distance between samples on the manifolds. 2) Pairwise affinities are calculated independently. According to the formula of Gaussian kernel, *i.e.*, $W_{ij} = \exp\left(-\|x_i - x_j\|^2 / 2\sigma^2\right)$, the affinity is defined on the feature representations of the pair of samples only, which is problematic especially in high-dimensional sparse space. The first issue can be addressed by sparse representation [36]. In addition to that, we propose to learn context-aware affinity using a diffusion process, in which the pairwise affinities are not calculated between one sample to another sample, but one set to another set, such as the k-nearest neighbor (kNN) set. We demonstrate the details of this in Chapter 3, where we embed the proposed idea in the context of the sparse subspace clustering.

- **Large-scale spectral problems with a diffusion process**. The approach of affinity learning mentioned above, followed by spectral clustering, can be used for various problems, such as pattern recognition and data mining. However, it is not feasible for large-scale datasets, due to the cubic time complexity $\mathcal{O}(N^3)$ of the spectral clustering. To address this scalability concern, we propose to use the idea of the diffusion process to set up a reduced eigenproblem for approximate eigendecomposition. The proposed approach can be universally utilized in conjunction with any other algorithms that involve spectral eigendecomposition, such as spectral clustering and many non-linear dimensionality reduction approaches based on eigenvectors.

- **Semi-supervised learning of non-local affinity**. Affinity learning is often a critical component of semi-supervised learning. The core idea is to learn pairwise affinities so that we can propagate class labels from the labeled samples to the unlabeled ones. Despite its drawbacks, the Gaussian kernel is still widely used in this filed, limiting the performance of the downstream label propagation process. We introduce the diffusion process to tackle this problem. To make full use of the supervisory signal, we propose a semi-supervised variant of the diffusion process. This also helps us to close another research gap since all existing diffusion processes are unsupervised, and we show that the diffusion process incorporating the global label information leads to more accurate affinities.

- **Representation learning with graph convolutional networks**. The success of machine learning approaches depends on having a good representation of data. The aforementioned research goals are based on an implicit assumption that the input space is good enough for affinity learning. However, this is often not true for complicated computer vision problems, in which the raw space is usually noisy and high-dimensional. In this case, we aim at learning useful representations specifically for the target problem. It means that feature extraction is no longer treated as a pre-processing step, but part of the machine learning problem itself. Deep convolutional networks have proven its capability of serving this purpose. To work on the graph structure, we study graph convolutional networks, a new type of convolutional networks that deal with graphical data directly, in which we try to embed iterative operations of the diffusion process into the layer-wise structure of neural networks.

1.2 Thesis Structure and Contribution

In this section, we briefly describe the content of each chapter and claim our contribution along the way.

- **Chapter 2** introduces the preliminary knowledge related to this thesis. We begin with the introduction of concepts of supervised learning and unsupervised learning. Two relevant subfields of unsupervised learning are discussed in more detail, *i.e.*, clustering, and dimensionality reduction. We then talk about the semi-supervised learning in which affinity learning is also important. We focus on the graph-based semi-supervised

learning, which propagates labels on the affinity graph. Next, we introduce a relatively new concept in machine learning, namely representation learning. It refers to the problem of learning high-level abstract feature representations from the low-level raw features, which is the core idea of deep learning. Next, we introduce the core data structure used throughout the thesis, the affinity matrix, or equivalently the affinity graph. The main objective of this research is to learn a proper affinity graph that can be used for various machine learning problems. Finally, we introduce the main technique used for affinity learning, the so-called diffusion process. It can be interpreted as Markov Random Walks on the affinity graph. The walkers randomly walk between the nodes and refine their affinities afterward, taking the neighbor context into consideration.

- **Chapter 3** focuses on affinity learning for high-dimensional data in the unsupervised setting. Instead of using the Gaussian kernel, which is problematic in this scenario, we utilize a self-expressive model that represents each sample as a linear combination of all other samples to learn the affinity. Regularizing the reconstruction coefficients with ℓ_1 norm encourages the model to learn spare representation using the samples within the same subspace or class. The coefficients can then be served as the affinities. These affinities are used as initial conditions and post-processed by a diffusion process that augments and re-evaluates these affinity values using the contextual information. The proposed idea is discussed in the field of clustering and presented as a novel subspace clustering algorithm. **The contribution in this chapter** is two-fold: in terms of sparse subspace clustering, the embedding of the diffusion process alleviates the connectedness problem brought by the ℓ_1 sparsity. On the other hand, from the diffusion process point view, we show that instead of finding kNN in the Euclidean space, the sparse property of ℓ_1 norm provides manifold-aware neighbor identification that can be used to construct a better initial affinity graph. Experiments on motion segmentation and facial image clustering demonstrate that the embedded affinity learning process does improve the clustering process significantly.

- **Chapter 4** addresses the scalability issue of the general spectral problem via the diffusion process. The clustering algorithm proposed in Chapter 3 works well for small-scale data. However, it is not suitable for large-scale applications due to the cubic computa-

tion cost of the embedded spectral clustering. The classic solution here is the Nyström method [44] that samples a subset of samples for a reduced eigenproblem to approximate the solution. **The contribution in this chapter** is that we formulate the Nyström method as a Markov Random Walk process and propose a generic framework that generalizes to all Nyström-like approaches using the idea of diffusion. With the proposed framework, we propose a novel approximated approach for the large-scale spectral problem using a higher-order Markov transition matrix. The proposed approach is evaluated in the context of spectral clustering and non-linear dimensionality reduction. The results show that the proposed approach achieves the best tradeoff between accuracy and efficiency, compared to existing approximate approaches.

- In **Chapter 5** we investigate affinity learning in graph-based semi-supervised setting, where the label is partially available. Various conventional label propagation approaches are discussed, and they often focus on designing propagation strategies that spread labels on a predefined affinity graph. We show that it is more important to construct a good affinity graph than selecting a propagation algorithm. **The contribution in this chapter** is that we propose a semi-supervised variant of the diffusion process to learn affinities that accords with both the local manifold structure and the global label information. We also propose the idea of "self-similarity" to incorporate the data distribution into the diffusion process. We then formulate the proposed approach as an optimization problem in which the target is to minimize an objective function that contains a smoothness term and a fitness term. In this way, the semi-supervised diffusion process can be interpreted as a procedure of learning affinity to fit the labeled data while being regularized by the manifold smoothness of the unlabeled data. We conduct experiments on synthetic data and real-world data, and it is verified that affinity learning based on the diffusion process significantly improves the classification results of label propagation methods, and the incorporation of partial label information and self-similarity improve the performance even further.

- **Chapter 6** presents an extension to the semi-supervised affinity learning framework proposed in Chapter 5. Graph-based semi-supervised learning involves two separate processes: affinity learning and label propagation. It is suboptimal to manage them independently as the intermediate results of label propagation might be useful for learn-

ing better affinity again. **The contribution in this chapter** is that we integrate these two processes into one unified framework by formulating them as fundamentally similar function estimation problems that can be optimized alternatively or jointly. Although the closed-form solution of the joint optimization problem can be derived, it is computationally expensive. An alternating diffusion process is then proposed to solve it iteratively. An efficient out-of-sample extension is derived from making the proposed approach inductive, *i.e.*, make predictions for unseen data. Extensive experiments show that the alternating diffusion process can yield superior performances in both transductive and inductive semi-supervised classification.

- **Chapter 7** proposes an unsupervised manifold smoothness loss to be used with graph convolutional networks. While most existing GCNs focus on encoding graph structure into layer-wise neural operations by a neighbor aggregation scheme, the proposed loss acts as a regularizer that makes use of graph structure to constrain the network output. **The contribution in this chapter** is that we establish connections between the proposed loss function and the aforementioned diffusion process and show that minimizing this loss is equivalent to running an iterative diffusion process that aggregates the network output of neighboring nodes on the graph. State-of-the-art performances are achieved in semi-supervised learning by plugging the proposed loss into existing graph convolutional networks.

- **Chapter 8** reviews the main contents and concludes the whole thesis. Some promising directions are also discussed for future works.

In the next chapter, we start to introduce the background knowledge related to this thesis. It includes the introduction of closely related subfields and some popular existing approaches in each of these subfields, the data structures that are frequently used, and the main techniques involved.

Chapter 2

Background

The goal of machine learning is to get computers to perform certain tasks without being explicitly programmed. The focus of this field is learning. Computers are supposed to learn the essential skills or knowledge to be able to perform the given specific task. The learning process is usually built upon the experience, or the historical data. The experience or historical data could come in various forms, resulting in different types of learning.

Imagine you are training a baby to distinguish dogs with cats. The baby will not get the concept of dogs unless you show her several dogs and explicitly tell her it is a dog. After enough time of training in this way, the baby will be able to distinguish dogs. This type of learning refers to *supervised* learning.

There are also a lot of other things learned by the baby directly through observation, such as gravity. A six-month-old baby will not be bothered if a toy hovers in the air. But if the same thing happened after several months, she will instantly realize something is not right. The baby had already learned the concept of gravity without being taught by anyone. She learned it by observing the historical falling experience of other objects. This type of learning corresponds to *unsupervised* learning.

Below we introduce these different types of learning strategies in the context of machine learning, including the already mentioned supervised learning, unsupervised learning, semi-supervised learning, and a recently established concept of representation learning. Some preliminary knowledge related to this research will also be covered, such as affinity graphs and diffusion processes.

2.1 Supervised Learning

Supervised learning builds a mathematical model based on training data that contains input-output pairs. As the goal is to get computers to perform specific tasks, we directly give the desired answer and ask the model to fit it. Supervised models usually focus on learning a function that maps the input to the correct output based on the training data. Once the learning is done, it can be used to make predictions for unseen data.

More formally, supervised learning can be described as follows. Given a training set that contains n example input-output pairs of the form $(x_1, y_1), ..., (x_n, y_n)$, where x_i is the feature vector of i-th training samples and y_i is its desired output or label, a supervised learning algorithm seeks a function $f_\theta : X \to Y$, where θ is the parameter of the function. Different θ corresponds to different models. The task becomes finding the parameter θ that makes the function f_θ best fit the training data. In order to measure the fitness of the function to the training data, a loss function $L : Y \times Y \to \mathbb{R}$ is defined. The model is trained by minimizing the average loss on the training set:

$$\min_\theta \frac{1}{n} \sum_i L(y_i, f_\theta(x_i)). \tag{2.1}$$

Classification and regression are two classic supervised learning problems. In classification, the goal is to predict a class label that represents one of the classes to which the sample belongs. The output of classification model is usually a vector $\mathbf{p}$, where each entry $\mathbf{p}_c$ represents the probability of belonging to the c-th class so that the model can be trained by the classic cross-entropy loss:

$$-\frac{1}{n} \sum_i^n \sum_c^k \mathbf{y}_c^i \log \mathbf{p}_c^i, \tag{2.2}$$

where k is the number of class and $\mathbf{y}$ is the ground-truth class label encoded by the one-hot vector, *i.e.*, $\mathbf{y}_c$ equals to 1 if and only if the i-th sample belongs to c-th class and 0 otherwise. Cross-entropy loss imposes a great penalty on the "confident and wrong" prediction due to the logarithm probability. Regression, on the other hand, is to predict a real value for each sample. The loss is usually measured directly between the true value y and the predicted value $\hat{y}$, such as the mean squared error:

$$\frac{1}{n}\sum_{i}^{n}(y^i - \hat{y}^i)^2. \tag{2.3}$$

Despite the difference of the predictions, classification and regression can both be treated as function estimation problems intuitively. The target function in classification is the decision boundary that separates every class from each other, while it is a fitting function between the input variables and the target value in regression. Fig.2-1 shows examples for a binary classification and a linear regression problem.

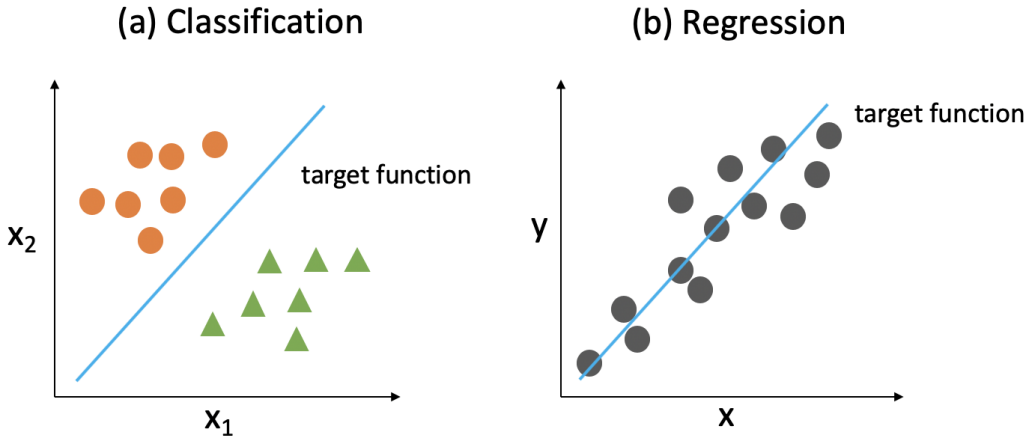

Figure 2-1: Supervised models interpreted as function estimation problems. (a) a binary classification; (b) a linear regression.

Supervised learning is the most commonly used strategy in machine learning, especially for the last ten years, in which the rapid progress is largely driven by deep learning. Supervised deep neural networks, combined with large annotated datasets such as ImageNet [32], have demonstrated superior performances in various domains, such as computer vision, speech recognition, and natural language processing. We now have computer systems that can recognize images with even better accuracy than humans. This leads to revolutions in many real-world applications, including face recognition, autonomous driving, and medical image analysis, to name a few. However, most of these systems currently use supervised deep learning that requires a large labeled dataset. This is prohibitive in some areas where labeled data is expensive to obtain, such as genome annotation, in which it takes several days or even months for a human expert to label a typical genome sequence. A sufficient training set required by supervised deep learning models is impractical in many situations,

which motivates the developments of unsupervised learning.

2.2 Unsupervised Learning

Unsupervised learning refers to the machine learning models that do not make use of supervisory signals. It usually focuses on finding the underlying structure or identifying natural patterns based on the input data itself. Due to the exploratory property, unsupervised learning approaches are usually used for data mining related tasks, where the goal is to discover knowledge from the given data. They are appropriate for exploring unknown patterns, compared to the discriminant analysis among existing patterns in supervised learning. The most commonly used techniques in unsupervised learning are clustering and dimensionality reduction.

2.2.1 Clustering

Clustering, also known as cluster analysis, refers to the process of grouping samples of a dataset into several clusters so that samples in the same cluster are more similar than the samples in different clusters. Clustering algorithms do not depend on the category label tagged on each sample, but on the input vector itself. The absence of supervisory signals distinguishes clustering from classification or discriminant analysis. It is a widely used exploratory data mining technique that finds itself in many applications involving analysis of multivariate data, such as biology, psychology, geology, and marketing, to name a few. The importance and interdisciplinary of clustering is evident in its vast literature. Here we list several typical examples. Image segmentation, a classic computer vision problem, can be formulated as a clustering problem [123]. Customers can be grouped into different types, using clustering for effective marketing [113]. Documents can be clustered to generate topic-wise hierarchies for efficient access [118] and information retrieval [16]. Clustering is also used to group service engagements for workforce management and planning [66], as well as to study genome data in biology [8].

The term clustering itself is not a specific algorithm, but the general task to be addressed. It can be solved by various algorithms that differ significantly in their assumptions for the form of clusters and the way to find them efficiently. Intuitively, clusters can be different in terms of shape, size, or density. The presence of noise in the data makes it even harder

to identify natural clusters. Popular notions of clusters include natural groups with small intra-cluster distance, connected components on a neighborhood graph, subspace with high density, or areas fall into particular probability distributions. The choice of these notions and clustering models depends heavily on the individual dataset and the intended usage of the result. Clustering as such, is not an automatic task but an interactive process of knowledge discovery that involves trial and error.

Due to the lack of supervisory label and prior knowledge, the notion of a "cluster" cannot be precisely defined. This is one of the reasons why there are so many clustering algorithms [37]. Different assumptions for the notion of clusters have been made by different cluster models, and different algorithms are developed for each of these cluster models. It is better to understand these cluster models before trying to distinguish the vast amount of clustering algorithms. Typical cluster models include:

- **Centroid-based models**: the most well-known representative is the k-means clustering that represents a cluster by the mean vector of all its members. It is an appropriate choice for ball-shape clusters.

- **Connectivity models**: the widely used hierarchical clustering falls into this category, where the main task is to calculate the distance or similarity between sets rather than individual samples. Commonly used distance metrics include single linkage, average linkage, and complete linkage [30].

- **Density models**: it defines clusters as connected dense regions of the data space. As the local density of each sample is analyzed, it is relatively robust to data noise and can be used for outlier detection as well, *e.g.*, DBSCAN algorithm.

- **Distribution models**: it assumes that data samples are drawn from specific statistical distributions, such as multivariate Gaussian distribution used by the EM algorithm. These algorithms work well when the data fall into the assumed distribution, but easily fail otherwise.

- **Subspace models**: it is a popular approach for high-dimensional data. The basic idea is the manifold assumption, which states that high-dimensional data usually lie on low-dimensional subspaces or manifolds that corresponds to clusters. Dimensionality

reduction techniques are commonly used by these models to find subspaces, such as PCA, ISOMAP, LLE, *etc.*

- **Graph-based models**: it represents data in the form of a node-edge graph where nodes represent samples, and edges between nodes indicate the pairwise similarities of the corresponding samples. The connected component, such as a clique in the graph, can be used to identify the cluster.

- **Neural network models**: Self-organizing map is the most well known unsupervised neural network model. It shares some commonalities with subspace models as the main idea is also dimensionality reduction.

Fig.2-2 shows several popular clustering models. It can be seen that different models work well on different types of datasets. The success of clustering highly relies on finding the appropriate model. After a particular model is selected, there are also various algorithms to be used. For example, k-means, k-modes, k-medians clustering for centroid-based models, DBSCAN, OPTICS for density-based models, please refer to [71, 119] for comprehensive reviews of clustering algorithms. Next, we discuss several clustering algorithms that are highly related to the research in this thesis.

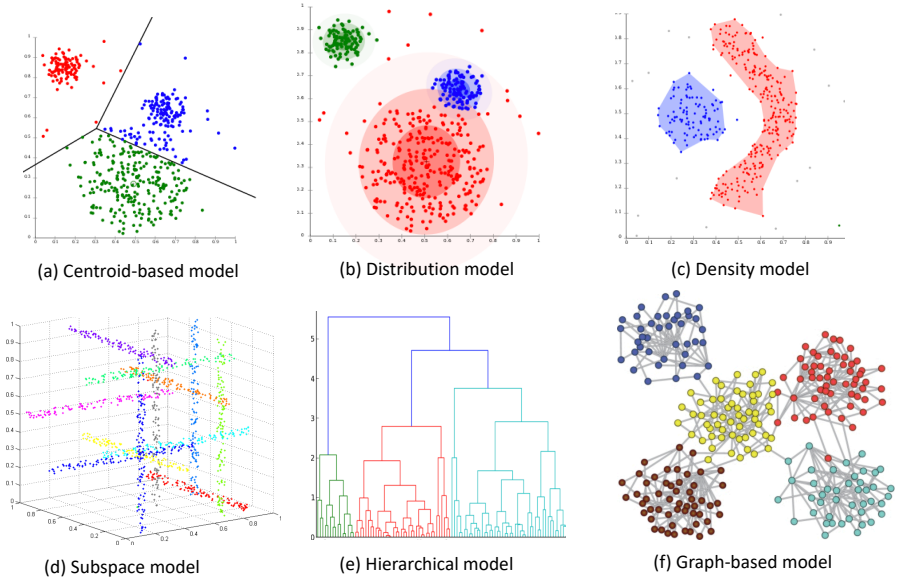

Figure 2-2: Several popular clustering models. Clusters are encoded by colors.

k-means Clustering

The term "k-means" was first coined by James MacQueen in 1967 [100], though the idea of the algorithm dates back to even earlier. After more than half a century, k-means clustering is still widely used in various scientific fields, due to its simple intuition, easy implementation, and empirical success.

Given a set of n samples $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^d$, k-means clustering aims to partition the n samples into a set of k clusters $\mathbf{C} = \{C_1, C_2, ..., C_k\}$ so that the within-cluster sum of squares (WCSS) is minimized. Mathematically, it can be presented as follows:

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{k} \sum_{x \in C_i} \|\mathbf{x} - \mu_i\|^2 , \tag{2.4}$$

where μ_i is the mean vector of samples in cluster C_i. It is NP-hard to find the optimal solution to the k-means clustering problem (2.4) even if there are only two clusters ($k = 2$). The most common algorithm for the k-means problem uses an iterative refinement approach that involves two stages: *assignment* and *update*. In the assignment stage, every sample is visited and assigned to the nearest cluster whose mean has the least squared Euclidean distance with the sample. In the update stage, a new mean or centroid is calculated based on the assigned samples in the cluster. This alternative approach continues until it converges, *i.e.*, assignments no longer change. The k-means algorithm does not guarantee to find the global optimum [58].

Despite its popularity, k-means clustering has two significant drawbacks: 1) Poor initializations may lead to long execution time and poor local minimums. This motivates numerous researches on the initialization of cluster centroids, such as k-means++ [2], rather than purely random initialization. 2) k-means clustering cannot deal with non-linearly separable clusters due to the Euclidean distance calculated in the raw input space. This problem can be addressed by two approaches: kernel k-means and spectral clustering [104]. Kernel k-means is basically the k-means algorithm plus the kernel trick so that the distance can be calculated in higher-dimensional kernel space. Spectral clustering, on the other hand, is essentially the naive k-means clustering with a non-linear dimensionality reduction technique, such as Laplacian Eigenmaps [11] so that the clusters are linearly separable in the new feature space. Several pieces of research show that the kernel k-means and spectral clustering have a close relation that they can be transformed to each other with certain

conditions [34, 43]. Next, we discuss spectral clustering in more detail.

Spectral Clustering

Spectral clustering methods refer to the approaches that make use of eigenvalues (spectrum) of an affinity matrix of the input data to perform dimensionality reduction so that conventional clustering approaches, such as k-means, can be used to partition the data in the lower-dimensional space. As it can be seen as a conventional clustering with a non-linear pre-processing, spectral clustering often outperforms its conventional counterparts, especially for data with arbitrary distributions, as shown in Fig.2-3.

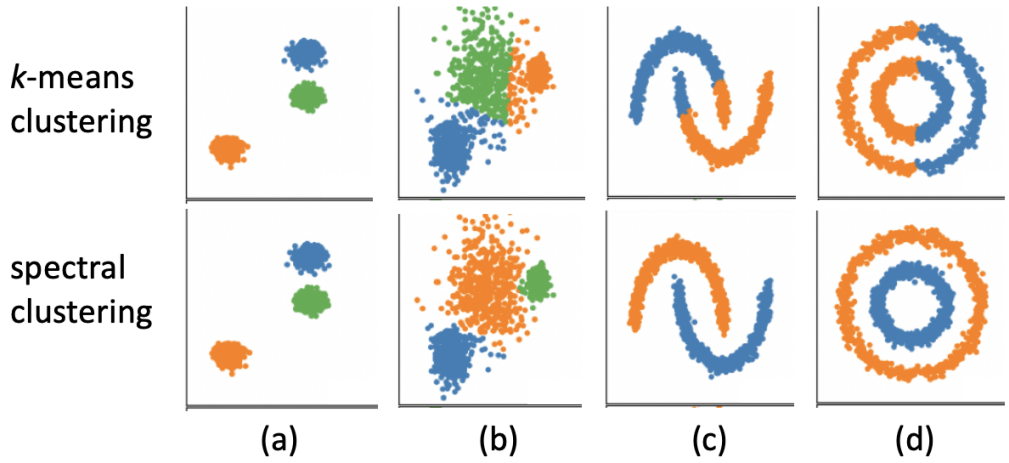

Figure 2-3: Comparison of the spectral clustering and k-means clustering. For the linearly separable cases (a) and (b), there is no big difference. But spectral clustering clearly gives better result than k-means clustering on (c) and (d) that are not linearly separable.

Given a set of n samples $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$, the goal is to partition them into k clusters. Spectral clustering first constructs a similarity or an affinity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ ($\mathbf{A}$ is usually positive and symmetric), where A_{ij} represents the similarity/affinity between $\mathbf{x}_i$ and $\mathbf{x}_j$. Then the unnormalized graph Laplacian matrix $\mathbf{L}$ is derived from the affinity matrix $\mathbf{A}$ as:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{2.5}$$

where $\mathbf{D}$ is a diagonal matrix that $D_{ii} = \sum_{j=1}^{n} \mathbf{A}_{ij}$, also known as the degree matrix. The final clustering result is obtained by performing k-means clustering on the top k eigenvectors that correspond to the k smallest eigenvalues of the matrix $\mathbf{L}$. The intuition of why

eigenvectors of graph Laplacian could be used for clustering is discussed in Spectral Graph Theory [27]. The readers are also referred to [143] for a detailed discussion on the connections between spectral clustering with graph cut problems, random walks, and perturbation theory.

There are several versions of spectral clustering algorithms and researchers are often confused by the difference between them. Due to the lack of precise definition of the so-called graph Laplacian, the main difference comes from the definition of the graph Laplacian matrix $\mathbf{L}$. There are mainly four types of graph laplacian matrix:

$$\begin{cases} \mathbf{L_{un}} = \mathbf{D} - \mathbf{A}, & \text{unnormalized graph Laplacian} \\ \mathbf{L_{rw}} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}, & \text{normalized random walk graph Laplacian} \\ \mathbf{L_{sym}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, & \text{symmetrically normalizaed graph Laplacian} \\ \mathbf{L_{ssym}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, & \text{simplified sysmmetrically normalized graph Laplacian} \end{cases}$$

Despite its various definition, the basic rationale of the graph Laplacian matrix is the same, *i.e.*, clustering is performed on the relevant eigenvectors of the graph laplacian matrix. There is no clear advantage of selecting one over another. In this thesis, we use the form of $\mathbf{L}_{ssym}$ unless otherwise stated. The spectral clustering in this thesis is referred to the widely used NJW algorithm [104]. The formal definition is given in Algorithm 1 for the sake of clarity.

Algorithm 1 Spectral Clustering [104].

Input: An affinity matrix $\mathbf{A}^{n\times n}$, the number of clusters k.

1: Construct the degree matrix $\mathbf{D}$, where $D_{ij} = \sum_{j=1}^{n} \mathbf{A}_{ij}$.
2: Construct the graph Laplacian matrix $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$.
3: Find the $\mathbf{f}_1$, $\mathbf{f}_2$, ..., $\mathbf{f}_k$, the largest k eigenvectors of $\mathbf{L}$, and stack them column-wise to a matrix $\bar{\mathbf{X}} = [\mathbf{f}_1, ..., \mathbf{f}_k] \in \mathbb{R}^{n\times k}$.
4: Treat each row of $\bar{\mathbf{X}}$, *i.e.*, $\bar{\mathbf{x}}_i$, as a sample and perform the k-means clustering.
5: Assign the original sample $\mathbf{x}_i$ to cluster j if and only if $\bar{\mathbf{x}}_i$ is assigned to cluster j, *i.e.*, $Y_i = j$.

Output: The cluster assignments of all samples $\mathbf{Y}$.

Spectral clustering in this thesis refers to Algorithm 1 unless otherwise stated.

2.2.2 Dimensionality Reduction

Advancements in sensing and data storage hardware have paved the way to an era of big data. Rapid growth in relevant applications, such as web search, digital imaging, and video surveillance, leads to abundant high-volumetric and high-dimensional data. Most of the data nowadays are stored in digital media, thus providing tremendous potential for automatic data analysis. However, it is not easy to work with high-dimensional data. The high-dimensionality of data leads to not only the increment of computational cost but also the degradation of the statistical model's performance, due to "the curse of dimensionality". This motivates the research on dimensionality reduction that targets at reducing the number of data dimensions.

Dimensionality reduction can be achieved by three different approaches, namely feature selection, feature projection, and feature extraction. Feature selection refers to the process of selecting a subset of features from existing ones. This type of method works well for relatively low-dimensional data in which redundant features exist, but it is limited by the input feature space as no new feature is generated. Feature projection, on the other hand, transforms the data in the high-dimensional space to a space of lower dimensions. The transformation could be linear, as in Principal Component Analysis (PCA) or Locality Preserving Projection (LPP) [61], so that a projection matrix can be derived. There are also non-linear feature projection algorithms, such as Laplacian Eigenmaps [11] and Locally Linear Embedding [117]. Feature extraction is the process of extracting new features from the given input feature space. It is commonly referred to as representation learning, especially in the deep learning community. Fig.2-4 gives an illustration of these dimensionality reduction models.

In this thesis, we focus on the dimensionality reduction based on feature projection in Chapter 4, and discuss representation learning, in Chapter 7.

Principal Component Analysis

Principal component analysis (PCA) is the most well-known technique for linear dimensionality reduction. It is a statistical process that uses an orthogonal transformation to project the input data to a space with lower dimensions. After projection, all dimensions are linearly uncorrelated and called principal components. From the information point of view,

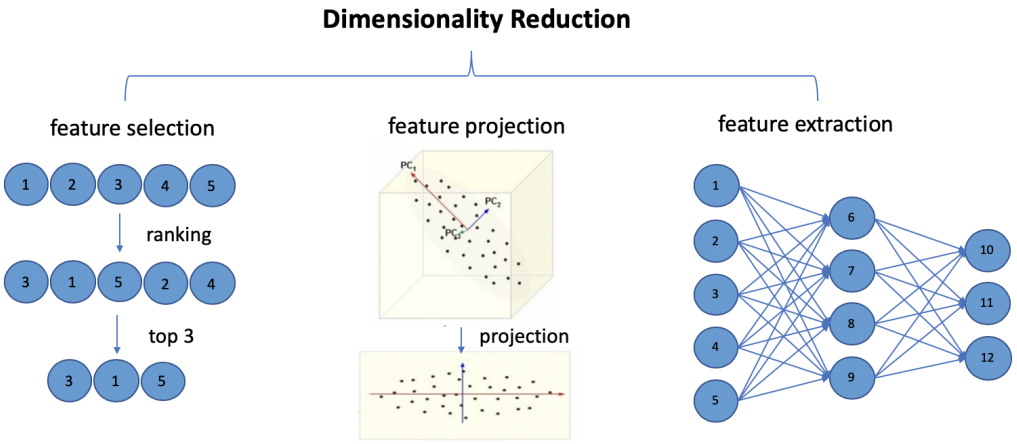

Figure 2-4: An illustration of various dimensionality reduction models.

PCA can be intuitively understood as using a minimal number of dimensions to keep data information as much as possible.

Given a set of data points $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, where $\mathbf{x_i}$ is a D-dimensional vector, PCA aims at finding a projection matrix $\mathbf{W} \in \mathbb{R}^{D \times d}$ so as to map each sample $\mathbf{x}_i$ to a d-dimensional vector ($d < D$) as follows:

$$\hat{\mathbf{x}}_i = \mathbf{W}^\top \mathbf{x}_i. \tag{2.6}$$

The projection matrix $\mathbf{W}$ can be calculated by maximizing the determinant of the total scatter matrix $\mathbf{S}_t$ defined as follows:

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} |\mathbf{W}^\top \mathbf{S}_t \mathbf{W}| \quad \text{where} \quad \mathbf{S}_t = \sum_i^n (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^\top, \tag{2.7}$$

where μ is the mean vector of all samples. The solution of $\mathbf{W}$ contains the top d eigenvectors of the scatter matrix $\mathbf{S}_t$ with the largest eigenvalues. In practice, a threshold for the portion of the preserved information is commonly used to automatically determine the value of d. It is also worth mentioning that PCA often improves the model performance when used in machine learning, despite the information loss caused by the projection. A common assumption is that real-world data often contain noise that is better to be removed for statistical learning.

After the projection matrix $\mathbf{W}$ is learned, we can map any data samples in the training

set to the lower-dimensional space, as well as unseen testing samples. Due to its linearity, PCA is computationally efficient, but it is also limited by this linearity. PCA cannot preserve the structure of non-linear data, which motivates the development of kernel PCA, and other non-linear dimensionality reduction techniques.

Locally Linear Embedding

Locally linear embedding (LLE) [117] assumes high-dimensional data lie on a low-dimensional manifold that is globally non-linear. It further assumes that each data point and its neighbors lie on a locally linear patch of the manifold. The locally linear geometry of these manifold patches can be characterized by linear coefficients that can reconstruct each data point using its nearest neighbors. Such a local structure is expected to be valid after embedding so that the linear relations of data points learned in the input space can be fixed and used to derive the optimal coordinates in the embedded space. LLE is capable of generating non-linear embeddings, and its optimization does not involve any local minimal.

Given a set of n data points $\{\mathbf{x}_1, ..., \mathbf{x}_n\} \in \mathbb{R}^D$, LLE first reconstructs each data point $\mathbf{x}_i$ as a linear combination of its neighbors $\mathbf{x}_j$. The reconstruction error is defined as:

$$\mathbb{E}(\mathbf{W}) = \sum_{i=1}^{n} \left\| \mathbf{x}_i - \sum_{j=1}^{n} w_{ij} \mathbf{x}_j \right\|^2 . \tag{2.8}$$

The reconstruction error is minimized subject to two constraints: 1) Each data point $\mathbf{x}_i$ can be reconstructed using only its k-nearest neighbors pre-defined by a distance measurement, *i.e.*, enforcing $w_{ij} = 0$ for any $\mathbf{x}_j$ not in the neighbor set; 2) The neighbor weights sum up to 1, *i.e.*, $\sum_{j=1}^{n} w_{ij} = 1$. The sum-to-one constraint can enforce the translation invariance of the weight matrix. The optimal reconstruction weights $\mathbf{W}^*$ can be obtained by solving a least square problem as discussed in [117]. In the embedded space, the linear reconstruction weights should still be valid so that LLE can learn a low-dimensional representation $\hat{\mathbf{x}}_i \in \mathbb{R}^d (d < D)$ for a given data point $\mathbf{x}_i$ by minimizing the embedding error:

$$\mathbb{E}(\hat{\mathbf{X}}) = \sum_{i=1}^{n} \left\| \hat{\mathbf{x}}_i - \sum_{j=1}^{n} w_{ij} \hat{\mathbf{x}}_j \right\|^2 . \tag{2.9}$$

This error is defined as the locally linear reconstruction error just like Eq.(2.8). The optimal embedding $\mathbf{X}^*$ can be obtained by solving a $n \times n$ sparse eigen-decomposition of the matrix

$\mathbf{M}$:

$$\mathbf{M} = (\mathbf{I} - \mathbf{W})^\top (\mathbf{I} - \mathbf{W}), \tag{2.10}$$

where $\mathbf{I}$ is the identity matrix of size $n \times n$. The d eigenvectors corresponding to the smallest eigenvalues provide an ordered set of orthogonal coordinates that can be used for the embedding. It can be seen that both PCA and LLE end up with a similar eigenvector problem. It turns out that the final embedding of many dimensionality reduction approaches is obtained by solving an eigenvector problem, including Laplacian Eigenmaps to be discussed next.

Laplacian Eigenmaps

Laplacian Eigenmaps (LE) [11] is another popular approach for non-linear dimensionality reduction. It also holds the manifold assumption similar to LLE that high-dimensional data lie on a low-dimensional manifold. The basic idea of LE is that the neighbor points in the input space should still be neighbors after projection. In this sense, LE is another "local" method that is trying to preserve the local structure.

Given a set of n data points of the form $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$, where $\mathbf{x}_i$ is a D-dimensional vector, the target of LE is to find another set of points $\{\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_n\} \in \mathbb{R}^d$ (d $<$ D) such that $\hat{\mathbf{x}}_i$ "represents" $\mathbf{x}_i$. The first step is to model neighboring relations in the input space. LE starts with a weighted affinity graph in which only the neighboring nodes are connected by the Gaussian kernel weighted edges. This can be represented by a weight matrix $\mathbf{W}$, such that if $\mathbf{x}_i$ and $\mathbf{x}_j$ are neighbors, then $W_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$ and otherwise $W_{ij} = 0$. The weight matrix $\mathbf{W}$ characterizes the adjacency in a way that $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar if W_{ij} is large. The next step is to use $\mathbf{W}$ to select a mapping. LE thinks that a reasonable criterion for selecting a "good" mapping is to minimize the following objective function:

$$\sum_{i,j} \|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\|^2 W_{ij}, \tag{2.11}$$

under certain constraints. The objective function Eq.(2.11) imposes a large penalty if neighboring points are projected far away. It therefore enforces that neighboring points are still neighbors after projection. To minimize the objective function, it is first shown that:

$$\frac{1}{2}\sum_{i,j}\|\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j\|^2 W_{ij}=\hat{\mathbf{x}}^\top\mathbf{L}\hat{\mathbf{x}}, \tag{2.12}$$

where $\mathbf{L}=\mathbf{D}-\mathbf{W}$ is the unnormalized graph Laplacian. Note that $\mathbf{W}$ is symmetric and $D_{ii}=\sum_j W_{ij}$. Thus,

$$\begin{aligned}\hat{\mathbf{x}}^\top\mathbf{L}\hat{\mathbf{x}}&=\hat{\mathbf{x}}^\top\mathbf{D}\hat{\mathbf{x}}-\hat{\mathbf{x}}^\top\mathbf{W}\hat{\mathbf{x}}\\&=\sum_i\hat{\mathbf{x}}_i^2D_{ii}-\sum_{ij}\hat{\mathbf{x}}_i\hat{\mathbf{x}}_jW_{ij}\\&=\frac{1}{2}(\sum_i\hat{\mathbf{x}}_i^2D_{ii}-2\sum_{ij}\hat{\mathbf{x}}_i\hat{\mathbf{x}}_jW_{ij}+\sum_j\hat{\mathbf{x}}_j^2D_{jj})\\&=\frac{1}{2}\sum_{i,j}\|\hat{\mathbf{x}}_i-\hat{\mathbf{x}}_j\|^2W_{ij}.\end{aligned} \tag{2.13}$$

Therefore, minimizing the objective function Eq.(2.11) can be represented as:

$$\arg\min_{\hat{\mathbf{x}}}\hat{\mathbf{x}}^\top\mathbf{L}\hat{\mathbf{x}}\qquad\text{s.t.}\qquad\hat{\mathbf{x}}^\top\mathbf{D}\hat{\mathbf{x}}=1,\quad\hat{\mathbf{x}}^\top\mathbf{D}\mathbf{1}=0. \tag{2.14}$$

The constraint $\hat{\mathbf{x}}^\top\mathbf{D}\hat{\mathbf{x}}=1$ removes an arbitrary scaling factor and $\hat{\mathbf{x}}=1,\quad\hat{\mathbf{x}}^\top\mathbf{D}\mathbf{1}=0$ eliminates the trivial solution [11]. The optimal solution is once again obtained by solving an eigenvector problem:

$$\mathbf{L}\hat{\mathbf{x}}=\lambda\mathbf{D}\hat{\mathbf{x}}. \tag{2.15}$$

It can be seen that both LLE and LE are motivated by preserving local structure on the manifold, and they end up with solving a similar eigenvector problem. Due to the graph interpretation, LE also draws a tight connection to spectral clustering [104] and Normalized Cuts algorithm [123]. In Chapter 4, we put all these algorithms into one framework, which is referred as the spectral problem. We discuss the limitation of these eigenvector related approaches, typically on the cubic computational cost, and propose an approximated solution to address such computational complexity.

2.3 Semi-Supervised Learning

Semi-supervised learning (SSL) refers to learning from both labeled points and unlabeled points. It falls between supervised learning (with fully labeled points) and unsupervised learning (without any labeled points). SSL is of great interest as it is often difficult, expensive, or time-consuming to acquire labeled data. In contrast, unlabeled data are relatively easy to obtain in terms of the involvement of experienced human annotators. Also, it has been observed by many researchers that unlabeled data used in conjunction with a small amount of labeled data can improve the learning accuracy. As it requires less human efforts and produces better accuracy, SSL draws considerable attention both in theory and in practice.

SSL can be broadly categorized into two groups, namely transductive and inductive. A learner is transductive if it only works on the training data (including labeled and unlabeled points as in SSL), but cannot handle unseen data. Inductive learner, on the other hand, is capable of learning a function that maps the input to the output for the whole space so that it can be used to make predictions on both training and testing data.

Popular SSL approaches include Expectation-Maximization (EM) with generative mixture models, self-training, co-training, transductive support vector machine (TSVM), graph-based semi-supervised learning (GSSL). Each approach has its own advantage and drawbacks, and practitioners have to select models that best fit their data. For example, EM-based generative models should be used if the data are drawn from certain probability distributions. Self-training could be used as a wrapper method if a complicated supervised learner has been built. TSVM is a good option if naive SVM works well for the labeled part. GSSL methods could be considered if there is no parametric information or other prior knowledge about the data. Below we discuss these approaches in more detail.

2.3.1 EM with Generative Models

EM-based generative models are probably the earliest SSL method. It assumes a generative model on the joint probability distribution as

$$p(x, y) = p(y)p(x|y), \tag{2.16}$$

where $p(x|y)$ is an identifiable mixture distribution, such as Gaussian mixture models. The components of the mixture distribution can be identified if there is a large amount of unlabeled data. Ideally, given one labeled points per component, the mixture distribution can be fully determined.

EM algorithm was used for text classification, and they found that unlabeled data improved accuracy [106]. A reweighing scheme was proposed in [45], which provides a provably unbiased estimator for arbitrary distributions. Variational inference and deep neural networks have been also brought into semi-supervised generative models [76].

The limitation of these generative models is on the model assumption itself. They make a strong assumption on the probability distribution. If the mixture model assumption fits the data, unlabeled data is guaranteed to improve the accuracy [20]. On contrary, it may hurt the performance if the model assumption is wrong, which has been observed in [170].

2.3.2 Self-training and Co-training

Self-training is probably the most intuitive SSL approach. The basic idea is to first train a classifier with the small amount labeled data, and the classifier is then used to make predictions on the unlabeled data. Among a large amount of unlabeled data, those who have been classified with the highest confidence are added to the training set. The classifier is retrained by the augmented training set, and the procedure repeats until converge. Self-training, as a wrapper method, is useful when there is a sophisticated classifier trained on the labeled data that is able to produce reasonable predictions.

Self-training was used for word sense disambiguation, a natural language processing task, *e.g.*, predict the word "plant" means a factory or a living organism in a specific context [157]. It was also applied to an object detection system, in which it showed a significantly improved accuracy [116]. PN-learning proposed in [74] extended self-training by adding structural constraints to the unlabeled data. Synthetic examples was proposed to aid classifiers based on the idea of oversampling and positioning adjustment [133]. Self-training was used in conjunction with decision trees in [129].

Co-training [19] is similar to self-training since it also makes use of the classifier's prediction to enhance itself, except that co-training involves two classifiers instead of one, as shown in Fig.2-5. It assumes that data points can be represented by two different feature sets (or views) that provide complementary information about the data. Ideally, co-training

requires the two views to be conditionally independent, and each view is sufficient to train a reasonable classifier, *i.e.*, the class of samples can be accurately predicted using one view alone. Initially, two classifiers are trained on the labeled data of the two feature sets, respectively. The most confident predictions of each classifier on the unlabeled data are then used to construct additional training data for the other classifier. Each classifier is then retrained by the augmented training set, and the procedure repeats.

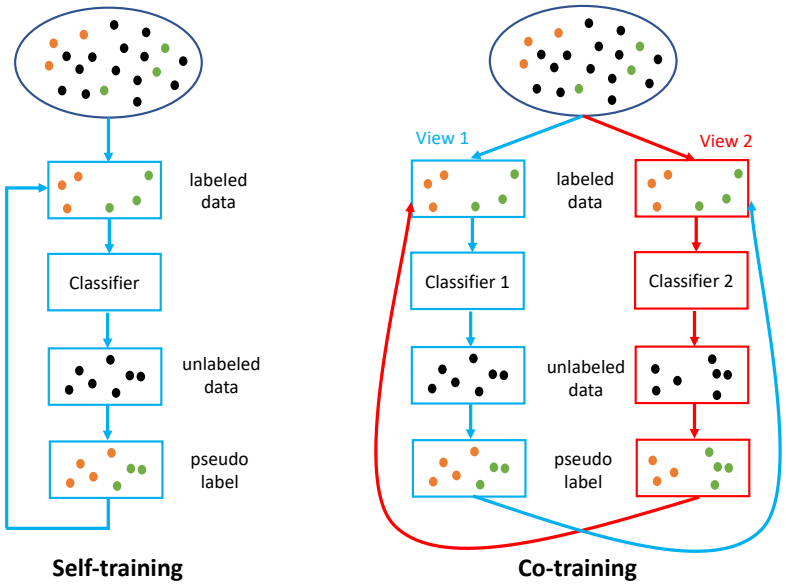

Figure 2-5: An illustration of basic pipeline of self-training and co-training models.

Extensive empirical experiments were performed by [105] to analyze the behaviors of co-training. They showed that co-training performs well if the conditional independence assumption indeed holds. They also demonstrated that in the case of only one feature set is available, co-training with two random artificial feature splits still improve the accuracy compared to the classifier training on the labeled data only, though not as much. The idea of co-training was applied in multi-view spectral clustering [81]. A variant of co-training that learns the feature splits was proposed and used for domain adaption [24]. A recent work [89] combined co-training and self-training to graph convolutional networks for semi-supervised learning.

2.3.3 Transductive Support Vector Machine

Transductive support vector machines (TSVMs) is an extension of the classic support vector machine (SVM) with the help of unlabeled data for finding decision boundary. The goal of a standard SVM is to find a linear boundary (or a hyperplane) that maximizes the margin between labeled points from different classes. Often, data is not linearly separable in the original finite-dimensional space. For this reason, SVM is usually used in conjunction with a kernel function, such as the Gaussian kernel, that maps data points to a higher or infinite-dimensional space in which the separation is presumably easier. However, the decision boundary, even in the kernel space, may not generalize well if the training data is scarce. This motivates the transductive SVM that makes use of unlabeled data. In a TSVM, the goal is to find labels of the unlabeled data so that a linear boundary has the maximum margin on both labeled and unlabeled points. With a large amount of unlabeled data, it is believed that the decision boundary generalizes well, as it has the smallest generalization error bound on the unlabeled data [136]. Intuitively, unlabeled data reveals the dense regions that the decision boundary should not go through, as shown in Fig.2-6.

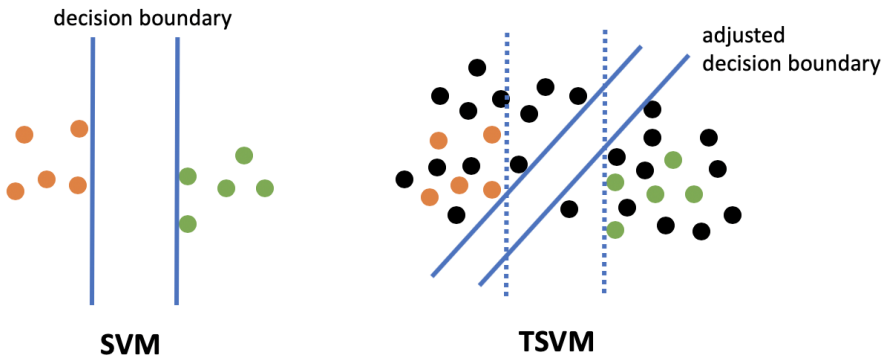

Figure 2-6: An illustration of how unlabeled data (black dots) adjust the decision boundary of SVM to avoid dense regions.

Not like SVM, finding the exact solution of TSVM is NP-hard. Researchers have been working on solving it approximately. A greedy local search was used in [73] to approximate the optimal solution and applied it to text classification. The idea of low-density separation based on gradient descent was proposed in [22]. A semi-definite programming approach that works for both TSVM and unsupervised SVM was also proposed in [151]. Researchers in [21] aimed at obtaining the global optimal solution based on branch-and-bound techniques.

2.3.4 Graph-based Semi-Supervised Learning

Graph-based semi-supervised learning (GSSL) is another family of SSL that has recently attracted much attention due to its superior performance and computational efficiency. GSSL methods define a node-edge graph that represents both labeled and unlabeled data points as nodes and builds undirected edges between these nodes with edge weights reflected as pairwise affinity. The labels are then propagated from the small portion of labeled nodes to the rest of unlabeled ones by various GSSL algorithms. These methods usually assume the label smoothness on the graph, which is closely related to the manifold assumption as the affinity graph can be seen as an approximation of the underlying data manifold. GSSL methods are non-parametric, discriminative, and transductive in nature [170].

There are many GSSL methods in the literature. It was formulated as a graph min-cut problem which targets at partitioning the graph so as to minimize the number of similar pairs of data points with different labels [18]. A Markov Random Walk approach was proposed to connect the pairwise affinity to the probability of walking from one node to another [127]. The Gaussian random fields and harmonic function (GFHF) method proposed in [173] formulated SSL as a regularized optimization problem. The local and global consistency (LGC) method [166] considered both local smoothness and global fitness by adapting the objective function proposed in GFHF. The greedy gradient max-cut (GGMC) [146] method formulated SSL as a bivariate optimization problem that is equivalent to a graph max-cut problem.

In this thesis, we focus on the GSSL methods based on the idea of graph regularization. These methods can be viewed as a function estimation problem in which the target function satisfies two conditions: 1) it fits to the given labels on the labeled nodes; 2) it is smooth on the whole graph, *i.e.*, the function should take similar values for the neighboring nodes, including both labeled and unlabeled ones. This can be formulated as a regularization framework that contains two terms, where the first term is a loss function that measures the fitness to the labeled points, and the second term is a regularizer that regularizes the smoothness of both labeled and unlabeled points, as shown below:

$$\arg\min_{\mathbf{F}} \left(\mathcal{J}_{fit}(\mathbf{F}) + \mu \mathcal{J}_{smooth}(\mathbf{F}) \right), \tag{2.17}$$

where $\mathbf{F}$ is the target function. For example, it could be a classification function of size $n \times c$ in classification, where F_{ij} represents the probability of the i-th samples belonging to the j-th class. Many aforementioned GSSL methods fall in this framework, and the main difference is on the choice of loss functions and regularizers.

As all these methods work on an affinity graph, we believe it is more important to construct an appropriate graph than to select among these methods. Affinity graph construction, however, is not a well-studied area, as we will discuss in the next section. This motivates us to learn affinity in the context of GSSL, which leads to the basic GSSL framework that contains two steps, namely affinity learning and label propagation, as discussed in Chapter 5. It is shown that many label propagation methods can be significantly improved to similar optimal performance, if an appropriate affinity graph is given. Moreover, additional performance gain can be achieved if the two processes are performed alternately, as shown in Chapter 6.

2.4 Deep Learning

All learning schemes discussed above have an implicit assumption that there exists a feature set of data, and it is good enough for the target problem. When this assumption does not hold in some cases, the success of machine learning methods heavily depends on finding good data representation (or features) to which they are applied. Due to this reason, a large portion of efforts in using machine learning algorithms has been devoted to the design of data preprocessing, transformation, and feature extraction that output a sensible representation of the data so as to support effective machine learning. Conventionally, this is achieved by a labor-intensive feature engineering process that takes advantage of human ingenuity and prior knowledge to produce hand-crafted feature descriptors, such as LBP, SIFT, and HOG used in computer vision. Despite its effectiveness of representing data, hand-crafted features require expensive efforts of highly skilled experts, and the acquired feature descriptors usually do not adapt to different problems, which hinders the efficient deployment of machine learning systems in various real-world applications.

Deep learning, on the other hand, makes use of the depth as the prior knowledge and constructs a deep architecture to extract layer-wise representations of the data. More importantly, the layer-wise structure of representation and downstream predictive tasks, such as

classifications, can be composed together to a neural network so that they can be trained simultaneously in an end-to-end manner. Features are no longer crafted by a hand-engineered process but learned automatically via a data-driven and task-driven approach. Feature extraction is not viewed as a preprocessing step of machine learning but treated as the machine learning task itself. It is sometimes referred to as representation learning, currently one of the most active research areas in the field of machine learning.

Typical deep learning architectures include multi-layer perceptrons, deep belief networks, auto-encoders, recurrent neural networks (RNNs), and convolutional neural networks. They have been successfully applied to various fields, such as computer vision [80], natural language processing [33], speech recognition [62], and autonomous driving [70], in which they have produced results comparable and in some cases surpassing performances of human experts. In this thesis, we focus on the convolutional neural networks due to its superior performance for computer vision problems.

2.4.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) [86] is a class of neural networks that uses convolution in place of general matrix multiplication in at least one of their layers [85]. It can be seen as a regularized version of multi-layer perceptrons (MLPs) that is specifically designed for grid data. For example, the time-series data can be thought of as 1D grid data taking samples at fixed time intervals, while images can be treated as a 2D pixel grid.

CNNs have achieved tremendous success in practical applications due to three ideas:

- Sparse connections. Vanilla fully-connected MLPs interact each neuron with all neurons in the next layer. The "fully-connectedness" of these networks not only increases the computational cost but also makes them prone to overfitting data. On contrary, CNNs achieve sparse connections by using a convolutional kernel smaller than the input. The sparse interactions increase computational efficiency and enable neurons to extract strong local features.

- Shared weights. MLPs use each element of the weight matrix only once, while each member of the kernel is used at every position of the input via a sliding window scheme in CNNs. Parameter sharing reduces the memory requirements dramatically, and it also enables translation robustness of networks, as the same feature can be detected

everywhere in the input.

- Multi-layer structure. Due to the high efficiency of the sparse connections and shared weights, CNNs are able to use a deep architecture. The multi-layer structure of CNNs allows it to detect high-level semantic features in the deep layers by assembling the low-level geometric features in the shallow layers. By going deep, a small convolutional kernel will have a larger respective field, *i.e.*, it can interact with a larger portion of the input. This allows CNNs to efficiently model complicated global interactions using local connections.

2.4.2 Graph Convolutional Networks

Despite its empirical success, CNNs are limited to data of the grid-like topology. For other non-structured data, such as graphs, existing CNNs are not applicable. However, graphs are a ubiquitous data structure, which is widely employed in many fields, such as social networks, citation networks, molecular structure, and biological protein-protein networks. It is definitely desirable to make use of CNNs in these graphical data. Such a desire motivates the recent research on the graph convolutional networks (GCNs) [77]. However, it is not straightforward to accomplish the convolutional operation on graphs. A typical graph is a non-Euclidean structure consisting of nodes and edges. The sliding window operation as in images cannot be used, as there is no obvious order of nodes. Local connection and weight sharing are also hard to define due to the lack of a fixed neighborhood structure.

Researchers in [57] attempted to implement a spectral convolution in the Fourier domain, where an eigenvector problem needs to be solved for the graph Laplacian matrix. It leads to intensive computations and non-spatially localized operation. A CNN was constructed on graphs with a localized constraint that forces neighbors to be within k hops so that the computation of eigenvectors was removed [31]. The spectral graph convolutions was later extended by further fixing the convolutional operation to $k = 1$, to alleviate the serious overfitting on local neighboring nodes with a large node degree [77]. It results in a clear layer-wise propagation formula that can be used to stack a multi-layer architecture similar to CNNs. They named their method Graph Convolutional Networks (GCNs) and demonstrated superior performances on semi-supervised classification. Since then, a large body of research has been presented in this subfield, including GraphSAGE [56], GAT [137], APPNP [78], to

name a few. Fig.2-7 shows an example of GCN that works on the graph directly with the multi-layer structure.

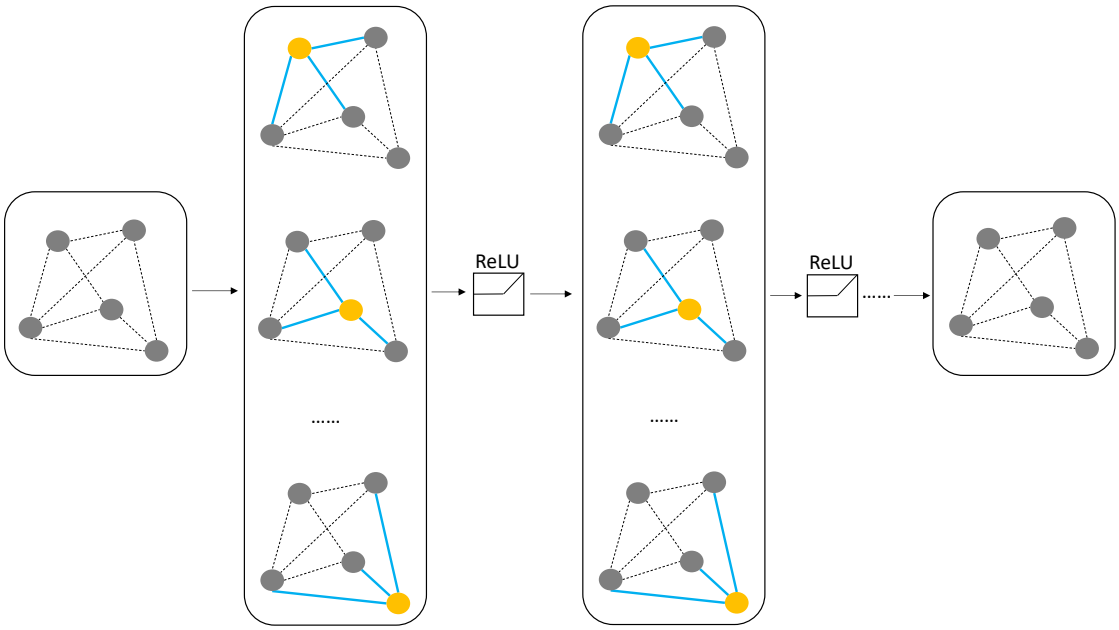

Figure 2-7: The graph convolutional network operates on the graph directly with the layer-wise structure.

Existing GCNs often focus on learning a discriminative representation so that classification can be easily achieved in the feature space. The essential idea of GCNs is to learn the representation of a target node by aggregating the representation of its neighboring nodes. In doing so, the topological graph information is encoded in the local neighborhood structure and utilized in a convolutional way. Neighbor aggregation, linear transformation, and non-linear mapping are integrated as a single layer operation. A deep GCN can be constructed by stacking multiple layers, and it can be trained with backpropagation by calculating a loss with respect to a particular objective function.

2.5 Affinity Graph

Affinity graph is the most frequently used data structure in this thesis. It is defined as an undirected vertex-edge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathbf{V}$ is the set of vertices that represent data points, and $\mathcal{E}$ is the set of vertex-to-vertex edges that describes pairwise adjacency. Edges are usually weighted by a weight matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$, where n is the number of data

points. $\mathbf{W}$ is positive and symmetric. $W_{ij} > 0$ reflects the affinity or similarity between data point $\mathbf{x}_i$ and $\mathbf{x}_j$, while $W_{ij} = 0$ means there is no edge between them. Fig.2-8 shows an example of an affinity graph and the corresponding weight matrix. By definition, a graph $\mathcal{G}$ can be fully represented by a weight matrix $\mathbf{W}$. Therefore, the problem of graph construction is formulated to build a $n \times n$ weight matrix $\mathbf{W}$.

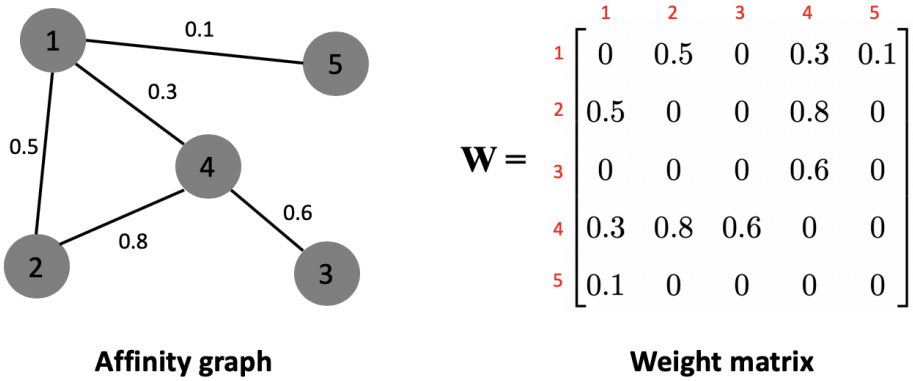

Figure 2-8: An example of the affinity graph and its associated weight matrix.

2.5.1 Graph Construction

Gaussian Kernel based Graphs

For pairwise affinities, the most commonly used approach is the kernel function. It defines the dot product of two vectors (two data points in our case) in a higher-dimensional kernel space, such as the Gaussian kernel. The Gaussian kernel, also known as the radial basis function (RBF) kernel, computes the dot product of $\mathbf{x}_i$ and $\mathbf{x}_j$ in an infinite-dimensional feature space.

$$W_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right). \tag{2.18}$$

where σ is a hyper-parameter that controls the decay of affinity values. σ can be viewed as a measurement of the underlying data density. An appropriate value of σ effectively controls how fast the affinity should change with respect to the density. By definition, it is challenging to pick up an optimal σ as it depends on the unbounded Euclidean distance. Often, practitioners have to spend a lot of time on tuning this hyper-parameter. In addition, even a reasonable σ is found, it is obvious that a unique global σ cannot handle all data

points as they are often not uniformly distributed. This motivates the usage of the locally adaptive Gaussian kernel.

Adaptive Gaussian Kernel based Graphs

The adaptive Gaussian kernels focus on finding different values of σ that adapt to the local structure, typically the local data density. It encourages bigger values of σ in the low-density area and smaller values of σ in the high-density area to enforce an adaptive affinity decay with respect to the data distribution. In this way, the obtained affinity values will be comparable to each other.

The self-tuning Gaussian kernel, proposed in [160], calculates a local scaling σ_i for each data point $\mathbf{x}_i$. Specifically, it calculates the pairwise affinities by an adaptive Gaussian kernel of the form:

$$W_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma_i \sigma_j}\right), \tag{2.19}$$

where σ_i represents the distance from $\mathbf{x}_i$ to its k-th neighbor $\mathbf{x}_i^k$, *i.e.*, $\sigma_i = \left\|\mathbf{x}_i - \mathbf{x}_i^k\right\|^2$. σ_i defined in this way can be seen as a measurement of the local statistics that differs from a data point to another. The selection of k from a typical range, such as $[5, 20]$, is much easier than selecting a value of σ from $[10^{-3}, 10^3]$ in the original Gaussian kernel. The authors suggested to use $k = 7$ to get a reasonable result for most cases. The idea of local adaption was extended even further in [145], where the mean distance to all of the kNN is considered, *i.e.*, $\sigma_i = \frac{1}{k}\sum_{\mathbf{x}_p \in \mathcal{N}_{\mathbf{x}_i}} \|\mathbf{x}_i - \mathbf{x}_p\|^2$, where $\mathcal{N}_{\mathbf{x}_i}$ is the kNN set of $\mathbf{x}_i$. It is believed that the mean distance is more robust to noise and outliers than the distance to k-th neighbor.

Tree Structure based Graphs

Tree-based graphs make use of the structure of decision trees to exploit pairwise affinities. The core idea is to use the overlap path from the root node to a leaf node traversed by two data points as a measurement of their similarity. Intuitively, two data points should be dissimilar if they split at the very beginning. On contrary, if two points reach the same leaf node by traveling through the same set of intermediate nodes, they should be considered as very similar.

This idea is used for the nearest neighbor search, where two data points are considered

as neighbors if they reached the same leaf node [92]. Instead of using a single tree, they make use of an ensemble of trees, also known as random forests. Each tree split a different subset of data points based on a different subset of features, an idea known as bagging, so that subtle similarities in various feature subspaces are considered. The average similarity over all trees is believed to be more robust than the one obtained from a single tree.

This framework is later extended to exploit the whole tree hierarchy [171]. Instead of considering the leaf nodes only, it defines the similarity of two data points to be proportional to the number of tree nodes that they traversed together. In addition, it is proposed to assign larger weights to deeper nodes, as there are fewer data points in the deeper nodes so that they should share more similarities [171]. In this way, the pairwise affinity obtained from each tree is a comparable real number rather than a binary number, leading to a more accurate affinity value.

Sparse Representation based Graphs

State-of-the-art graph construction methods, especially for high-dimensional data, are usually self-expressive models based on sparse representation. The basic idea is to express each data point as a linear combination of all other points while regularizing the reconstruction coefficients with certain norms. The sparse constraint on the coefficients enforces using fewer data points for reconstruction. If the manifold assumption holds, *i.e.*, high-dimensional data lie on low-dimensional subspaces, it encourages the reconstruction with data points in the same subspace, corresponding to data points from the same class. The coefficients can then be used to construct the weight matrix.

The main difference of various approaches in this area is the choice of norm. For example, in sparse subspace clustering (SSC) [36], ℓ_1 norm is used to encourage many zero coefficients, which results in the following optimization problem:

$$\min_{\mathbf{C}} \|\mathbf{X} - \mathbf{X}\mathbf{C}\|_{\mathbf{F}}^2 + \lambda \|\mathbf{C}\|_1 \quad \text{s.t.} \quad \text{diag}(\mathbf{C}) = \mathbf{0}, \tag{2.20}$$

where λ controls the trade-off between the two terms, and the constraint $\text{diag}(\mathbf{C}) = \mathbf{0}$ eliminates the trivial solution that each data point uses itself for reconstruction. Similarly, the low rank representation (LRR) was proposed in [94], which seeks the lowest rank representation to reconstruct each data point as a linear combination of all others. The rank function

is then relaxed to the nuclear norm, which gives the following optimization problem:

$$\min_{\mathbf{C}} \|\mathbf{X} - \mathbf{X}\mathbf{C}\|_{2,1} + \lambda \|\mathbf{C}\|_* , \tag{2.21}$$

where $||\cdot||_*$ represents the nuclear norm, *i.e.*, the sum of the singular value of the matrix. $\ell_{2,1}$ norm is used for reconstruction error as it is better suited for corrupted data. The constraint on the diagonal elements of the coefficient matrix is not necessary here, as it optimizes all data points jointly.

2.5.2 Graph Sparsification

After a graph is constructed, it is essential to sparsify its edges. Graph sparsification can improve computational efficiency as well as the accuracy of the downstream applications by providing the robustness to noise. In addition, graph construction methods, such as the Gaussian kernel, cannot produce reliable pairwise affinities for data points that are far away, as it is difficult to capture the geodesic distance on the manifold especially in the high-dimensional space.

Starting with a fully connected graph represented by a full weight matrix $\mathbf{W}$, the graph sparsification can be viewed as multiplying a binary matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$ element-wise. $B_{ij} = 1$ means the edge between $\mathbf{x}_i$ and $\mathbf{x}_j$ is kept, while $B_{ij} = 0$ represents the edge is removed. The goal is to find such a matrix $\mathbf{B}$. Next, we discuss the choices of $\mathbf{B}$.

The ϵ-ball Neighbor Graph

The ϵ-ball neighbor graph uses a distance threshold ϵ to select neighbors. For each data point $\mathbf{x}_i$, it can be seen as drawing a ball with a radius of ϵ. $B_{ij} = 1$ if $\mathbf{x}_j$ falls in this ball, otherwise $B_{ij} = 0$. The ϵ-ball neighbor graph has a good geometrical intuition, but it is not robust to data density anomalies [146].

The k-nearest Neighbor Graph

The k-nearest neighbor (kNN) graph sets $B_{ij} = 1$ if $\mathbf{x}_j$ is among the kNN of $\mathbf{x}_i$. It can be achieved by either seeking kNN first for all data points using any NN-search algorithm or computing the weight matrix $\mathbf{W}$ and setting all but the largest k elements in each row to zero. kNN graph is probably the most commonly used graphs, due to its simplicity of im-

plementation, hyper-parameter tuning, and robustness to many kinds of data imperfection, such as noise, outliers, and density anomalies.

The b-matching Graph

The b-matching graph can be seen as a symmetric extension of the kNN graph. Recall the way of obtaining kNN graph, where we enforce zero for all but the largest k elements in each row of $\mathbf{W}$, this may result an asymmetric matrix that is not desirable in many occasions. The b-matching method, which generalizes the linear assignment problem, produces a symmetric weight matrix directly by seeking a binary matrix that satisfies the symmetric constraint, leading to the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{B}} \sum_{ij} B_{ij} D_{ij} \\ &\text{s.t.} \quad \sum_{j} B_{ij} = k, B_{ii} = 0, B_{ij} = B_{ji}, \forall i, j \in 1, ..., n, \end{aligned} \tag{2.22}$$

where $\mathbf{D}$ is the matrix that represents pairwise distances. This gives us a symmetric neighbor indicator $\mathbf{B}$ without post-processing. However, minimizing Eq.(2.22) requires $\mathcal{O}(n^3)$ computational cost which is impractical for a dataset of medium size. A faster alternative by loopy belief propagation is discussed in [67].

2.6 Diffusion Process

The so-called diffusion process is a dynamic mechanism that propagates information on the graph from nodes to nodes or edges to edges. It is a widely used technique in the field of image retrieval, where the goal is to retrieve the most similar images from a potentially large database, given a query image. Conventionally, image retrieval is achieved by calculating the similarities/distances between all the images with the query and returning the top k images that are the most similar to the query image. Obviously, the success of these retrieval systems depends on an accurate calculation of pairwise distances. Such kind of retrieval systems cannot perform well for high-dimensional data as the underlying manifold is completely ignored by the pairwise measurement. Diffusion processes, on the other hand, re-evaluate the pairwise relationships in the context of all other members in the database. Given an

initial affinity graph, it diffuses the affinities with respect to the manifold approximated by the graph. Intuitively, while calculating the similarity of a and b, it is observed that both a and b are very similar to c. The similarity between a and b are therefore increased due to this observation.

More formally, a diffusion process can be interpreted as Markov random walks on the graph. Given a graph represented by a weight matrix $\mathbf{W}$, we first define a random walk transition matrix as follows:

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}, \tag{2.23}$$

where $\mathbf{D}$ is the diagonal degree matrix obtained from $\mathbf{W}$, $D_{ii} = \sum_{j=1}^{n} W_{ij}$. By the definition, $\mathbf{P}$ is a row-stochastic matrix whose rows sum up to 1, and P_{ij} represents the probability of walking from node i to node j. Given an initial probability distribution state $\mathbf{f}_0 \in \mathbb{R}^{1\times n}$, the distribution vector $\mathbf{f}_t$ after t steps of random walks can be defined as:

$$\mathbf{f}_t = \mathbf{f}_{t-1}\mathbf{P} = \mathbf{f}_{t-2}\mathbf{P}\mathbf{P} = \ \ ... \ \ = \mathbf{f}_0\,\mathbf{P}^t. \tag{2.24}$$

Apart from the explicit formula, the recurrence relation of the distribution sequence can be given as:

$$\mathbf{f}_{t+1} = \mathbf{f}_t\,\mathbf{P}. \tag{2.25}$$

This defines an update formula for a single step of the diffusion process. Note that this iterative method relates to the power iteration method which approximates the eigenvector of a matrix. Indeed, the iterative process Eq.(2.25) converges to the left eigenvector the transition matrix $\mathbf{P}$ with corresponding eigenvalue 1 [35].

The random walk model was then extended to one of the most widely used retrieval systems: the Google PageRank system [108]. This system was built for ranking millions of webpages in terms of human interest. To make use of the random walk type of diffusion process, webpages are represented by nodes, and hyperlinks are used to define the edges. The main innovation of the PageRank system is that a user may not always navigate by clicking the links on the webpages, as he/she might jump to a random webpage. The random walk model is modified so that a walker takes a standard step of random walking with a

probability α, while it can also jump to an arbitrary node with the probability $1-\alpha$. This can be characterized by the following update rule:

$$\mathbf{f}_{t+1} = \alpha\, \mathbf{f}_t\, \mathbf{P} + (1-\alpha)\mathbf{y}, \tag{2.26}$$

where $\mathbf{y}$ defines the prior probability of jumping to the corresponding nodes. The final ranking of the webpages can be obtained by iterating this updated formula until it converges. It is free to initialize the jumping probability $\mathbf{y}$. For example, it can be fixed to a uniform vector as in [108], which is referred to as the global PageRank. It also can be initialized differently for each user to enable personalized random jump preferences, and this is referred to as the personalized PageRank. The PageRank system Eq.(2.26) can be written in matrix form as:

$$\mathbf{A}_{t+1} = \alpha\, \mathbf{A}_t\, \mathbf{P} + (1-\alpha)\mathbf{Y}, \tag{2.27}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ contains the ranks of interest, and $\mathbf{Y} \in \mathbb{R}^{n\times n}$ is a stacking of all jumping vectors $\mathbf{y}$.

Ranking on Manifolds method [168] used an update strategy very similar to Eq.(2.27) as in the PageRank. The main difference is the definition of the transition matrix:

$$\mathbf{P}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}. \tag{2.28}$$

$\mathbf{P}_{\text{sym}}$ uses a different normalization that leads to a symmetric version of the transition matrix. It corresponds to the different version of the graph Laplacian as shown in Section 2.2.1.

Researchers in [154] highlighted the importance of using locally constrained diffusion process (LCDP). They proposed to consider the kNN sets of two points when measuring the affinity between them, which leads to the following diffusion strategy:

$$\mathbf{A}_{t+1} = \mathbf{P}\, \mathbf{A}_t\, \mathbf{P}^\top. \tag{2.29}$$

A variant of LCDP was proposed in [155], where an additional term is added, resulting in the update formula as follows:

$$\mathbf{A}_{t+1} = \mathbf{P}\,\mathbf{A}_t\,\mathbf{P}^\top + \mathbf{I}, \tag{2.30}$$

where $\mathbf{I}$ is the identity matrix of size $n \times n$. More importantly, they showed that the iterative formulation Eq.(2.30) is guaranteed to converge to a closed-form solution:

$$\mathbf{A}^* = vec^{-1}((\mathbf{I} - \mathbb{W})^{-1} vec(\mathbf{I})), \tag{2.31}$$

where $vec : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ is an operator that stacks the columns of a matrix one by one to a column vector, and its inverse exists as denoted by vec^{-1} which converts a vector to a matrix. $\mathbb{W}$ is the Kronecker product of the weight matrix $\mathbf{W}$ with itself, *i.e.*, $\mathbb{W} = \mathbf{W} \otimes \mathbf{W}$, which can be viewed as a tensor product graph (TPG) [155]. Therefore, the diffusion process in the form of Eq.(2.30) can be seen as spreading the context information in a higher-order graph, which in turn leads to larger capacity of modeling the data manifold in the high-dimensional space.

Despite the superior performance of TPG, it is not backed up by any mathematical justification. A recent work [7] attempted to build a connection between the iterative formulation of TPG Eq.(2.30) and an optimization framework. They found that the following regularization framework has the same close-form solution (as in Eq.(2.31)):

$$\min_{\mathbf{A}} \frac{1}{2} \sum_{i,j,k,l=1}^{n} W_{ij} W_{kl} \left(\frac{A_{ki}}{\sqrt{D_{ii} D_{kk}}} - \frac{A_{lj}}{\sqrt{D_{jj} D_{ll}}}\right)^2 + \mu \sum_{k,i=1}^{n} (A_{ki} - I_{ki})^2, \tag{2.32}$$

where μ controls the trade-off between the two terms. This regularization framework is related to the one introduced in Eq.(2.17), which contains two terms, namely the smoothness term and the fitness term. Similarly, the first term regularizes the smoothness of the pairwise affinity on the graph defined by $\mathbf{W}$, and the second term measures the fitness of the affinity $\mathbf{A}$ to the identity matrix $\mathbf{I}$. Note that the identity matrix $\mathbf{I}$ can be viewed as a prior of the pairwise affinity, in which the self-similarity is uniformly set to 1 and the between-sample similarity is set to 0. The modification of this prior is discussed in Chapter 5, where it is presented that non-uniform self-similarity and label information can be used to improve the quality of the prior.

In this chapter, all the related preliminary knowledge have been presented. Next, we discuss affinity learning in the unsupervised setting, where a sparse representation model

and the diffusion process are used to learn context-aware affinity.

Chapter 3

Unsupervised Affinity Learning for Subspace Clustering

As discussed in Chapter 2, unsupervised learning focuses on finding the underlying structures and identifying natural patterns based on data itself. As it does not require labeled data, unsupervised learning is considered to be a promising technique for achieving the general artificial intelligence (AI), which can behave adequately and perform tasks with little human interaction. Due to the lack of data-label pairs, data-data pairs are usually studied based on pairwise affinities. In this chapter, we mainly discuss affinity learning for subspace clustering in the computer vision domain, where data is often high-dimensional images, and the simple Gaussian kernel cannot work well.

3.1 Introduction

Many computer vision problems, such as image compression [65], motion segmentation [114] and face clustering [64], have to deal with high-dimensional data. The high-dimensionality of data not only results in the high computational costs of time and memory for related algorithms but more importantly, could also degrade their performances due to the inevitable noise and an insufficient number of samples with respect to its feature dimension, commonly referred to as the "curse of dimensionality" problem [13]. Fortunately, although many data are high-dimensional, their intrinsic dimension is often much lower than the dimension of

[0] Part of this chapter is reprinted, with permission, from [Li, Q., Liu, W., & Li, L. Affinity learning via a diffusion process for subspace clustering. *Pattern Recognition*, 2018. DOI: 10.1016/j.patcog.2018.07.002].

the ambient space [138], which has motivated many researchers to develop various techniques for finding low-dimensional subspaces for high-dimensional datasets. A natural idea is dimensionality reduction, such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), in which a common assumption is that the data lie in only one low-dimensional subspace. However, this assumption is generally not true in practice. High-dimensional data are usually drawn from several low-dimensional subspaces corresponding to multiple categories or classes to which the data belong [36]. In such scenarios, the task of clustering a high-dimensional dataset into multiple classes becomes a task of assigning each data point to its own subspace while preserving the underlying low-dimensional data structure, a problem known in the literature as *subspace clustering* [138].

In terms of machine learning and computer vision, existing subspace clustering methods can be categorized into four groups, namely algebraic methods [29, 139], iterative methods [96, 135], statistical methods [98, 115], and spectral clustering-based methods [23, 36, 49, 94, 165]. Among them, spectral-clustering based methods have become extremely popular due to their theoretical foundation and empirical success. They generally include two steps: 1) constructing an affinity matrix, and 2) applying spectral clustering to the affinity matrix. In this chapter, we focus on the first step as it is of the essence for the success of spectral clustering.

The most commonly used affinity learning method is the Gaussian kernel, where a global bandwidth σ has to be carefully tuned. Quite a few approaches have been proposed for learning a local adaptive σ [130, 131, 160]. The ideas are somewhat similar, as stated below: for each data point, use a local σ, which is defined by its distances to its k-nearest neighbors. However, when dealing with high-dimensional data, these methods are susceptible to the presence of noisy and irrelevant features [91]. Instead of using the Euclidean distance based Gaussian kernel, a tree-structure affinity learning method was proposed in [171]. With the whole travel path from the root node to a leaf node in consideration, this method derives the affinity values that are proportional to the number of nodes traveled together by a pair of data points. More recently, a novel affinity learning approach based on a fuzzy set was proposed in [91]. By exploiting discriminative feature subspaces, this method can capture and combine subtle similarity information distributed over feature subspaces, resulting in more accurate and robust affinity estimation.

State-of-the-art methods for constructing the affinity matrix in terms of subspace clus-

tering are based on the *self-expressiveness property* of the data [36], i.e., each data point in a union of subspaces can be efficiently reconstructed by a linear combination of all other data points: $x_j = \sum_{i \neq j} c_{ij} x_i$, where the coefficient c_{ij} is used to define the affinity between points i and j as $w_{ij} = |c_{ij}| + |c_{ji}|$. However, this leads to an ill-posed problem with many possible solutions [138]. In order to deal with this issue, the principle of *sparsity* is employed. Specifically, every point is represented by a sparse linear combination of all other data points while regularizing its coefficient matrix. The problem can be formulated as:

$$\min_{C} ||C|| \quad \text{s.t.} \quad X = XC, \quad \text{diag}(C) = 0, \tag{3.1}$$

where $X = [x_1, ..., x_N]$ is the data matrix, $C = [C_1, ..., C_N]$ is the coefficient matrix, and $|| \cdot ||$ is a properly chosen norm.

Which norm is a sensible choice? In Sparse Subspace Clustering (SSC) [36], the ℓ_1 norm is used as a convex relaxation of the ℓ_0 norm to promote the sparseness of C. As pointed out by [36, 102, 159], even though the coefficients obtained by SSC could be subspace preserving ($c_{ij} = 0$ when x_i and x_j are not in the same subspace), its connectivity may be poor (i.e., $c_{ij} = 0$ even if x_i and x_j are in the same subspace) as the coefficient matrix is sparse. In the Low-Rank Representation (LRR) [94] and Low-Rank Subspace Clustering [40], the nuclear norm $|| \cdot ||_*$ is applied as a convex relaxation of the rank function. One benefit of the nuclear norm is that the coefficient matrix is generally dense, in which the common connectivity issue for sparse representation-based methods is alleviated. However, such a representation matrix is known to be subspace preserving only when the subspaces are independent [158], which significantly limits its applicability since such condition is not satisfied generally.

In this chapter, the ℓ_1 norm is selected for sparsity constraints to retain the subspace preserving property, and a diffusion process is adopted for deriving the substructure of the affinity graph to mitigate the connectedness problem. Specifically, the first step is to learn a sparse affinity matrix by applying the ℓ_1 norm on the optimization problem of (3.1). A diffusion process is then adopted to spread the affinity values through the entire graph built upon the affinity matrix. Such a process can be interpreted as a Markov random walk, where the stochastic affinity (transition) matrix determines the probabilities of walking from one node to another. Since the affinity matrix learned by ℓ_1 norm is subspace preserving, the random walk on this matrix is guaranteed to be subspace constrained. Therefore, the connectivity

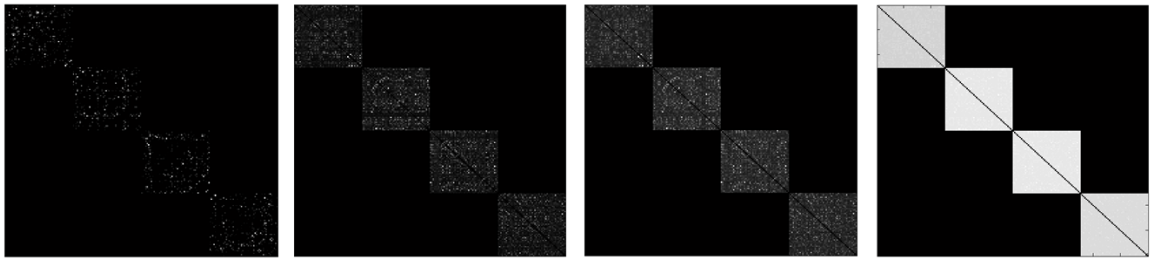

Figure 3-1: Visualization of affinity matrices after different iterations of the diffusion process: (a) C_0 is the original coefficient matrix obtained by SSC(ℓ_1 norm), (b)$\backsim$(d) C_t are the matrices after t steps of diffusion process applied on C_0. Clearly, as the diffusion process goes forward, the block-diagonal structure of the affinity matrix becomes more evident.

within the subspace is significantly enhanced while the subspace preserving property remains unaltered. An illustrative example is given in Fig.3-1. It clearly demonstrates that based on the sparse affinity matrix obtained by the ℓ_1 norm, the diffusion process could evidently improve the affinity within subspaces while retaining the sparsity between subspaces, yielding an affinity matrix with exactly block-diagonal structure.

The main contributions of this chapter can be summarized as follows:

- For subspace clustering, instead of combining and balancing different norms for subspace preserving and connectivity property, the ℓ_1 norm is used in the proposed approach, and the corresponding connectivity problem is alleviated by an efficient diffusion process. We argue that instead of using the sparse coefficients directly as the affinity values, the ℓ_1 norm serves more like a neighborhood selection in DSSC, while the affinity learning is managed by the diffusion process.

- From the diffusion point of view, we show that instead of choosing K nearest neighbors in the Euclidean space, the sparse property of ℓ_1 norm provides manifold-aware neighborhood construction, which is locally constrained in the corresponding subspaces. Applying the diffusion process on the affinity matrix obtained by ℓ_1 norm yields a more appropriate estimation of the true geodesic distances between data points on the manifold.

It should be noted that the ℓ_1 affinity matrix and KNN affinity matrix are both sparse. However, to construct the neighborhood structure, ℓ_1 is much more flexible. The reason is that the number of nonzero elements of each column of affinity matrix is variational, which

is controlled by the optimization method, while in the KNN affinity matrix, they are all fixed to K. Clearly, it is often inappropriate to set a uniform K for all data points, as they are often not uniformly distributed. Dense areas need a larger K to cover the local structure, while sparse areas need a smaller K.

This chapter is organized as follows. In Section 3.2, we briefly discuss the general model of subspace clustering and several existing diffusion strategies. In Section 3.3, we present and justify the proposed method. The interpretation of our method from the graph and the random walk view is also given. In 3.4, we compare the performance of the proposed method with existing methods through experiments on both synthetic and real data. The conclusion is presented in Section 3.5.

3.2 Related Work

State-of-the-art subspace clustering methods are often built on the self-expressiveness property. The choice of the regularizers for the coefficient matrix is the main difference among these methods. While different regularizers have their own advantages and drawbacks, it is proposed to use mixed norms [109, 147, 158]. The low-rank sparse subspace clustering (LRSSC) method [147] combines the ℓ_1 and nuclear norm regularizer as:

$$||C||_* + \lambda||C||_1, \tag{3.2}$$

where λ plays the trade-off between the two regularizers. Likewise, a mixture of ℓ_1 and ℓ_2 norms is proposed in [109, 158] as:

$$\lambda||C||_1 + \frac{1-\lambda}{2}||C||_2, \tag{3.3}$$

where the value λ represents the trade-off between the two norms. These methods basically attempt to bridge the gap between the subspace preserving and connectedness property by controlling the trade-off between different norms.

It is proposed in [42] to explicitly impose a block-diagonal constraint by fixing the rank of the Laplacian matrix. One benefit of this approach is that it can be applied to all the affinity construction methods directly. The Structured Sparse Subspace Clustering (SSSC) [88] integrates the two stages, affinity learning, and spectral clustering, into one unified

optimization framework. Their observation is that the clustering results can help the self-expressiveness model to yield a better affinity matrix.

All these subspace clustering methods are based on spectral clustering, where the affinity matrix is of great importance. Regardless of the various norms used by existing methods, the affinity values are often learned independently. Without considering the context information when the pairwise affinity is obtained, it is less likely to capture the underlying manifold. We hence propose to adopt a diffusion process to make full use of the local neighborhood structure iteratively. Each pairwise affinity is augmented and re-evaluated by their affinities with others, yielding an affinity matrix that can better represent the true geometry of the data manifold.

We propose to the diffusion process to mitigate the connectedness problem of ℓ_1 norm in terms of subspace clustering. By combining the ℓ_1 norm and diffusion process, the learned affinity matrix can be extremely effective when used in the scenario of subspace clustering. To the best of our knowledge, this is the first attempt to adopt the idea of diffusion process into this field. Since the idea of using a ℓ_1 norm for subspace clustering is originally from Sparse Subspace Clustering (SSC), we name our method as the Diffusion-based Sparse Subspace Clustering (DSSC).

3.3 Diffusion based Sparse Subspace Clustering

Before introducing the proposed approach, we first formulate the problem as below.

Problem 1 *(**Subspace clustering**). Given a set of data points $\{x_j \in \mathbb{R}^D\}_{j=1}^N$ drawn from an union of k subspaces $\{S_i\}_{i=1}^k$ of dimensions $d_i = dim(S_i), 0 < d_i < D, i = 1, ..., k$, the goal is to cluster these points into their corresponding subspaces.*

Sparse Subspace Clustering (SSC) attempts to solve the above problem based on the so-called self-expressiveness model, which states that each data point can be expressed as a linear combination of all other data points, i.e., $X = XC + E$, where C is the coefficient matrix and E is the matrix of error (noises or outliers). In order to guarantee the uniqueness of the solution, the sparsity constraint on C is invoked by ℓ_1 minimization, which leads to the optimization problem:

$$\min_{C} ||C||_1 + \lambda||E||_\ell \quad \text{s.t.} \quad X = XC + E, \quad \text{diag}(C) = 0, \tag{3.4}$$

where $||\cdot||_\ell$ represents the Frobenius norm or ℓ_1 norm to handle noise or outliers. With any ℓ_1 solver, e.g., ADMM, an optimal coefficient C can be obtained by solving the optimization problem (3.4), which is used to build a symmetric affinity matrix $|C| + |C^T|$ later on. The final data segmentation is obtained by applying spectral clustering using the affinity matrix.

As mentioned previously, the key difference among existing subspace clustering methods lies in the choice of the regularizer. Clearly, ℓ_1 norm of the coefficient matrix C is the core of SSC. It gives a unique optimal solution of C with a sensible property, sparsity. While the reconstruction of a single data point is forced to use as few coefficients as possible, it is believed that the data points located in the same subspace should be used as they share a better similarity. Nevertheless, the sparsity property could cause some problems. For clean data, SSC is guaranteed to be subspace-preserving, i.e., there are no connections between points from different subspaces. However, the within-subspace connections are usually sparse, i.e., C_{ij} could be zero even when x_i and x_j are in the same subspace. It is not a problem as long as the connections are still subspace-preserving. However, for noisy data, there is no theoretical guarantee that the nonzero coefficients correspond to points in the same subspace. It is then possible for SSC to make an inappropriate segmentation of the data since the within-subspace connections are weak while there are noisy connections between subspaces.

We argue that such a drawback exists in all SSC based algorithms as SSC seeks the sparse coefficients of each data point independently. The diffusion process is capable of dealing with such an issue by exploiting the contextual affinities. Given an affinity matrix $W = |C| + |C^T|$, where C is obtained by solving (3.4), we first normalize the row of W by $W = D^{-1}W$, where D is a diagonal matrix with $D_{ii} = \sum_{k=1}^{n} W(i,k)$, then classic diffusion process can be encoded into computing the power of the affinity matrix, which is

$$W_t = W^t, \tag{3.5}$$

where t corresponds to t steps of the diffusion process. It is obvious that such a process is sensitive to the step t. In order to make the diffusion process robust in terms of t, we can consider the accumulation of all ts. Thus, the diffusion process is reformulated as below:

$$W_t = \sum_{i=0}^{t} W^i, \tag{3.6}$$

where W is assumed to be nonnegative, and the sum of each row is smaller than 1. A stochastic matrix can be used to construct the matrix that satisfies the requirements. Note that the absolute eigenvalues are bounded by the largest row sum. Therefore, the maximum eigenvalues of W is less than 1, resulting in a converged closed-form given by $\lim_{t\to\infty} W_t = (I - W)^{-1}$ of (3.6), where I is the identity matrix.

In a graph representation, the relationship between samples is normally represented by pairwise affinities. Such naive pairwise representation is too simple to reveal the complex real-world data structure, especially when the data lie on a complex manifold, and squeezing the complex relationships into pairwise ones will undoubtedly result in information loss. A natural way of remedying this issue is to represent the data as a higher-order tensor graph, where the edge can connect more than two vertices. As demonstrated by several works [68,155,167], the higher-order tensor graph is indeed helpful for revealing the intrinsic relationships among data points and leads to better performance. Given a graph $G = (V, E)$ constructed from an affinity matrix W, the tensor product graph is defined as the Kronecker product of the original graph with itself, $\mathbb{G} = G \bigotimes G$. The corresponding affinity matrix is $\mathbb{W} = W \bigotimes W$. In particular, we have

$$\mathbb{W}(\alpha, \beta, i, j) = W(\alpha, \beta) \cdot W(i, j) = w_{\alpha,\beta} \cdot w_{i.j}. \tag{3.7}$$

Thus, if $W \in \mathcal{R}^{n\times n}$, $\mathbb{W} = W \bigotimes W \in \mathcal{R}^{nn\times nn}$. The diffusion process is then defined on the higher order tensor as

$$\mathbb{W}_t = \sum_{i=0}^{t} \mathbb{W}^i. \tag{3.8}$$

With the assumption that the maximum row sum of W is less than 1, we have

$$\sum_{\beta j} \mathbb{W}(\alpha, \beta, i, j) = \sum_{\beta j} w_{\alpha\beta} w_{ij} = \sum_{\beta} w_{\alpha\beta} \sum_{j} w_{ij} < 1. \tag{3.9}$$

Similar to (3.6), the process (3.8) also has a closed form solution,

$$\lim_{t\to\infty} \mathbb{W}_t = \lim_{t\to\infty} \sum_{i=0}^{t} \mathbb{W}^i = (I - \mathbb{W})^{-1}. \tag{3.10}$$

Since our goal is to learn a new affinity matrix W^* of size $n \times n$, it can be defined as

$$W^* = vec^{-1}((I - \mathbb{W})^{-1} vec(I)), \tag{3.11}$$

where I is the identity matrix and vec is an operation that transfers a matrix to a vector by stacking all the columns one after another. The inverse of vec is denoted as vec^{-1}.

While the tensor graph provides an adequate underlying structure of the data, it is impractical for large scale problems due to the high storage and computing cost. Therefore, we use an iterative algorithm for the diffusion process on a tensor graph. Firstly, we define $A_0 = W$ and the update strategy as

$$A_{t+1} = W A_t W^T + I, \tag{3.12}$$

where I is the identity matrix. The diffusion process becomes the iteration of (3.12) until convergence. To prove the convergence of (3.12), we transform (3.12) to

$$\begin{aligned} A_{t+1} &= W A_t W^T + I = W(W A_{t-1} W^T + I) W^T + I \\ &= W^2 A_{t-1} (W^T)^2 + W I W^T + I = ... \\ &= W^t W (W^T)^t + W^{t-1} I (W^T)^{t-1} + ... + I \\ &= W^t W (W^T)^t + \sum_{i=0}^{t-1} W^i I (W^T)^i. \end{aligned} \tag{3.13}$$

Since the sum of each row is assumed to be less than 1, $W^t = \mathbf{0}$, therefore we have $\lim_{t\to\infty} W^t W (W^T) t = 0$. Consequently,

$$\lim_{t\to\infty} A_{t+1} = \lim_{t\to\infty} \sum_{i=0}^{t-1} W^i I (W^T)^i. \tag{3.14}$$

As shown in [155], it can be proven by induction that

$$\lim_{t\to\infty} \sum_{i=0}^{t-1} W^i I (W^T)^i = vec^{-1}((I - \mathbb{W})^{-1} vec(I)). \tag{3.15}$$

Algorithm 2 Diffusion based Sparse Subspace Clustering (DSSC).

Input: A set of data points $X^{n\times d}$, the regularization parameter λ, the number of subspaces k.

1: Solve the ℓ_1-minimization problem 3.4 to get the coefficients c_i.
2: Form an affinity matrix $A = |C| + |C|^T$, where $|C_{ij}|$ is the absolute value of C_{ij}.
3: Normalize the row of W by $W = D^{-1}W$, where D is a diagonal matrix with $D_{ii} = \sum_{k=1}^{n} W(i,k)$.
4: Initialize the iteration $A_0 = W$, $t = 0$
5: **while** not converged **do**
6: $\quad A_{t+1} \leftarrow WA_tW^T + I$
7: $\quad t \leftarrow t + 1$
8: **end while**
9: Construct the corresponding Graph Laplacian matrix $L = I - D^{-1/2}AD^{-1/2}$, where D is a diagonal matrix with $D_{ii} = \sum_{k=1}^{n} A(i,k)$.
10: Form the matrix $V^{n\times k}$ by stacking the first k normalized eigenvectors of L corresponding to its k smallest eigenvalues.
11: Treat each row of V as a data point, and perform k-means to get the cluster labels.

Output: The cluster labels Y.

Hence we have

$$\lim_{t\to\infty} A_{t+1} = vec^{-1}((I - \mathbb{W})^{-1}vec(I)). \tag{3.16}$$

The iterative algorithm (3.12) produces the same affinity matrix as the diffusion process on tensor graph. Once the final affinity matrix is obtained, the clustering result can be achieved by performing spectral clustering as described in Algorithm 2.

3.3.1 Justification from the Graph View

With ℓ_1 minimization, SSC seeks sparse representation for each data point individually. The corresponding affinity graph reveals the pairwise affinity as the "shortest path" between them, which is susceptible to noise. The proposed DSSC derives the affinity by considering the "volume of paths" through a diffusion process. One significant property is, with ℓ_1 norm, the "volumes of paths" are restricted in subspaces, resulting in the enhancement of within-subspace affinity and the reduction of between-subspace affinity.

Fig.3-2 shows that for clean data, SSC is guaranteed to be subspace-preserving, i.e., there are no connections between subspaces. In such situations, spectral clustering can properly cut the graph. However, due to the sparse property of ℓ_1 norm, it is very likely that two points in the same subspace are not connected. A diffusion process can accurately complete

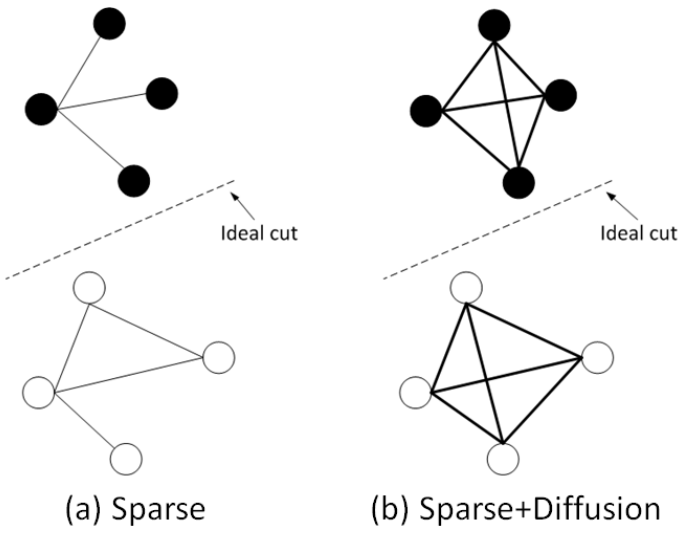

Figure 3-2: An example of affinity graph for clean data, obtained by different methods: (a) sparse, (b) sparse with diffusion. While the cuts remain the same, the within subspace connections are clearly enhanced after the diffusion process. The value of edge weights are reflected by the thickness of the edges.

the graph for each subspace in such cases, leading to more robust subspace graphs.

For noisy data, the advantage of diffusion becomes even more significant. As shown in Fig.3-3, when there are noisy edges between subspaces, spectral clustering might not be able to find the ideal cut of the graph. As the pairwise affinity is individually established in SSC, it can be difficult to distinguish "noisy edges" from "real edges". With DSSC, due to the ability to combine contextual information, the within-subspace affinities are evidently enhanced so that the noisy edges can be properly severed by spectral clustering. Real-world data are usually noisy, which explains the significant improvements of DSSC over SSC on real-world data sets, shown later in the experiments in Section 3.4.2.

3.3.2 Justification from the Random Walk View

As derived by [101], the objective function of spectral clustering can be expressed in the framework of the random walk as follows. For two disjoint subsets A, $\overline{A} \subset V$, assume that we run a random walk starting with X_0 in the stationary distribution π, we have

$$P(\overline{A}|A) = P(X_1 \in \overline{A}|X_0 \in A) \tag{3.17}$$

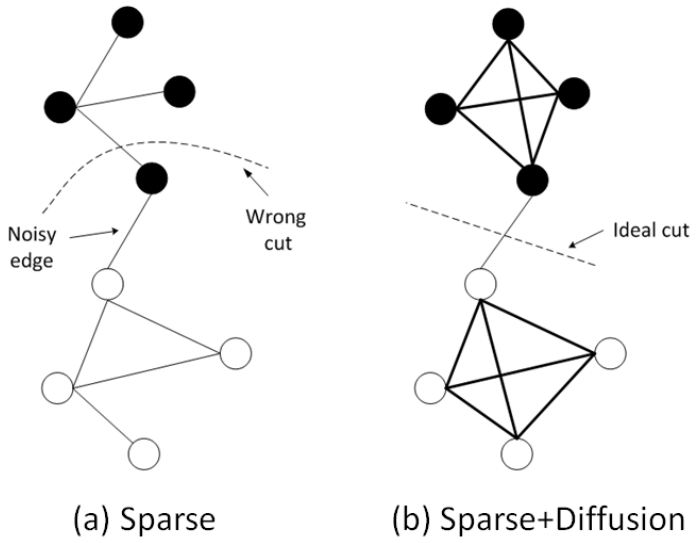

Figure 3-3: An example of affinity graph for noisy data, obtained by different methods: (a) sparse, (b) sparse with diffusion. By enhancing the within subspace connections, the diffusion process could help to find the ideal cut of the affinity graph. The value of edge weights are reflected by the thickness of the edges.

as the probability of the random walk transition from set A to set $\overline{A}$. Then, the goal of spectral clustering is to minimize the objective function

$$F = P(\overline{A}|A) + P(A|\overline{A}). \tag{3.18}$$

As mentioned previously, the proposed diffusion process can be interpreted as Markov random walks [127], where the stochastic affinity matrix is the transition matrix that defines probabilities for walking from one node to another. One crucial observation is that for an affinity graph, there will be many connections within a subspace, and fewer connections between subspaces. Therefore, if we start a walker at one node and randomly travel to a connected node, the random walk is more likely to stay within the same subspace than travel between different subspaces. Intuitively, diffusion processes can be imagined as random walks starting at every node on the graph. As the random walks continue, the probabilities of traveling between nodes within the same subspace increase, while the probabilities of traveling between subspaces decrease, which means $P_t(\overline{A}|A) < P(\overline{A}|A)$ as well as $P_t(A|\overline{A}) < P(A|\overline{A})$, where P_t is the transition probabilities after t steps of diffusion process. Let F^* be the objective of spectral clustering applied to the affinity matrix obtained by the diffusion

process. In conjunction with Eq.3.18, we have

$$F^* = P_t(\overline{A}|A) + P_t(A|\overline{A}) < P(\overline{A}|A) + P(A|\overline{A}) = F, \tag{3.19}$$

which explains why the diffusion processes can improve the performance of spectral clustering.

3.4 Experiments

The experiments are divided into three parts. In section 3.4.1, we compare the proposed DSSC to other existing affinity learning methods. As the proposed DSSC method uses ideas from both the diffusion and subspace clustering fields, it is necessary to examine how DSSC compares to existing methods in both fields, as in Section 3.4.2 and Section 3.4.3. The clustering results are evaluated using performance metrics: clustering accuracy and Normalized Mutual Information (NMI).

Clustering accuracy is defined as the ratio of the number of correctly clustered data points divided by the total number of points. Be aware that a permutation mapping function needs to be used to map the predicted cluster index to the ground-truth index before they can be compared. Clustering accuracy is calculated as:

$$\text{clustering accuracy} = \frac{\sum_{i=1}^{n} \delta\big(\hat{y}_i, map(y_i)\big)}{n}, \tag{3.20}$$

where n is the number of data points, $\hat{y}$ is the ground-truth class index, y is the predicted cluster index, and $\delta(x, y)$ is a function that equals to 1 if $x = y$, or 0 otherwise.

Normalized mutual information (NMI) is another widely used performance metric for clustering, which is defined as:

$$\text{NMI} = \frac{\sum_{l=1}^{c}\sum_{h=1}^{c} n_{l,h}\log(\frac{n n_{l,h}}{n_l \hat{n}_h})}{\sqrt{(\sum_{l=1}^{c} n_l \log\frac{n_l}{n})(\sum_{h=1}^{c}\hat{n}_h \log\frac{\hat{n}_h}{n})}}, \tag{3.21}$$

where c is the number of class, n_l denotes the number of data points in the l-th cluster, $\hat{n}_h$ is the number of pints in the h-th class, and $n_{l,h}$ represents the number of points in both the l-th cluster and h-th class. The larger the NMI, the better the performance.

Experimental settings. For the experiments of Hopkins 155 and Extended YaleB,

the regularization parameter λ is set to be the same as they are in the SSC [36] for a fair comparison. For all other experiments, the λ in DSSC and all other parameters in other methods are chosen by grid search, and those given the best results are used. The iteration number t in DSSC is set to be 200 for all experiments, which is found to be large enough to converge. Note that to speed up the process, one could also set up a convergence condition, such as $||A_{t+1} - A_t|| < \epsilon$, where $\epsilon = 1e-4$, and check this condition in every iteration. The number of subspaces k is set to be the exact number of classes for all methods.

Toy example. We first use a toy example to show how the proposed method could improve the affinity matrix by taking the context information into account. Suppose we have an affinity matrix W_{33} that contains affinity values of three data points, as shown in Fig.3-4 (a_{ij} represents the affinity value between i and j). For simplicity, we show the affinity value between 1 and 3, a_{13}, after one iteration. According to the updating strategy $A_{t+1} = WA_tW^T + I$, we have

$$a_{13}^{(2)} = a_{12} \cdot a_{23} \cdot (a_{11} + a_{22} + a_{33}). \tag{3.22}$$

It can be seen that initially $a_{13} = 0$, but after one step of the diffusion process, it starts to re-evaluate the affinity value by considering the context data (a_{12} and a_{23}). Intuitively, we know if A is similar to B and B is similar to C, then A should be somewhat similar to C, i.e., a_{13} should not be 0 when $a_{12}, a_{23} > 0$. From this toy example, it can be clearly seen that the diffusion process can augment the original affinity values by spreading the context affinities. Instead of considering each pair of data points individually, the affinity values obtained by the proposed method could leverage all affinity information of neighbor data points, yielding more precise affinity values. We stress its effectiveness when combined with ℓ_1 norm. Due to its sparsity constraint, ℓ_1 norm will produce many 0 affinities within subspace, resulting in poor clustering performance. By integrating the diffusion process, data pairs will share their neighbors and update their affinities iteratively, leading to more compact within-subspace affinities.

3.4.1 Comparison on Affinity Learning

The proposed algorithm DSSC is essentially an affinity learning method, where the affinity value is initialized by ℓ_1-norm and updated by the diffusion process. To evaluate the

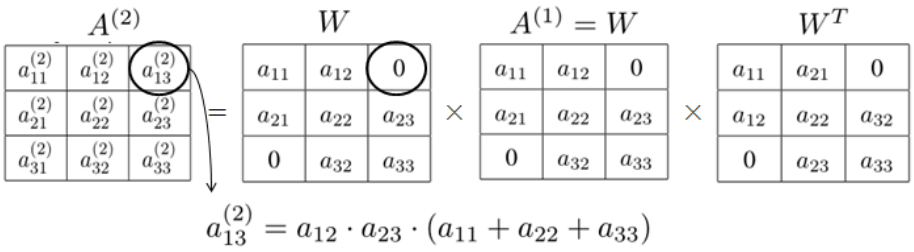

Figure 3-4: Example of showing how the diffusion process could improve the affinity matrix by leveraging the context information. $a_{13}^{(1)} = 0$, after one iteration $a_{13}^{(2)} = a_{12} \cdot a_{23} \cdot (a_{11} + a_{22} + a_{33})$.

effectiveness of DSSC on affinity learning, we compared it to several other affinity learning methods:

- **SC**. A widely used NJW spectral clustering [104], where the affinity is obtained by the Gaussian kernel with global bandwidth sigma.
- **STSC**. Self-tuning spectral clustering [160]. The affinity is obtained by the Gaussian kernel with an adaptive sigma, which is defined by the KNN distance.
- **RFSC**. Random forest based spectral clustering [171] uses a tree-structure affinity inference model, where the affinity value is proportional to the number of nodes traveled together by a pair of data points, starting from the root node to the leaf node.
- **AFSSC**. Fuzzy based affinity learning method [91], in which the affinity is defined by a data-driven fuzzy membership function. Instead of using all features in the ambient space, this model can capture and combine subtle proximity distributed in discriminative feature subspaces.

While the affinity matrices are learned by different algorithms, the results are obtained by applying the same NJW spectral clustering on them. The parameters in each method are chosen by grid search, as the same in [91].

Following the same settings, we redo two experiments in [91]. The first one is digits clustering on the widely used USPS dataset [69]. This dataset consists of numeric data sampled from the scanning of handwritten digits on the envelopes from the U.S. Postal Service. There are in total 9298 images, and each digit image is normalized to 16×16 grayscale, resulting in a feature vector with 256 dimensions. To fully investigate the performance with

respect to the number of clusters and data points, digit subsets {0,8}, {4,9}, {0,5,8}, {3,5,8}, {1,2,3,4}, {0,2,4,6,7} and the full set {0 ∼ 9} are used to test the algorithms.

Table 3.1 shows the clustering accuracies of different methods on the USPS datasets. It can be seen that the proposed DSSC outperforms all other competitors in all test cases. While the other methods only perform well only on the easy cases, i.e., {0,8} and {1,2,3,4}, DSSC achieves satisfactory results in all cases. Particularly, in the hard cases, significant accuracy improvements are observed, 14% and 12% in the case of {4,9} and {3,5,8}, respectively.

Table 3.1: Clustering accuracy on the USPS datasets.

Dataset	{0,8}	{4,9}	{0,5,8}	{3,5,8}	{1,2,3,4}	{0,2,4,6,7}	{0 ∼ 9}
SC	0.966	0.756	0.829	0.713	0.937	0.713	0.538
STSC	0.843	0.776	0.724	0.604	0.921	0.622	0.624
RFSC	0.872	0.738	0.814	0.767	0.945	0.737	0.698
AFSSC	0.985	0.781	0.902	0.833	0.981	0.805	0.722
DSSC	**0.991**	**0.922**	**0.975**	**0.950**	**0.987**	**0.979**	**0.850**

Table 3.2 shows the NMI of different methods on the USPS datasets. The results are consistent with clustering accuracy that DSSC outperforms all other methods in all test cases, and especially in the hard cases and multi-cluster cases with a large number of clusters, DSSC wins with a large margin compared to the second best.

Table 3.2: Clustering NMI on the USPS datasets.

Dataset	{0,8}	{4,9}	{0,5,8}	{3,5,8}	{1,2,3,4}	{0,2,4,6,7}	{0 ∼ 9}
SC	0.752	0.110	0.653	0.352	0.597	0.706	0.515
STSC	0.430	0.248	0.424	0.253	0.632	0.609	0.578
RFSC	0.649	0.235	0.607	0.474	0.854	0.782	0.652
AFSSC	0.912	0.268	0.682	0.496	0.920	0.834	0.775
DSSC	**0.920**	**0.607**	**0.877**	**0.797**	**0.946**	**0.924**	**0.824**

Next, to fully evaluate the robustness of the affinity matrices, we test the clustering performances using the KNN affinity matrix, i.e., given an affinity matrix W, we set $W_{ij} = W_{ij}$ if $j \in knn(i)$, otherwise, $W_{ij} = 0$. In the scope of spectral clustering, even though there is no theoretical result showing that which type of affinity matrix is better, it has been reported in [112, 171] that in practice KNN affinity matrix is more effective than the full affinity matrix, especially for noisy data. In our experiment, we use face images from Yale [10] and CMU-PIE [124]. The Yale dataset contains 165 grayscale images of 15 subjects,

and each subject has 11 images of the frontal face with various lighting conditions and minor expressions. CMU-PIE contains 41,368 images of 68 people, each with 13 different poses, 43 different illumination conditions, and 4 different expressions. We use all the frontal faces of the 68 people with no expression or pose, resulting in 20 images per person. Example images are shown in Fig.3-5 and Fig.3-6. All the images are cropped and down-sampled to 32×32 resolution. The raw pixel values are used as features.

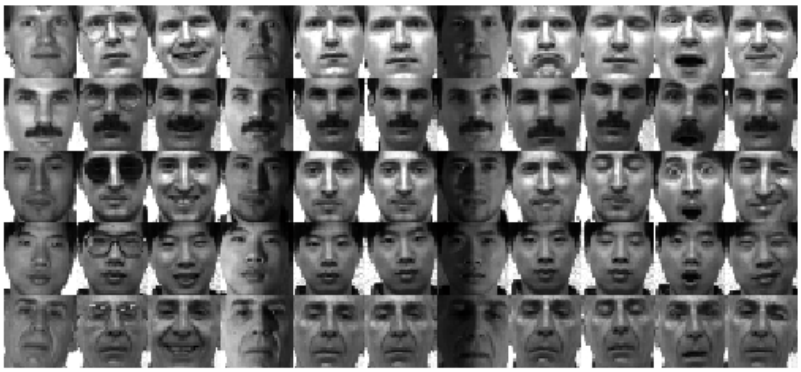

Figure 3-5: Example images from Yale datasets. Each row corresponds to one subject.

Figure 3-6: Example images from CMU-PIE datasets. Each row corresponds to one subject.

Fig.3-7 presents the clustering performances NMI with respect to a different number of neighbors in the affinity matrices. It can be seen that the proposed DSSC achieves the best results for all values of neighborhood size k. It is worth to mention that DSSC achieves significant improvements in the CMU-PIE dataset. One possible reason is that the face images of PIE used are purely frontal face without any poses or expressions (as shown in Fig.3-6), resulting in the satisfaction of our assumption that face images of a single subject

lie in a linear subspace. It also can be observed that as k increases, DSSC produces more robust results against other methods. This is very important in real applications where the number of samples in each class is unknown, or the numbers of samples are different for each class. Note that we observe the best clustering results can be obtained with a very small value of k (typically $k < 10$). This observation agrees with [112, 171]: in practice, the KNN affinity matrix is more effective than the full affinity matrix ($k = N$) for noisy data, which are face images in our case.

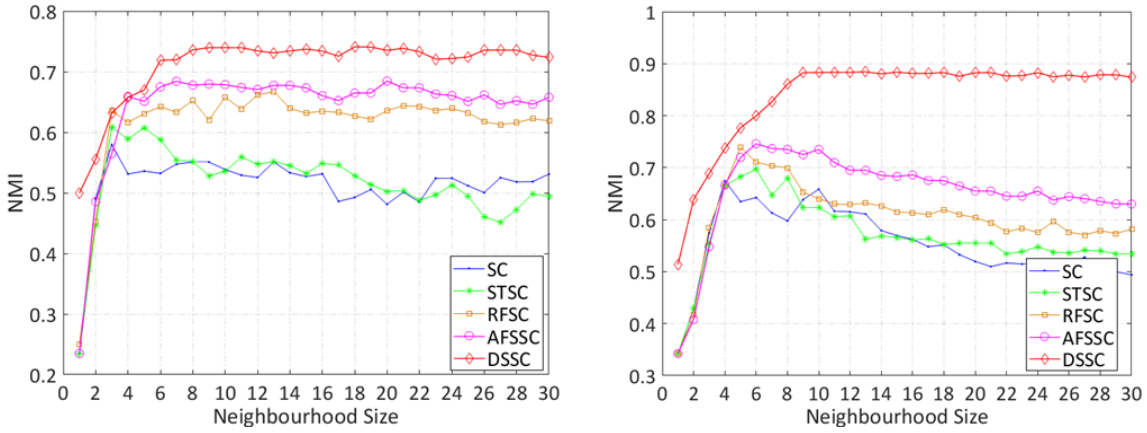

Figure 3-7: Clustering performance (NMI curve) with respect to the neighborhood size of the affinity matrix. *Left*: Yale. *Right*: CMU-PIE.

3.4.2 Comparison on Diffusion Process

Although the diffusion process has been used for the clustering problem [144, 155], it is not clear how it will perform on the subspace clustering, where the data is often of high dimension while the intrinsic dimension is much smaller. To show the effectiveness of the proposed DSSC, we compare it to:

- **Gaussian**. Gaussian affinity based on Euclidean distance. The bandwidth sigma is locally tuned as in [145], where it is determined by the neighbor distance.
- **TPG**. Affinity learned on Tensor Product Graph [155]. It starts with a Gaussian affinity matrix sparsed by the k-nearest neighbor.
- **SSC**. Sparse Subspace Clustering [36]. Affinity is learned with ℓ_1-norm minimization.

The clustering results are obtained by applying spectral clustering on the affinity matrices learned by these methods on synthetic data and real data.

Synthetic data. We construct 5 subspaces $\{S\}_{i=1}^{5} \subset \mathbb{R}^{100}$ whose bases $\{U_i\}_{i=1}^{5}$ are obtained by $U_i = T_i U, 1 \leq i \leq 5$, where U is a random matrix of dimension 100×5 and $T_i^{100\times100}$ is a random rotation. We sample 50 data points from each subspace by $X_i = U_i Q_i$, where the entries of $Q_i \in \mathbb{R}^{5\times50}$ are independent and identically distributed samples from a standard Gaussian. To further evaluate the robustness of these methods, certain percentage of data samples x are randomly chosen to be corrupted by Gaussian noise with zero mean and variance $0.2||x||$.

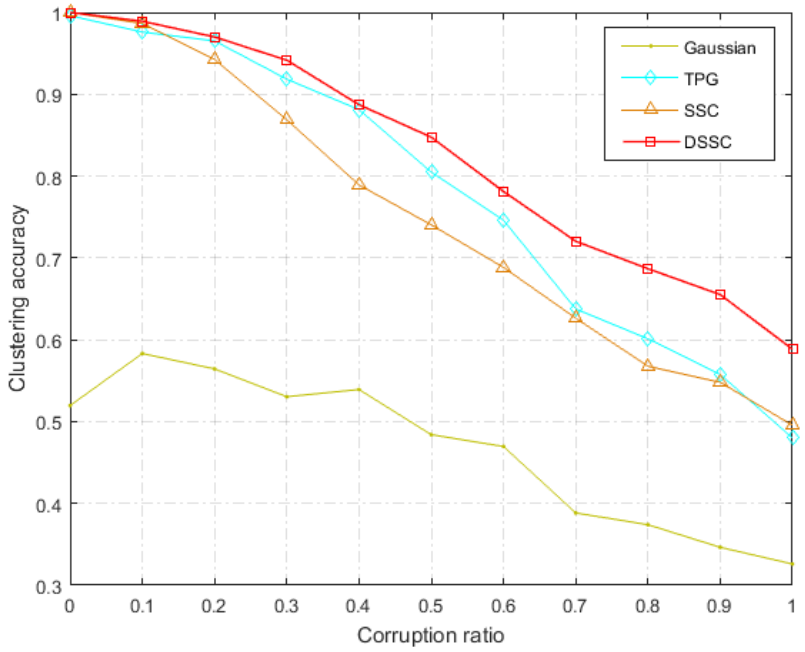

Figure 3-8: Clustering performance with respect to different ratio of corruption on the synthetic data.

Fig.3-8 demonstrates the clustering performances of these methods with respect to different ratios of corruption on the synthetic data. It can be seen that the proposed DSSC outperforms other methods in all cases. Gaussian affinity is obviously not working on such a subspace clustering problem, as it achieves only 50% accuracy even with no corruption. It becomes clear that based on Euclidean distance, Gaussian affinity cannot capture the geometry of the data manifold, which in this case, is the low-dimensional subspaces that lie in the high-dimensional ambient space. SSC performs quite well when the corruption

ratio is low, which implies that the sparse representation based on ℓ_1-norm is able to reveal the manifold structure of noise-free data, where, for each data point, the other data points invoked by the sparse reconstruction are indeed its "true" neighbors on the manifold. As the noise level increases, these subspaces are not mutually independent anymore, resulting in inadequate reconstruction coefficients (affinities), which explains the dramatic degradation of the performance.

TPG and the proposed DSSC are two methods that involve the diffusion process. Their main advantage is that instead of considering pairwise measurement, the diffusion process will re-evaluate the affinity using the context information. As TPG starts with Gaussian affinity, the performance gain of TPG over Gaussian clearly shows that the diffusion process is able to estimate the geodesic distance on the data manifold appropriately. However, when the noise percentage is high, the performance of TPG decreases significantly. One possible reason is that TPG starts with a KNN version of Gaussian affinity $\big(W_{ij} \neq 0$ if and only if $x_j \in knn(x_i)\big)$. When the noise level is low, the data manifold is closer to a convex subset of the feature space, where the Euclidean distances of local neighbors are still a reasonable estimation of the geodesic distances. As the noise level increases, the data manifold becomes arbitrary and often curved. In such cases, the Euclidean distances of neighbors can no longer provide good approximations of the geodesic distances.

DSSC consistently performs the best on this synthetic dataset. The success comes from two-fold: firstly, DSSC starts with the affinity matrix learned with ℓ_1-norm. Such sparse representation could conclusively reveal the exact geometry of the data manifold compared to the Gaussian affinity based on the Euclidean distance. Secondly, all the pairwise affinities are augmented by the context of each pair, where the context is defined by the previous sparse representation. As we can see, in the proposed DSSC method, ℓ_1-norm serves more like a neighborhood selection due to its sparse property. The local manifold is revealed by the sparse coefficients, and then the diffusion process combines such local manifolds to a complete global manifold by propagating the neighbor affinities following the geometry of the data structure. Using such a coarse-to-fine process, DSSC could appropriately estimate subspaces and achieve better clustering performance.

Face images. Given face images of different subjects obtained under various illumination conditions and a fixed pose, we consider the problem of clustering the images with respect to their subjects. Under the Lambertian assumption, it has been shown that the face

images of a single subject with varying illumination and fixed pose lie in a linear subspace of dimension nine [9]. Thus, face clustering can also be considered as a subspace clustering problem, where each subject lies in a 9D subspace.

The face image datasets used here are ORL and CMU-PIE [124]. ORL contains 400 grayscale face images of 40 subjects, each with 10 images of near frontal poses, various lighting conditions, and slight facial expressions.

To demonstrate the effectiveness of DSSC, we first evaluate the affinity matrices obtained by different methods, which could qualitatively reflect the performance of affinity learning. Fig.3-9 shows the visualization of these affinity matrices, obtained directly by applying MATLAB function *imagesc()*. The ground truth of the affinity matrix should have a block-diagonal structure, as all data samples are organized so that samples in the same cluster stay together.

It can be seen that the affinity matrices obtained by DSSC show much more clear block-diagonal structure and more sparse off-diagonal parts (between-class affinities), compared to the other methods. Specifically, in the case of ORL, due to the existence of noise produced by facial expressions and minor face poses, the main concern is the between-class affinity. Comparing the affinity matrices of SSC and DSSC, it clearly shows that DSSC could significantly reduce the between-class affinity. In the case of CMU-PIE, the within-class affinity is the concern. SSC produces a highly sparse affinity matrix by ℓ_1-norm, which causes low affinity values even within the same class. By propagating the local context affinities, DSSC successfully re-evaluates the affinity values and produces nearly perfect block-diagonal structure. The affinity matrices of Gaussian and TPG are clearly worse as they are all based on the Euclidean distance. These face images lie in high-dimensional feature space (1024D) while their intrinsic dimension is very low (9D), hence the Euclidean distance cannot represent the geodesic distance.

Table 3.3 shows the clustering performances obtained by applying spectral clustering on these affinity matrices. It can be seen that clustering performance coincides well with the quality of the affinity matrix. Gaussian and TPG, based on Gaussian affinity, achieve unsatisfactory results on both ORL and PIE datasets. While SSC produces reasonable results on these face images, clustering performances have been boosted with large margins by the proposed DSSC, 10%, and 20% on ORL and PIE, respectively. It should be noted that DSSC achieves nearly perfect clustering on PIE, which is also shown by the optimal

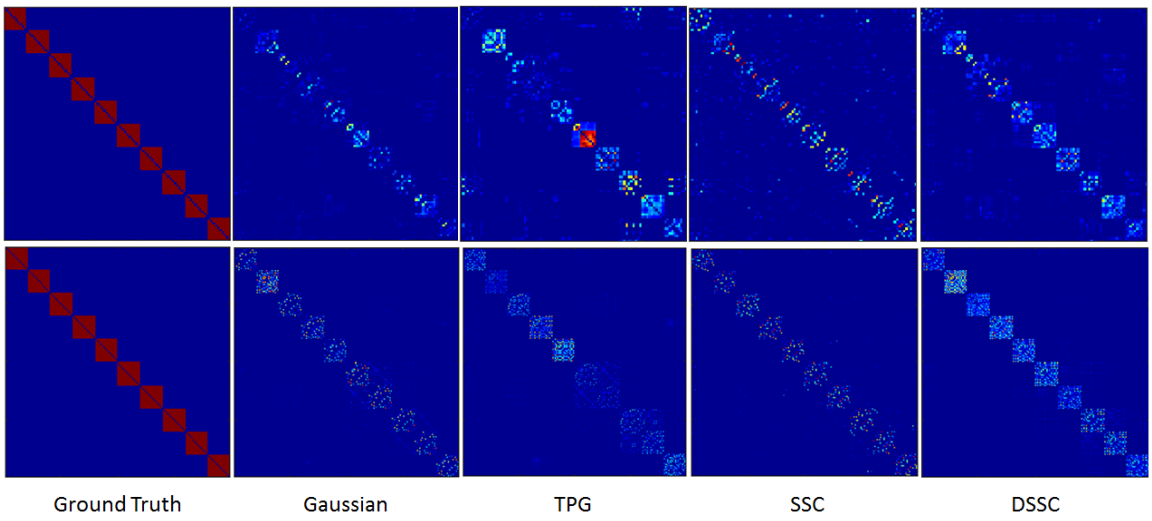

Figure 3-9: Visualization of affinity matrices obtained by different methods. *Top*: ORL. *Bottom*: PIE.

block-diagonal structure of the corresponding affinity matrix in Fig.3-9.

Table 3.3: Clustering accuracy on face images.

Methods	Gaussian	TPG	SSC	DSSC
ORL	0.60	0.68	0.72	**0.82**
PIE	0.30	0.48	0.75	**0.95**

3.4.3 Comparison on Subspace Clustering

In this section, we evaluate the proposed DSSC algorithm on two commonly used datasets in the field of subspace clustering, Hopkins 155 [134] and Extended Yale B [47], against other subspace clustering methods, i.e., SSC [36], SSSC [88], LRR [94], LSR [95], BDSSC [42], LRSC [40]. The experimental results of these methods are cited directly from [88], which provides a fair comparison as we use exactly the same experimental settings, as introduced in [36].

Hopkins 155. Motion segmentation refers to the problem of segmenting a video sequence of multiple rigidly moving objects into multiple spatiotemporal regions that correspond to different motions in the scene [36] (see Fig.3-10). The objects are often represented and tracked by a set of feature points through all frames in the video. Stacking the trajectories of all feature points, we obtain a data vector that represents the motion of the

corresponding object. Under the affine projection model, all feature trajectories associated with a single rigid motion lie in an affine subspace with its dimension no more than 3 [36]. Therefore, the motion segmentation problem reduces to the problem of clustering these trajectories in a union of subspaces.

Figure 3-10: Example frames from videos in the motion segmentation dataset Hopkins 155 [134].

We evaluate the proposed DSSC algorithm against existing subspace clustering methods on the Hopkins 155 motion segmentation dataset [134]. It consists of 155 video sequences, where 120 of them contain two motions, and 35 contain three motions. We evaluate the average performance of three different cases: 2 motions, 3 motions, and all. It can be clearly seen that the proposed DSSC achieves the best performances in all cases.

Table 3.4: Motion Segmentation Errors (%) on Hopkins 155 Dataset.

Methods	LRR	LRSC	LSR	BDSSC	SSC	SSSC	DSSC
2 motions	3.76	3.69	2.20	2.29	1.95	1.94	**1.68**
3 motions	9.92	7.69	7.13	4.95	4.94	4.92	**4.64**
All	5.15	4.59	3.31	2.89	2.63	2.61	**2.35**

Extended YaleB. In the last experiments, we evaluate the clustering performance of the proposed DSSC against other subspace clustering methods used for Hopkins 155 on the Extended Yale B dataset [47]. This dataset contains 2,414 frontal face images of 38 subjects, with 64 images per subject acquired under different illumination conditions. In terms of subspace segmentation, this dataset is more challenging than the Hopkins 155 dataset due to the heavy noise, high-dimensional space, and a large number of subspace in the data. In our experiments, we follow exactly the same settings as introduced in [36]: 1) each image is down-sampled to be 48×42 pixels, resulting in a 2,016-dimensional data point per image; 2) all the 38 subjects are divided into four groups, i.e., 1-10, 11-20, 21-30,

and 31-38. For the first three groups, we conduct experiments using all possible choices of $n \in 2, 3, 5, 8, 10$ subjects, and all possible choices of $n \in 2, 3, 5, 8$ for the last group.

Experimental results are presented in Table 3.5. It can be observed that the proposed DSSC once again achieves the best results in all cases. It is worth noting that on this more challenging data set, DSSC records more significant improvements compared to other subspace clustering methods.

Table 3.5: Clustering Errors (%) on the Extended Yale B Dataset.

Methods	LRR	LRSC	LSR	BDSSC	SSC	SSSC	DSSC
2 subjects	6.74	3.15	6.72	3.90	1.87	1.27	**0.61**
3 subjects	9.30	4.71	9.25	17.70	3.35	2.71	**1.25**
5 subjects	13.94	13.06	13.87	27.50	4.32	3.41	**2.80**
8 subjects	25.61	21.25	25.98	33.20	5.99	4.15	**4.04**
10 subjects	29.53	29.58	28.33	39.53	7.29	5.16	**4.84**

3.5 Summary

In this chapter, we proposed a novel affinity learning method that combined with spectral clustering, named as Diffusion based Sparse Subspace Clustering, which successfully incorporates the ideas from both diffusion and subspace clustering. Comparing to the original diffusion method, which uses Gaussian affinity and manually enforced KNN sparsity, DSSC applies the ℓ_1 norm that naturally produces sparse affinity based on the optimal reconstruction. For the subspace clustering problem, where the data are often drawn from low-dimensional subspaces which lie in very high-dimensional feature space, we have qualitatively and quantitatively shown that ℓ_1 norm affinity is a more sensible choice to estimate the geometry of the data manifold than the Euclidean distance based Gaussian affinity. Comparing to Sparse Subspace Clustering, we have demonstrated that ℓ_1 norm could serve as a neighborhood selection criteria which captures local manifold structure, while a simple yet effective diffusion process can be used to re-evaluate all the affinities by spreading the local structure following the global geometry of the data manifold. Extensive experiments on various types of data have shown the superiority of the proposed DSSC method over existing methods in both diffusion and subspace clustering fields.

The proposed DSSC, and any other spectral-based subspace clustering methods, obtain the final clustering result by applying k-means algorithm on the top k eigenvectors (k is

the number of class). One limitation of such algorithms is the expensive computational cost of the eigendecomposition of the graph Laplacian matrix, which makes it prohibitive for these algorithms to be applied for large-scale problems. In the next chapter, the classic Nyström method is formulated as a Markov Random Walk process, and based on that, a novel approximate approach for solving large-scale spectral problems is proposed using higher-order Markov transition matrices.

Chapter 4

Towards Large-Scale Spectral Problems with Diffusion Process

The clustering approach proposed in Chapter 3 performs very well for small and moderate datasets, but it is not readily practical for large-scale problems. The bottleneck is the eigendecomposition involved in the embedded spectral clustering, which has the $O(n^3)$ time complexity. In fact, not only spectral clustering, there are quite some popular non-linear dimensionality reduction methods based on eigendecomposition, which have the same problem. In this chapter, we formulate it as a general spectral problem and attempt to address it using the diffusion process, which is tightly related to random walks.

4.1 Introduction

High-dimensional data is ubiquitous due to the advancements in sensing and storage technology and the dramatic growth in applications such as Internet search, digital imaging, and video surveillance. The high dimensionality of data not only means high computational cost in terms of time and memory but also possibly restricts the performance of learning algorithms because of the noise effect and an insufficient number of data points with respect to the dimensionality of the ambient space, commonly referred to "curse of dimensionality" [13]. Fortunately, many high-dimensional datasets have much smaller intrinsic dimensionality than their corresponding ambient space, which has motivated the develop-

[0]

ment of a number of dimensionality reduction techniques. Linear dimensionality reduction techniques, such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), etc., can be powerful, but often miss the important non-linear structure of the data.

Manifold learning, also known as non-linear dimensionality reduction, has shown promising performance on revealing the nonlinearity of high-dimensional data. The assumption is that the data lie on a low-dimensional manifold despite the high dimensionality of its ambient space. Spectral methods that using an eigendecomposition for manifold learning have become a standard technique in this field, such as Local Linear Embedding (LLE) [117], Isomap [132], Laplacian Eigenmaps (LE) [11], and spectral clustering [104]. We consider a spectral problem of this type:

$$\min_{\mathbf{X}} \operatorname{tr}(\mathbf{X}\mathbf{W}\mathbf{X}^T) \quad \text{s.t.} \quad \mathbf{X}\mathbf{X}^T = \mathbf{I} \tag{4.1}$$

where $\operatorname{tr}(\cdot)$ represents the matrix trace, $W_{N\times N}$ is a symmetric matrix (affinity matrix or graph Laplacian) constructed on data points $\mathbf{Y} \in \mathbb{R}^{D\times N}$, and $\mathbf{X} \in \mathbb{R}^{d\times N}$ are the embedding for the N data points ($d < D$), which is given by the d trailing eigenvectors of $\mathbf{W}$. The problem can be solved efficiently with existing linear algebra routines when the number of points N is small, but it becomes computationally impractical when N is large due to the $O(N^3)$ complexity of the eigendecomposition.

We focus on solving the problem (4.1) approximately. A common method is sampling, i.e., solving the eigenproblem for a small subset of data points ("landmarks") and then extrapolating the solution to the entire dataset. The prototype is the Nyström method [44, 148], in which the embedding of out-of-sample points is obtained by the linear combination of landmarks' solution. This method is simple and appealing since, for a given number of landmarks, its complexity scales linearly with the data size N. Its fundamental disadvantage is that the reduced eigenproblem on the landmarks, to which the Nyström out-of-sample formula applies, uses only the landmarks-landmarks affinity values. Thus, it achieves a good approximation only if a sufficient number of landmarks are used [140].

Several other Nyström-like methods are proposed to use more information from the affinity matrix rather than just landmarks-landmarks affinities. In Column Sampling [46], the points-landmarks affinities are used. Local Linear Landmarks [142] and Variational Nyström [140] utilize the entire affinity matrix, where the Nyström formula is used as a

constraint to solve problem (4.1). The reduced eigenproblem does contain all the information of the affinity and thus better represents the manifold structure of the landmarks. Hence, it obtains a more sensible landmarks' embedding than the Nyström method given the same number of landmarks.

In this chapter, **our contribution can be summarized as**: 1) we propose a generic framework that includes all the Nyström-like methods. Based on the idea of the diffusion process (or Markov random walk), we show clearly that the differences of different methods are in the degree of Markov transition matrix order, which is used to construct the reduced affinity matrix and the corresponding eigenproblem afterward. This observation provides theoretical justification on why one method should outperform another. 2) Within the same framework, the *Diffusion based Variational Nyström* (DVN) method is proposed to approximately solve the spectral problem (4.1), in which a high-order Markov transition matrix is involved. The so-called diffusion process is actually a Markov Random walk on the affinity graph where the affinity is spread over as it goes on, leading to a modified, reduced affinity matrix that could benefit the approximate eigenproblem. 3) We applied the proposed DVN on dimensionality reduction and clustering tasks, and experiment results show that DVN can achieve the best tradeoff between accuracy and efficiency, compared to other existing approximate approaches.

The rest of the chapter is organized as follows: Section 4.2 briefly introduces some related work. The proposed framework and the Diffusion based Variational Nyström (DVN) are described in Section 4.3. Experiments are demonstrated in Section 4.4 and the chapter is concluded in Section 4.5.

4.2 Related Work

Nyström method was originally proposed as a quadrature method for finding numerical approximation to continuous eigenfunction problem [3]. After introduced to machine learning by [148], it becomes the most widely used approximate method for finding eigenvectors of large matrices, e.g. in kernel methods [28, 163], spectral clustering [14, 44], and manifold learning [128, 162].

Given a symmetric affinity matrix $\mathbf{W}_{N\times N}$, for the sake of simplicity, the matrix is organized so that the L chosen landmarks come first. We write $\mathbf{W}$ by blocks as:

$$\mathbf{W} = \begin{bmatrix} \mathbf{A} & \mathbf{B}_{21}^T \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B}_{21} \end{bmatrix}$$

where $\mathbf{A}_{L\times L}$ is the landmarks-landmarks affinity, and $\mathbf{B}_{21}$ is the affinity between non-landmarks and landmarks. In order to avoid expensive eigen-decomposition of the large matrix $\mathbf{W}_{N\times N}$, the Nyström method makes use of the eigen-decomposition of the small matrix $\mathbf{A}_{L\times L}$. Letting $\mathbf{A} = \mathbf{U_A}\mathbf{\Lambda_A}\mathbf{U_A^T}$, the approximate eigenvectors $\widetilde{\mathbf{U}}_{\mathbf{W}}^{Nys}$ and eigenvalues $\widetilde{\mathbf{\Lambda}}_{\mathbf{W}}^{Nys}$ of $\mathbf{W}$ are given by

$$\widetilde{\mathbf{U}}_{\mathbf{W}}^{Nys} = \begin{bmatrix} \mathbf{U_A} \\ \mathbf{B}_{21}\mathbf{U_A}\mathbf{\Lambda_A}^{-1} \end{bmatrix} = \mathbf{C}\mathbf{U_A}\mathbf{\Lambda_A}^{-1}, \quad \widetilde{\mathbf{\Lambda}}_{\mathbf{W}}^{Nys} = \mathbf{\Lambda_A}.$$

Column Sampling (CS) [46] method makes use of the left singular vectors of $\mathbf{C}_{N\times L}$ (affinity between landmarks and all points) to approximate the eigenvectors of $\mathbf{W}_{N\times N}$. Let $\mathbf{C} = \mathbf{U_C}\mathbf{\Sigma_C}\mathbf{V_C^T}$, $\widetilde{\mathbf{U}}_{\mathbf{W}}^{CS} = \mathbf{U_C} = \mathbf{C}\mathbf{V_C}\mathbf{\Sigma_C}^{-1}$. To further reduce the computation cost, the eigen-decomposition is applied on the matrix $\mathbf{C}^T\mathbf{C} = \mathbf{U}_{\mathbf{C}^T\mathbf{C}}\mathbf{\Lambda}_{\mathbf{C}^T\mathbf{C}}\mathbf{U}_{\mathbf{C}^T\mathbf{C}}^T$, as we know $\mathbf{V_C} = \mathbf{U}_{\mathbf{C}^T\mathbf{C}}$, $\mathbf{\Sigma_C} = \mathbf{\Lambda}_{\mathbf{C}^T\mathbf{C}}^{1/2}$. The approximation then becomes $\widetilde{\mathbf{U}}_{\mathbf{W}}^{CS} = \mathbf{C}\mathbf{V_C}\mathbf{\Sigma_C}^{-1} = \mathbf{C}\mathbf{U}_{\mathbf{C}^T\mathbf{C}}\mathbf{\Lambda}_{\mathbf{C}^T\mathbf{C}}^{-1/2}$.

Variational Nyström (VN) [140] is an approximate method which utilizes all information in $\mathbf{W}$ to better retain the manifold structure. Clearly, in the reduced eigenproblem of Nyström and CS, the affinity between non-landmarks and non-landmarks ($\mathbf{B}_{22}$) is totally abandoned. Thus, the performance of Nyström and CS heavily rely on having enough landmarks, resulting in poor performance with few landmarks, as shown in Fig.4-1 in the experiment. Motivated by the minimization problem (4.1), the reduced eigenproblem of VN is derived directly from the optimization problem itself by adding in the constraint that the embedding of all points $\mathbf{X}$ is the linear combination of the landmarks' solution $\widetilde{\mathbf{X}}$, using the point-landmark affinities $\mathbf{C}$ as coefficients, i.e., $\mathbf{X} = \widetilde{\mathbf{X}}\mathbf{C}^T$. The problem (4.1) then becomes

$$\min_{\widetilde{\mathbf{X}}} \ \mathrm{tr}\,(\widetilde{\mathbf{X}}\mathbf{C}^T\mathbf{W}\mathbf{C}\widetilde{\mathbf{X}}^T) \quad \text{s.t.} \quad \widetilde{\mathbf{X}}\mathbf{C}^T\mathbf{C}\widetilde{\mathbf{X}}^T = \mathbf{I} \tag{4.2}$$

whose exact solution $\widetilde{\mathbf{X}} = \widetilde{\mathbf{U}}$ is given by the $L \times L$ generalized eigen-problem

$$(\mathbf{C}^T\mathbf{W}\mathbf{C})\widetilde{\mathbf{U}} = (\mathbf{C}^T\mathbf{C})\widetilde{\mathbf{U}}\widetilde{\mathbf{\Lambda}}. \tag{4.3}$$

The final approximation is then obtained by $\widetilde{\mathbf{U}}_{\mathbf{W}}^{VN} = \widetilde{\mathbf{U}}\mathbf{C}^T$. VN does contain all affin-

ity information in the reduced eigenproblem by the added constraint, but no theoretical justification or interpretation of how the constraint works are presented.

Local Linear Landmarks (LLL) [142] is very similar to VN. Instead of using points-landmarks affinities $\mathbf{C}$, LLL reconstructs each data point using its nearest landmarks, resulting in a $N \times L$ reconstruction matrix $\mathbf{Q}$. After obtaining the embedding of the landmarks, the solution for all points is constructed by the linear combination of landmarks using $\mathbf{Q}$. The underlying assumption is that the locally linear structure in the high-dimensional space should be retained in the low-dimensional space, as originally formulated in LLE. The problem is that it is expensive to search for the nearest landmarks for all the data points, and the feature vectors have to be given.

Random Projection (RP) [55] method adopted different idea from the Nyström-like methods. Instead of sampling elements directly from $\mathbf{W}$, RP first uses a random matrix (normally Gaussian) $\mathbf{G}_{N \times L}$ to project $\mathbf{W}$ to a low-dimensional space $\mathbf{W_G} = \mathbf{W}^q\mathbf{G}$. With the QR decomposition, the orthogonal basis $\mathbf{Q}_{N \times L}$ of $\mathbf{W_G}$ can be obtained. The reduced eigenproblem is applied to the matrix $\mathbf{Q^T W Q}$. The core of RP is the Johnson-Lindenstrauss lemma, which states that random projection can preserve matrix structure with certain conditions [1]. The bottleneck of RP is the expensive computation of the random projection and QR decomposition.

4.3 Diffusion based Variational Nyström Method

The Nyström-like methods, e.g., Nyström, CS, and VN, mainly consist of two steps: (1) constructing a small affinity matrix $\mathbf{M}_{L \times L}$ for the landmarks; (2) solving the reduced eigenproblem and extrapolating the solution to all data points by an out-of-sample matrix $\mathbf{Z}_{N \times L}$. Despite the various ways applied by different methods, we show that it is possible to generalize them into one framework based on the idea of the diffusion process.

The Nyström method uses the affinity values of landmarks in $\mathbf{W}$ to form $\mathbf{M}$, i.e., $\mathbf{M} = \mathbf{A}$. Assuming $\mathbf{M}$ is a row-stochastic matrix, we first construct a undirected graph $G = (V, E)$, consisting of L nodes $v_i \in V$ that represent data points, and edges $e_{ij} \in E$ that link pairs of nodes. The edge weights are set to the affinity values m_{ij}. $\mathbf{M}$ can be seen as a one-step Markov transition matrix, in which the entry m_{ij} defines the probability of random walk from node i to node j with one step. Let $\{v_k\}_{k=1}^{N}$ denote all the data points and $\{u_k\}_{k=1}^{L}$

denote the landmarks, and $p^{(q)}(u_j|u_i)$ is the transition probability of walking from landmarks i to landmarks j with t step(s). In Nyström, we have

$$m_{ij} = a_{ij} = p^{(1)}(u_j|u_i). \tag{4.4}$$

In CS, the small affinity matrix is constructed as $\mathbf{M} = \mathbf{C}^T\mathbf{C}$, which we show as a two-step transition matrix. Using the same interpretation, $\mathbf{C}_{N\times L}$ can be seen as the one-step transition matrix from all points to landmarks. By the chain rule of Markov random walks,

$$\begin{aligned} m_{ij} = \sum_{k=1}^{N} c_{ik} \cdot c_{kj} &= \sum_{k=1}^{N} p^{(1)}(v_k|u_i) \cdot p^{(1)}(u_j|v_k) \\ &= p^{(2)}(u_j|u_i). \end{aligned} \tag{4.5}$$

$\mathbf{C}^T\mathbf{C}$ serves as the two-step transition matrix between landmarks, where the internal paths are not restricted to the landmarks, but to all data points.

Similarly, VN constructs $\mathbf{M}$ using all information in $\mathbf{W}$: $\mathbf{M} = \mathbf{C}^T\mathbf{W}\mathbf{C}$. Clearly, $\mathbf{W}$ is the one-step transition matrix that defines the probabilities of walking between all data points. Thus, $\mathbf{M}$ can be seen as a three-step transition matrix, where the first step is walking from the landmarks to all data points, the second step is walking from all data points to itself, and the third step is walking from all data points back to the landmarks, as shown below:

$$\begin{aligned} m_{ij} &= \sum_{k=1}^{N}\Big(\sum_{k'=1}^{N} c_{ik'} \cdot w_{k'k}\Big) \cdot c_{kj} \\ &= \sum_{k=1}^{N}\Big(\sum_{k'=1}^{N} p^{(1)}(v_{k'}|u_i) \cdot p^{(1)}(v_k|v_{k'})\Big) \cdot p^{(1)}(u_j|v_k) \\ &= \sum_{k=1}^{N} p^{(2)}(v_k|u_i) \cdot p^{(1)}(u_j|v_k) \\ &= p^{(3)}(u_j|u_i). \end{aligned} \tag{4.6}$$

While the way of constructing $\mathbf{M}$ varies among Nyström, CS, and VN, the out-of-sample extrapolations remain the same. They all use the submatrix $\mathbf{C}$ from $\mathbf{W}$. It is simple and

straightforward to use $\mathbf{C}$ as it initially represents the affinities between landmarks and all data points.

Based on the above discussion, it is clear that the essence of Nyström-like methods lies in the choice of small affinity matrix $\mathbf{M}$. Hence, we propose to construct $\mathbf{M}$ as

$$\mathbf{M} = \mathbf{C}^T \mathbf{W}^{2q+1} \mathbf{C}. \tag{4.7}$$

$\mathbf{M}$ is a higher-order Markov transition matrix, where the hyper-parameter q determines the degree of orders. $\mathbf{W}^{2q+1}$ is the core of the so-called diffusion process. As q increases, it allows us to use as much information as we want from the original affinity matrix. Intuitively, the more information we use from the original affinity matrix $\mathbf{W}$, the better $\mathbf{M}$ captures the manifold structure.

After the standard eigendecomposition is applied on $\mathbf{M}$, the final step is to extrapolate the solution of landmarks $\mathbf{U_M}$ to all data points. Unlike the other methods, which utilize the landmarks-points affinities $\mathbf{C}$ directly, we propose a new out-of-sample formula:

$$\widetilde{\mathbf{U}}_{\mathbf{W}} = \mathbf{U_M} \mathbf{C}^T \mathbf{W}^q. \tag{4.8}$$

The intuition is that if the diffusion process can improve the affinity matrix of landmarks, it can also do the trick for the extrapolation. The differences of the reduced affinity matrix and extrapolation formula between DVN and the other Nyström-like methods are concluded in Table 4.1.

Table 4.1: Choices of different algorithms for the reduced affinity matrix and out-of-sample formula

Algorithms	affinity $\mathbf{M}_{L\times L}$	extrapolation $\mathbf{Z}_{N\times L}$
Nyström	$\mathbf{A}$	$\mathbf{C}$
CS	$\mathbf{C}^T\mathbf{C}$	$\mathbf{C}$
LLL	$\mathbf{Q}^T\mathbf{W}\mathbf{Q}$	$\mathbf{Q}$
VN	$\mathbf{C}^T\mathbf{W}\mathbf{C}$	$\mathbf{C}$
DVN	$\mathbf{C}^T\mathbf{W}^{2q+1}\mathbf{C}$	$\mathbf{W}^q\mathbf{C}$

Optimization. We now state the proposed DVN from the optimization point of view. Recall the original optimization problem (4.1), we add $\mathbf{X} = \widetilde{\mathbf{X}}\mathbf{C}^T\mathbf{W}^q$ into the out-of-sample formula as a constraint. It becomes

Algorithm 3 Diffusion based Variational Nyström (DVN).

Input: The full affinity matrix $\mathbf{W}_{N\times N}$, the number of landmarks L, the dimensionality of the embedding d, the degree of the diffusion q.

1: Randomly choose L landmarks out of N.

2: Re-organize the affinity matrix so that the landmark-landmark affinities $\mathbf{A}$ come first and write $\mathbf{W}$ by blocks

$$\mathbf{W}_{N\times N} = \begin{bmatrix} \mathbf{A} & \mathbf{B}_{21}^T \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix}, \quad \mathbf{C}_{N\times L} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B}_{21} \end{bmatrix}$$

.

3: Construct the $L \times L$ reduced affinity matrix by $\mathbf{C}^T\mathbf{W}^{2q+1}\mathbf{C}$.

4: Solve the $L \times L$ generalized eigen-problem $(\mathbf{C}^T\mathbf{W}^{2q+1}\mathbf{C})\widetilde{\mathbf{U}} = (\mathbf{C}^T\mathbf{W}^{2q}\mathbf{C})\widetilde{\mathbf{U}}\widetilde{\mathbf{\Lambda}}$

5: Obtain the approximate embedding for all the points by applying the out-of-sample formula $\widetilde{\mathbf{U}}_{\mathbf{W}} = \widetilde{\mathbf{U}}\mathbf{C}^T\mathbf{W}^q$.

Output: An approximate full embedding $\widetilde{\mathbf{U}}_{\mathbf{W}}$.

$$\min_{\mathbf{X}} \ \mathrm{tr}\,(\mathbf{X}\mathbf{W}\mathbf{X}^T) \quad \text{s.t.} \quad \mathbf{X}\mathbf{X}^T = \mathbf{I},\ \mathbf{X} = \widetilde{\mathbf{X}}\mathbf{C}^T\mathbf{W}^q \tag{4.9}$$

where $\mathbf{W}$ is the full affinity matrix partitioned in (4.1). We can obtain a reduced $L \times L$ spectral problem w.r.t $\widetilde{\mathbf{X}}$:

$$\min_{\widetilde{\mathbf{X}}} \ \mathrm{tr}\,(\widetilde{\mathbf{X}}\mathbf{C}^T\mathbf{W}^{2q+1}\mathbf{C}\widetilde{\mathbf{X}}^T) \quad \text{s.t.} \quad \widetilde{\mathbf{X}}\mathbf{C}^T\mathbf{W}^{2q}\mathbf{C}\widetilde{\mathbf{X}}^T = \mathbf{I} \tag{4.10}$$

where we can see the reduced affinity matrix becomes $\mathbf{C}^T\mathbf{W}^{2q+1}\mathbf{C}$ which is exactly Eq.4.7. The exact solution $\widetilde{\mathbf{X}} = \mathbf{U}_{\mathbf{M}}$ for the above reduced spectral problem (4.10) is obtained by calculating the d trailing eigenvectors of the following generalized eigen-problem

$$(\mathbf{C}^T\mathbf{W}^{2q+1}\mathbf{C})\mathbf{U}_{\mathbf{M}} = (\mathbf{C}^T\mathbf{W}^{2q}\mathbf{C})\mathbf{U}_{\mathbf{M}}\widetilde{\mathbf{\Lambda}}. \tag{4.11}$$

Thus, by substituting it back to the out-of-sample formula we have the solution for all the points

$$\mathbf{X} = \widetilde{\mathbf{X}}\mathbf{C}^T\mathbf{W}^q = \mathbf{U}_{\mathbf{M}}\mathbf{C}^T\mathbf{W}^q. \tag{4.12}$$

which gives us exactly Eq.(4.8). The algorithm is summarized in Algorithm 1.

Complexity analysis. The proposed algorithm mainly consists of three steps: 1) Construct the reduced affinity matrix; 2) Solve the reduced eigenproblem; 3) Apply the out-of-sample formula. Suppose we use L landmarks out of N, the first step is dominated

by the matrix multiplication. Depending on the sparsity of $\mathbf{W}$, its complexity is between $\mathcal{O}(NL)$ and $\mathcal{O}(N^2L)$. Solving the eigenproblem for landmarks is $\mathcal{O}(L^3)$. The complexity of applying the out-of-sample extension is also sensitive to the sparsity, which is between $\mathcal{O}(NL)$ and $\mathcal{O}(N^2L)$. It is impossible to give an exact complexity measure without specifying the sparsity of the affinity matrix. Considering the practical case, however, where $\mathbf{W}$ is sufficiently sparse, and $L \ll N$, the computational complexity of the proposed algorithm is roughly $\mathcal{O}(NL)$, which is significantly less than that of the original $\mathcal{O}(N^3)$.

4.4 Experiments

To evaluate the proposed DVN, we set up the spectral problem on dimension reduction and spectral clustering. For all the experiments, the affinity matrix is constructed by entropic affinities [63, 141], in which the Gaussian bandwidth for each data point is adapted to its neighbors. Each row of the affinity matrix is also sparsified by zeroing all but the 200 largest values. The hyper-parameter q of DVN is set to 1 in all the experiments. All the eigenproblems in our experiments are solved with the *svd* routine of MATLAB 2014b. Table 4.2 lists details of the datasets used in the experiments.

Table 4.2: Datasets used in our experiments.

Datasets	**# of instances**	**# of dimensions**	**# of class**
Swiss roll	10000	3	-
MNIST20000	20000	784	-
Pendigits	10992	16	10
Caltech101	9197	1024	101
20Newsgroups	18846	5000	20
MNIST	70000	784	10
Covype	581012	54	7

4.4.1 Large-Scale Dimensionality Reduction

For the experiment of dimension reduction, we use Laplacian Eigenmaps (LE) to set up the spectral problem (4.1). As the solution quality is not indicated by the objective function itself, we compare the approximate solutions to the exact embedding of LE after Procrustes alignment, as problem (4.1) is invariant to rotation and translation. The error is reported as the relative least square error between the exact solution $\mathbf{X}$ and the approximation $\widetilde{\mathbf{X}}$:

$$error = \frac{||\mathbf{X} - \mathcal{P}(\mathbf{X}, \widetilde{\mathbf{X}})||_F}{||\mathbf{X}||_F}, \tag{4.13}$$

where $\mathcal{P}$ is the procrustes alignment, and $||\cdot||_F$ is the Frobenius norm.

As the exact LE is needed to run for comparison, we use two relatively small sets: one synthetic dataset Swiss roll where there are 10,000 points and MNIST20000, in which 20,000 digits are randomly selected from MNIST. The dimensionality is reduced to $d = 2$ and $d = 10$ for Swiss roll and MNIST20000 respectively. To completely show the approximation quality of different methods, we try 20 log-spaced values between 10 and N for the number of landmarks. For different numbers of landmarks L, the experiment is run for five times, each with different random landmarks. The proposed DVN is compared to Nyström (Nystrom), Column Sampling (CS), Variational Nystrom (VN), Locally Linear Landmarks (LLL) [142], and Random Projection (RP) methods. Fig.4-1 shows the approximation quality of the different methods, in which the relations of a number of landmarks, relative error, and runtime are presented.

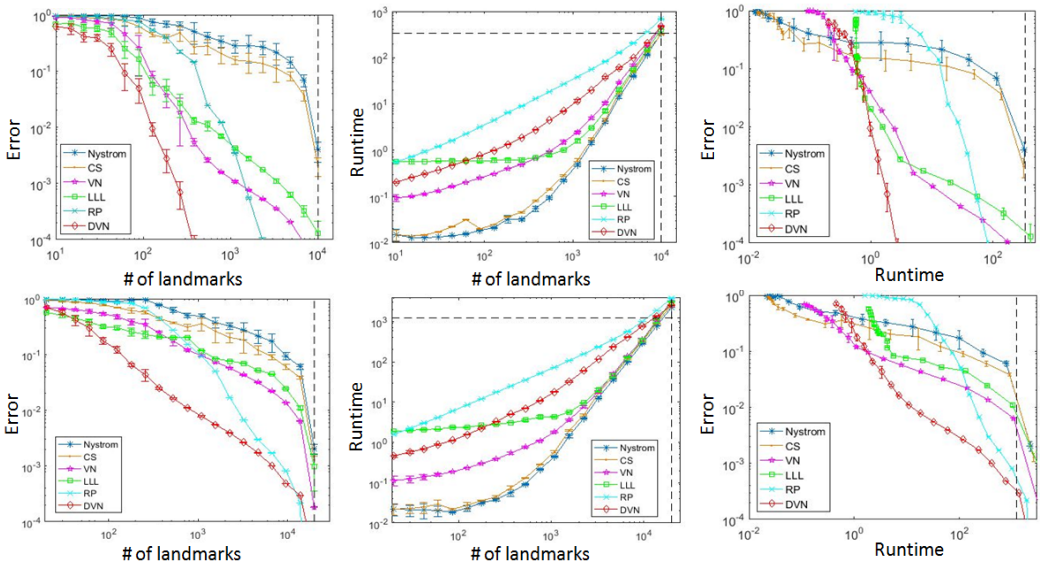

Figure 4-1: *Top*: Performance analysis for approximating the trailing two eigenvectors of the normalized graph Laplacian on Swiss roll. *Bottom*: Performance analysis for approximating the trailing ten eigenvectors of the normalized graph Laplacian on MNIST20000. *From left to right*: relative error for a particular number of landmarks, run time for a particular number of landmarks, and relative error decrease per second of run time. The dashed lines mean the number of landmarks reaches to N or the run time is equal to the run time of the exact LE.

Fig.4-1 (left) shows the relative error decrease per landmark. It can be seen that the proposed DVN continuously outperforms all the other competitors. Nystrom and CS are not working well. The error did not drop much until the number of landmarks is close to N. LLL and VN show a faster error decrease but not as much as DVN. RP has a good drop slope but requires much more landmarks compared to DVN. Considering a decent error rate of 10^{-2}, DVN needs only around 100 landmarks while nearly 1,000 needed by RP on Swiss roll. On MNIST20000, DVN has shown an even larger margin in the early stage. Given 1,000 random landmarks, which is 5% of all points, DVN has reached the error 10^{-2} while all the others are between 1 and 10^{-1}.

Fig.4-1 (middle) presents the run time increment per landmark. Nystrom and CS take the least time for a given number of landmarks, as they only use part of the affinity matrix. LLL is quite fast at the later stage but needs some time to compute the local linear reconstruction matrix. RP clearly needs more time to run due to the expensive projection and QR decomposition. Since VN and DVN do use all the affinity matrix, they are slow as the run time is dominated by the matrix product when the number of landmarks L is small. However, when L is large, the run time starts to be dominated by the reduced eigenproblem, which explains why the differences between run time become smaller and smaller.

Fig.4-1 (right) shows the error decrease per second of run time, which is combined from the previous two plots. Clearly, DVN results in a better tradeoff between speed and accuracy after just a very short of running time. As explained previously, DVN uses the entire affinity matrix so that the matrix product dominates the runtime, given a small number of landmarks. With a reasonable L (typically 2% of all points), DVN could easily outperform all the other methods. The methods can be ranked as DVN, VN, LLL, RP, CS, Nystrom, from best to worst.

4.4.2 Large-Scale Spectral Clustering

To evaluate the ability of the algorithms in handling large-scale spectral problem, we set up another experiment for the spectral clustering. A few large real-world datasets are used: Pendigits, Caltech101, 20Newsgroups, MNIST, and Covtype. The details of these data are listed in Table.4.2. Specifically, for Caltech101 [41], a pre-trained convolution neural network called VGG-net [125] is used to extract CNN features. For 20Newsgroups [82], the first 5,000 tf-idf features are used as the representation.

The proposed DVN is compared to several other methods for large-scale spectral clustering:

- **K-means** classic clustering algorithm used as a baseline.
- **KASP** k-means based approximate spectral clustering method, proposed in [153].
- **Nyström** classic method for finding numerical solution for eigenfunction problem, which was used by [44] for large spectral problem.
- **LSC** landmarks based sparse coding method for large-scale spectral clustering proposed in [25] , which is similar to Column Sampling (CS).
- **SSSC** scalable sparse subspace clustering [110], in which the large-scale problem is handled with an out-of-sample extension.

For a fair comparison, the number of landmarks used for all methods is fixed to $L = 1000$. As all methods involve k-means clustering at the end, the settings are the same: $maxIter = 1000$, $numRep = 10$. The average performance from 10 runs is reported since the landmarks are randomly selected.

Table 4.3: Clustering accuracy (%) of different methods.

Datasets	Pendigits	Caltech101	20Newsgroups	MNIST	Covtype
k-means	77.1	60.2	45.3	54.2	29.2
KASP	78.8	63.2	47.3	55.9	24.5
Nystrom	73.6	65.1	52.4	47.0	24.5
LSC	80.5	61.3	50.8	61.5	25.3
SSSC	81.7	57.4	46.7	57.2	31.1
DVN	**84.1**	**68.7**	**63.8**	**72.5**	**47.6**

Table 4.4: Clustering time (s) of different methods.

Datasets	Pendigits	Caltech101	20Newsgroups	MNIST	Covtype
k-means	6.8	57.3	557.8	151.2	142.5
KASP	12.2	38.2	322.6	145.2	151.4
Nystrom	6.2	15.3	23.6	28.4	133.8
LSC	**2.8**	**7.6**	**12.2**	25.6	62.2
SSSC	5.8	25.7	254.2	164.8	178.6
DVN	4.5	9.8	17.5	**22.4**	**35.3**

Table.4.3 presents the clustering accuracy of all methods. It is shown that the proposed DVN outperforms the other methods on all datasets with a large margin. Especially on

large datasets 20Newsgroups, MNIST and Covtype, 10% performance gain over the second best method can be observed. As it is impractical to apply the exact spectral clustering on these large datasets, the performance of k-means is reported as a baseline method for exact clustering (compare to the other approximate methods). It can be seen that DVN has clearly shown its superiority over k-means (almost 20% accuracy gain).

Table.4.4 shows the execution time of different methods. K-means are the slowest as it processes all the data points. Among the approximation methods, KASP is slow to run as it is based on k-means. SSSC is slow when the data dimension is large since it involves a ℓ^1-minimization problem. LSC is faster than DVN when data size is small. For large datasets MNIST and Covtype, the proposed DVN shows a clear advantage.

4.5 Summary

Nyström formula serves as a prototype of methods that approximately solve the large-scale spectral problem by sampling. The main drawback is that only part of the affinity matrix (landmark-landmark affinities) is utilized in the eigenproblem. While several other methods are proposed based on Nyström that use all the information of the affinity matrix, they provide no theoretical justification on why one method should be better than another. In this chapter, we proposed a generic framework based on the idea of the diffusion process, which covers all Nyström-like methods. Within this framework, we show clearly that the differences among these methods are in the degree of the Markov transition matrix, which is used to construct the reduced affinity matrix. As such a degree increases, more information from the original affinity matrix will be used so that the underlying manifold data structure would be better retained, which exactly explains the performance ranking of these methods.

Moreover, as the degree of Markov transition matrix used by existing methods is at most 3, we proposed a novel method, called Diffusion-based Variational Nyström (DVN), within the framework using a higher-order transition matrix, taking the power of affinity matrix by a diffusion process. Such a process could be easily understood if the affinity matrix is transferred to an affinity graph where the node represents the data point, and the edge is weighted by the affinity value. The diffusion is then interpreted as a Markov random walk on the graph. As the walking goes on, the affinity is spread to the entire graph, resulting in the obtained reduced affinity matrix contains as much information as the original. In this

case, the corresponding eigenproblem could be approximated as much as possible.

Experiments on dimension reduction have shown the superiority of the proposed framework, and the experiments on clustering demonstrate the capacity of the proposed DVN in handling large-scale spectral problems.

The next chapter starts to discuss affinity learning in semi-supervised learning. The benefits of learning affinity in such settings are emphasized, and a semi-supervised diffusion process is proposed to take advantage of label information.

Chapter 5

Semi-Supervised Learning with Self-Reinforced Diffusion Process

As discussed in Chapter 3 and 4, affinity learning is beneficial for many unsupervised learning approaches, *e.g.*, subspace clustering. It turns out that in a semi-supervised setting, where both labeled and unlabeled data are available, it is also important to learn the affinity between data points. As the common assumption is that labeled data is limited while unlabeled data is abundant, there is not enough labeled data to train a reliable supervised model directly. Hence, many semi-supervised models rely on establishing a connection from labeled points to unlabeled ones so that labels can be propagated from one to another. Such connections are usually revealed by the pairwise affinity. In this chapter, we discuss how to learn affinity using the diffusion process for semi-supervised learning.

5.1 Introduction

Semi-supervised learning (SSL) refers to the problem of learning from both labeled and unlabeled data. SSL is of particular interest in many real-world applications since labeled data is often scarce, whereas unlabeled data is abundant. The heuristic approach of self-training is probably the earliest approach to SSL, which dates back to the 1960s when examples of applications appeared [121]. Self-training starts with a "weaker" classifier trained only with the limited labeled data. The classifier is then applied to the unlabeled data to generate

more training samples so that the classifier can be re-trained with additional training samples and hopefully improved. Explicit constraints on selecting additional training samples can be added to aid the process, such as [74]. However, as a wrapper method, self-training is hard to analyze in general [174]. Co-training [19, 161] uses the same iterative training strategy but with two or more classifiers. These classifiers are first trained with a conditionally independent subset of features and refined with the output of another classifier. The success of co-training methods relies heavily on the existence of independent and complementary feature subsets [146]. Self-training and co-training are seen as semi-supervised methods as they incorporate both labeled and unlabeled data, but the training procedure of their classifier on its own is supervised. Transductive support vector machine (TSVM) [73] is another commonly used SSL model based on margin maximization. It leverages unlabeled data to find a linear boundary in low-density regions. TSVM can be viewed as the conventional SVM with an additional regularization term on unlabeled data [174]. However, it is NP-hard to solve the exact TSVM problem, which motivates the research focused on finding approximate solutions [17, 152]. Another type of SSL method, known as the graph-based approach, has also attracted much attention due to its elegant formulation and promising performance.

Graph-based semi-supervised learning (GSSL) represents both labeled and unlabeled data as vertices in an undirected graph $G = (V, E)$, where edges between vertices are usually weighted by the corresponding pairwise affinities/similarities. The critical insight of GSSL is the manifold/cluster assumption indicating that points on the same manifold are likely to have the same label [166]. With the large portion of unlabeled points revealing the underlying manifold structure represented by the graph, the small portion of labeled points are used to perform label propagation for transductive inference on the unlabeled points.

Over the last decade, many works on GSSL, such as *Label Propagation* [172], *Gaussian Field and Harmonic Function* (GFHF) [173] and *Local and Global Consistency* (LGC) [166], focused on how to effectively propagate labels on a pre-defined graph. The prototype approach is to formulate it as a regularized function estimation problem, targeting a trade-off between the accuracy of the classification function on the labeled points and the regularization, which favors a function that is sufficiently smooth with respect to the intrinsic manifold structure revealed by both labeled and unlabeled points. These label propagation methods can effectively spread labels, given that the underlying manifold is appropriately

represented by the affinity graph. What if the graph itself is problematic? That is indeed the case during the past decade when a typical graph is constructed using the Gaussian kernel in the Euclidean space, as it is well-known that the Euclidean distance cannot accurately approximate the geodesic distance on the manifold, especially in high-dimensional space due to the "curse of dimensionality".

Recently, the research focus of GSSL has been shifted to constructing an adequate graph [39, 87, 164, 176] that can better reveal the data manifold in order to facilitate the label propagation. As noted by the survey paper [174], it is more important to construct a better graph than selecting one of the label propagation approaches. State-of-the-art graph construction methods are usually based on the self-expressive model [36], which reconstructs each data point by a linear combination of all other data points. The reconstruction coefficients, regularized by sparsity or low-rankness, are then used to construct a graph. Using this type of graph as the initialization, we proposed the DSSC algorithm in Chapter 3, which post-processed the graph with a diffusion process, achieved very impressive results in unsupervised learning, *e.g.*, clustering.

The proposed DSSC, however, does not fit into semi-supervised settings, as it is not clear how to incorporate label information. Purely based on the neighbor propagation of the diffusion process may result in a fully local approach that is not aware of the global structure of data. On the other hand, the diffusion process defined in DSSC does not consider data density distribution. However, intuitively, larger affinities should be assigned to points lie in the same high-density regions or clusters, or in other words, data points, with higher local densities, should contribute more to the propagation of the diffusion process. It means that the data density distribution ought to be considered in addition to the manifold structure when evaluating pairwise affinities, as mentioned by [127].

To address these problems, we now propose a variant of the diffusion process, which efficiently takes advantage of both the partial label information and data density distribution. Specifically, the Label information is utilized by imposing neighbor constraints on points with the same label, while the data density distribution is explicitly modeled by introducing the self-affinity. The self-affinity of each point is defined to be proportional to its node degree on the graph, which is inherently a measurement of the local data density.

The existence of self-affinities allows the diffusion process to spread affinity values through itself, which can be thought of as a kind of *self-reinforcement*. Therefore, we name the pro-

posed diffusion process as *Self-Reinforced Diffusion* (SRD). Motivated by the recent advances in the diffusion process, particularly by [6], we formulate the proposed SRD into two equivalent forms: a regularization framework and an iterative formula. While the regularization framework gives theoretical insight on why SRD works, the iterative formula converges efficiently to the same closed-form solution as the regularization framework. We will elaborate them in Section 5.3.1 and Section 5.3.2 respectively.

The contribution of this chapter can be summarized as follows:

- We propose a semi-supervised variant of the diffusion process that can take advantage of valuable label information in graph construction.
- We propose the idea of self-similarity so that data density distribution can be efficiently incorporated in the diffusion process.
- We conduct experiments to demonstrate the significance of using label information for graph construction and validate the appropriate choice of self-similarity.

5.2 Related Work

Graph construction is of great importance to the success of all GSSL methods, as the effectiveness of the following label propagation stage depends heavily on having an accurate graph that adequately reveals the underlying manifold. A graph can be represented by a weight matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$, where n is the number of data points. A non-zero entry W_{ij} states that node i and node j are connected to each other, and the edge is weighted by the value of W_{ij}. The Gaussian kernel weight matrix $W_{ij} = \exp\left(-\|x_i - x_j\|^2 / 2\sigma^2\right)$ is widely used in this field. However, the bandwidth parameter σ has to be carefully tuned before a reasonable result can be obtained. Moreover, a global σ is not appropriate if data is not evenly distributed. Local adaptive Gaussian kernels [141, 160] are therefore proposed and used as default by many GSSL methods. Another challenge with the Gaussian weight matrix is that it usually performs poorly with high-dimensional data due to "the curse of dimensionality". This problem can be alleviated by graph sparsification, such as the k-nearest neighbor (KNN) graph, as it can filter a large portion of noisy edges which are harmful for the subsequent label propagation.

Recent works [60, 175] on GSSL have focused on constructing a better graph using the

self-expressive model, *i.e.*, expressing each data point as a linear combination of all other points, while regularizing the coefficients by sparsity or low-rankness. The problem can be formulated as:

$$\min_{C} ||C|| \quad \text{s.t.} \quad X = XC, \quad \text{diag}(C) = \mathbf{0}, \tag{5.1}$$

where $X = [x_1, ..., x_n]$ is the data matrix, $C = [c_1, ..., c_n]$ is the coefficient matrix, and $||\cdot||$ is a properly chosen norm, *e.g.*, $||\cdot||_1$ for sparse representation [36] and $||\cdot||_*$ for low-rank representation [94]. The weight matrix is then constructed by $|C| + |C|^\top$. Note that a clear advantage of the self-expressive model based weight matrix is that it is sparse in nature, and no KNN sparsity is needed. The number of non-zero elements is automatically chosen by the optimization algorithm and can be varied among data points rather than a fixed hyper-parameter k. This reduces a lot of effort of practitioners on tuning the hyper-parameters of the Gaussian kernel. However, these approaches appeared in subspace clustering do not fit into the SSL setting as the given labels are totally neglected. A natural extension [176] is to add a hard constraint to force the affinities to be zero for points with different labels (cannot-link), and one for points with the same labels (must-link). While such label information guided graph construction methods [39,176] showed a reasonable performance gain, they are still suboptimal as only the given labels are exploited. It was shown in [87] that the predicted unknown labels can provide additional valuable supervision for graph construction, with a proper feedback mechanism.

Label propagation. After the graph is constructed, the next step is to propagate labels on it. Many classic GSSL methods [166, 172, 173] formulated label propagation as a regularized function estimation problem consisting of two terms namely fitness and smoothness. The *fitness term* measures the fitness between the predicted labels and given labels, while the *smoothness term* measures the smoothness of the prediction with respect to the graph structure. The target prediction or classification function $\mathbf{F} \in \mathbb{R}^{n \times c}$, where its entry F_{ij} represents the probability of x_i belonging to the j-th class, can be obtained by finding a trade-off between these two terms:

$$\arg\min_{\mathbf{F}} \left(\mathcal{J}_{fit}(\mathbf{F}) + \mu \mathcal{J}_{smooth}(\mathbf{F}) \right). \tag{5.2}$$

A common choice of the fitness term is a quadratic loss between the predicted labels and the groundtruth labels, while imposing graph Laplacian as a smooth operator for the smoothness term, such as GFHF [173]:

$$\min_{\mathbf{F}} \infty \sum_{i=1}^{\ell} \|F_{i\cdot} - Y_{i\cdot}\|^2 + \sum_{i,j=1}^{n} W_{ij} \|F_{i\cdot} - F_{j\cdot}\|^2, \tag{5.3}$$

where ℓ is the number of labeled points, and $\mathbf{Y} \in \mathbb{R}^{n\times c}$ is the one-hot encoded label matrix, *i.e.*, $Y_{ij} = 1$ if $y_i = j$ and 0 otherwise. Note that the infinity weight ∞ clamps the predictions on the labeled points to be the given labels. LGC [166] relaxed the infinity weight to a trade-off parameter $\mu > 0$, and applied the fitness term on all data points instead of only labeled ones:

$$\min_{\mathbf{F}} \mu \sum_{i=1}^{n} \|F_{i\cdot} - Y_{i\cdot}\|^2 + \sum_{i,j=1}^{n} W_{ij} \left\| \frac{F_{i\cdot}}{\sqrt{D_{ii}}} - \frac{F_{j\cdot}}{\sqrt{D_{jj}}} \right\|^2, \tag{5.4}$$

where the diagonal matrix $\mathbf{D}$, with its element $D_{ii} = \sum_{j=1}^{n} W_{ij}$, serves as a normalization factor. GGMC [146] followed the same cost function as LGC but reformulated it into a bivariate optimization problem with respect to not only the classification function $\mathbf{F}$ but also the binary label matrix $\mathbf{Y}$. By updating these two variables alternately, $\mathbf{F}$ is learned iteratively by feeding the predicted labels back to reinitialize the given labels $\mathbf{Y}$, resulting in a robust classification function which is less dependent on the given labels. A novel label propagation method based on KL-divergence was proposed in [38]. Curriculum learning with "teacher-learner" frameworks [50, 52] was also used to facilitate label propagation.

5.3 Self-Reinforced Diffusion

5.3.1 Regularization Framework

The proposed SRD can be formulated by minimizing the following cost function:

$$\mathbf{A}^* = \arg \min_{\mathbf{A}\in\mathbb{R}^{n\times n}} \left(\mu \sum_{i,j=1}^{n} \Big(A_{ij} - Y_{ij}\Big)^2 + \frac{1}{2} \sum_{i,k,j,m=1}^{n} Z_{ik} Z_{jm} \Big(\frac{A_{ij}}{\sqrt{d_i d_j}} - \frac{A_{km}}{\sqrt{d_k d_m}} \Big)^2 \right), \tag{5.5}$$

where $\mathbf{Z} \in \mathbb{R}^{n\times n}$ and $\mathbf{Y} \in \mathbb{R}^{n\times n}$ are the proposed *self-reinforced affinity matrix* and *self-reinforced manifold neighbor indicator* respectively, which are the core innovations of our

algorithm. They are defined as follows:

$$Z_{ij} = \begin{cases} d_i & \text{if } i = j \\ W_{ij} & \text{if } i \neq j \end{cases}, \quad Y_{ij} = \begin{cases} d_i & \text{if } i = j \\ W_{ij} & \text{if } i \neq j \text{ and } y_i = y_j \\ 0 & \text{otherwise} \end{cases}, \tag{5.6}$$

where d_i is the node degree of i, *i.e.*, the sum of i-th row of $\mathbf{W}$.

As shown in Eq.(5.5), the proposed regularization framework consists of two parts: the *supervised fitting* term and the *unsupervised smoothness* term. The first term attempts to fit affinities related to the initial label information, and the second term imposes a graph smoothness constraint, forcing the neighbor nodes on the graph to have a large affinity value. The balance between these two terms is controlled by a positive parameter μ. Although the proposed regularization framework appears to be similar to the LGC [166], and the regularized diffusion process (RDP) [6], we argue that the intuition behind these terms is different. While both LGC and our proposed framework can be considered as attempting to smooth a function on the graph, LGC focused on the classification function, whereas we learn the pairwise affinities. RDP attempts to fit the initial affinity matrix $\mathbf{W}$ only for unsupervised retrieval scenarios, while we are able to fit the affinities related to the label information. The underlying assumption in our framework is that two data points lie on the same sub-manifold if their labels are the same, and a large affinity should be assigned between them. Most importantly, our method differs from all the existing methods by introducing self-affinity to both the fitness and the smoothness term. The self-affinity in the fitness term will encourage the learned affinity to fit the data density distribution, while the self-affinity in the smoothness term will be effective in adjusting the degree of impact of a particular point. The intuition behind this is that the data points lie in a higher density region (with large self-affinity) should spread its affinities more quickly. We will elaborate on why the self-affinity is of great importance in Section 5.3.2 and verify its effect in the experiments presented in Section 5.4.3.

Following the same algebraic derivation as in [6], the objective function in Eq.(5.5) can be transformed to:

$$\mathcal{J} = vec(\mathbf{A})^\top \mathbb{L} vec(\mathbf{A}) + \mu \left\| vec(\mathbf{A}) - vec(\mathbf{Y}) \right\|^2, \tag{5.7}$$

where $\mathbb{L} = I - \mathbb{S}$ is the normalized tensor graph Laplacian. $\mathbb{S} \in \mathbb{R}^{N^2 \times N^2}$ is the Kronecker product $\mathbf{S} \bigotimes \mathbf{S}$, given $\mathbf{S} = \mathbf{D}_z^{-1/2}\mathbf{Z}\mathbf{D}_z^{-1/2}$. $vec : \mathbb{R}^{m \times n} \to \mathbb{R}^{mn}$ is an operator that stacks the columns of a matrix into a column vector. Its inverse is denoted as vec^{-1}.

Taking the partial derivative of $\mathcal{J}$ with respect to $vec(\mathbf{A})$, we have

$$\frac{\partial \mathcal{J}}{\partial vec(\mathbf{A})} = 2\,(I - \mathbb{S})vec(\mathbf{A}) + 2\mu\,\Big(vec(\mathbf{A}) - vec(\mathbf{Y})\Big). \tag{5.8}$$

Setting the derivative Eq.(5.8) to zero, we obtain

$$vec(\mathbf{A}) = \frac{\mu}{\mu+1}\Big(I - \frac{1}{\mu+1}\mathbb{S}\Big)^{-1} vec(\mathbf{Y}). \tag{5.9}$$

Applying vec^{-1} on both sides and setting $\alpha = \frac{1}{\mu+1}$, we obtain the closed form solution for the target affinity matrix $\mathbf{A}$:

$$\mathbf{A}^* = (1-\alpha)\, vec^{-1}\left((I - \alpha\mathbb{S})^{-1}\, vec\,(\mathbf{Y})\right). \tag{5.10}$$

Note that even though the closed-form solution can be obtained, it is impractical to use due to the inverse of a tensor. In the next section, we introduce an efficient iterative framework that converges to the same solution as in Eq.(5.10).

5.3.2 Iterative Framework

The closed form solution in Eq.(5.10) can be obtained by iterating the following update strategy:

$$\mathbf{A}^{(t+1)} = \alpha\mathbf{S}\mathbf{A}^{(t)}\mathbf{S}^\top + (1-\alpha)\,\mathbf{Y}. \tag{5.11}$$

Following the convention of the regularized framework, the iterative framework also contains two parts. However, different intuitions can be drawn from the iterative process. Considering $\mathbf{A}$ as a function on the graph defined by $\mathbf{S}$, the optimal solution of $\mathbf{A}$ is a function that should be as smooth as possible with respect to the manifold structure revealed by the labeled and unlabeled data points. The iterative process attempts to achieve this by iteratively spreading the function value on the graph (the first term in Eq.(5.11)) while leveraging the initial label information (the second term in Eq.(5.11)).

The proof of the convergence of the iterative formula is straightforward, given the eigen-

values of $\mathbf{S}$ are bounded in [-1,1]. By running the iteration after t times and applying the operation *vec* on both sides, we can obtain:

$$vec\left(\mathbf{A}^{(t+1)}\right) = (\alpha\mathbb{S})^t\, vec\left(\mathbf{A}^{(1)}\right) + (1-\alpha)\sum_{i=0}^{t-1}(\alpha\mathbb{S})^i\, vec\left(\mathbf{Y}\right). \tag{5.12}$$

Since the spectral radius of $\mathbf{S}$ is bounded in [-1,1] and $0 < \alpha < 1$, we have

$$\lim_{t\to\infty}(\alpha\mathbb{S})^t = 0, \quad \lim_{t\to\infty}\sum_{i=0}^{t-1}(\alpha\mathbb{S})^i = (I-\alpha\mathbb{S})^{-1}. \tag{5.13}$$

Hence, Eq.(5.12) converges to

$$\lim_{t\to\infty} vec\left(\mathbf{A}^{(t+1)}\right) = (1-\alpha)\,(I-\alpha\mathbb{S})^{-1}\, vec\left(\mathbf{Y}\right). \tag{5.14}$$

Applying vec^{-1} on both sides, we obtain the same solution as in Eq.(5.10). Note that the solution is independent of $\mathbf{A}^{(0)}$. In practice different initializations only influence the convergence speed.

For a better understanding of the process, it can again be interpreted as a Markov Random Walk on the graph, where the edge weights define the probability of walking from one node to another. The function value on node i then becomes the probability distribution of walking from i to all other nodes. As the random walk goes on, the probability distribution is refined by taking information from both the transition probability defined by the graph and the additional probability of walking among nodes with the same labels. More relations between the iterative process and random walk can be found in [127, 168].

By interpreting the iterative process as a random walk, it naturally leads to a question: can a random walker walk from one node to itself? Or in other words, does the node in the graph have self-affinity? To answer these questions, let us examine the simplest diffusion process. Given an initial affinity matrix $\mathbf{W}$, a transition matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ can be defined, with P_{ij} representing the probability of walking from i to j in one step. The probability of walking from i to j in two steps $P_{ij}^{(2)}$can be obtained by:

$$P_{ij}^{(2)} = \sum_{k=1}^{n} P_{ik}P_{kj} = \sum_{k\neq i,j} P_{ik}P_{kj} + (P_{ii} + P_{jj})\,P_{ij}. \tag{5.15}$$

It can be seen that the probability consists of two parts, which are neighbor-diffused probability and self-reinforced probability, respectively. If all the self-affinities are initialized as zeros, *i.e.*, $P_{ii} = 0$, the probability of $P_{ij}^{(2)}$ will be estimated using the diffused probabilities P_{ik} and $P_{kj}, \forall k \neq i, j$ with probability P_{ij}, totally ignored. The impact of such a difference will be increased as the probabilities are spread iteratively over the entire graph. In fact, the self-affinities are implicitly introduced after the first iteration even if they are all initialized as zeros since $\sum_{k \neq i,j} P_{ik} P_{kj}$ is generally not zero (which verifies the existence of the so-called self-affinity). However, it is demonstrated in the experiments Section 5.4.3 that it is still important for the self-affinities to be initialized before the first iteration, due to their capability of modeling data density distribution.

The existence of the self-affinity brings another question: how should we initialize these self-affinities? To the best of our knowledge, this initialization is rarely discussed. They are usually clamped to zeros [166, 168] or set to be one (due to the widely-used Gaussian kernel which produces one for zero distance) without given any discussion. Note that if GSSL methods are used directly without involving any diffusion process, the self-affinity does not matter at all, as the label of a point will never be propagated to itself. However, when the affinity value is the target, it is of importance as the self-affinity of a given point affects how fast all affinities related to this point will spread, as demonstrated in Eq.(5.15).

Random walk on a graph can be used again to illustrate the idea here. A node should have a higher probability of walking back to itself if it has many short connections to the nearby points. In other words, a data point lying in a higher density area should have a larger self-affinity. Hence, the initial self-affinity is set to be the *node degree* on the graph, which is proportional to the weight of its connections, and accordingly, the local density of that point. As the self-affinity somewhat represents the data distribution, we also add it to the supervised signal $\mathbf{Y}$ in Eq.(5.5), constraining it to be something that should not change too much. Experiments confirm that this simple change to the self-affinity initialization can benefit each iteration of the diffusion process, leading to significant performance improvement of the GSSL methods.

Algorithm 4 Self-Reinforced Diffusion (SRD).

Input: a weight matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$, the known class labels y_i, the tradeoff parameter α

1: Obtain $\mathbf{Z}$ and $\mathbf{Y}$ as per Eq.(5.6).
2: Symmetrically normalize $\mathbf{Z}$: $\mathbf{S} \leftarrow \mathbf{D}_z^{-1/2}\mathbf{Z}\mathbf{D}_z^{-1/2}$, where D is a diagonal matrix with $D_{ii} \leftarrow \sum_{j=1}^{n} Z_{ij}$.
3: Initialize the iterator $\mathbf{A}^{(0)} = \mathbf{S}$, $t = 0$.
4: **while** not converged **do**
5: $\quad \mathbf{A}^{(t+1)} \leftarrow \alpha \mathbf{S}\mathbf{A}^{(t)}\mathbf{S}^\top + (1-\alpha)\,\mathbf{Y}$
6: $\quad t \leftarrow t+1$
7: **end while**

Output: The optimal affinity matrix $\mathbf{A}$ and the corresponding SRD graph.

5.4 Experiments

Experiments are conducted on various types of publicly available image datasets, including face [124], object [103], texture [84], and land usage datasets [156]. Multiple feature representations are used, such as raw pixel, PCA-based representation, and high-level feature representations obtained by deep convolutional networks. Three popular graph-based semi-supervised learning methods are used with the default KNN graph to set the baseline, including GFHF [173], LGC [166] and GGMC [146]. They are also used with the unsupervised diffusion method RDP [6], as well as the proposed SRD method to validate the effectiveness of the proposed method. An additional experiment is also conducted to demonstrate that the choice of self-affinity is of great importance to the success of diffusion-based GSSL methods.

Experimental setup. The initial weight matrix is obtained by Gaussian kernel, where the bandwidth σ is set to be the mean distance to the k-th nearest neighbor of all data points. All the graphs are sparsed by the k nearest neighbors with the same k. The parameters for the baseline GSSL methods are fixed to be the same as reported in their corresponding paper. The regularization parameter μ in RDP and SRD is set to be 0.18, as suggested in [6]. The maximum number of iteration is set to be 50 (SRD generally converges within 20 iterations).

5.4.1 Synthetic Data

We construct 5 sub-manifolds $\{S\}_{i=1}^{5} \subset \mathbb{R}^{100}$ whose bases $\{\mathbf{U}_i\}_{i=1}^{5}$ are obtained by $\mathbf{U}_i = \mathbf{T}_i\mathbf{U}, 1 \leq i \leq 5$, where $\mathbf{U}$ is a random orthogonal matrix of dimension 100×5 and $\mathbf{T}_i^{100\times 100}$

is a random rotation. 100 data points are sampled from each manifolds by $\mathbf{X}_i = \mathbf{U}_i\mathbf{Q}_i$, where the entries of $\mathbf{Q}_i \in \mathbb{R}^{5\times 50}$ are $i.i.d$ samples from a standard Gaussian. All three GSSL methods achieve perfect performances with or without the diffusion process on this clean toy dataset. To demonstrate how these methods would perform given noisy graph, certain percentage of data samples x are randomly chosen to be corrupted by Gaussian noise with zero mean and variance $0.3||x||$.

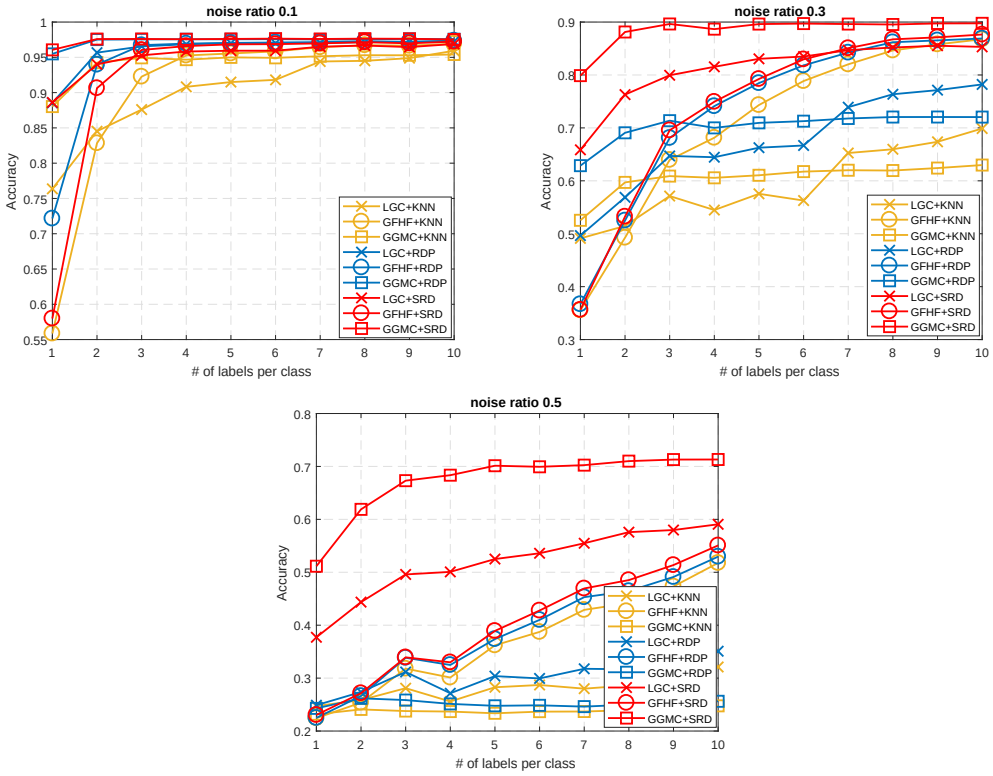

Figure 5-1: Classification performances of graph-based semi-supervised learning methods with the graphs produced by (1)KNN, (2)RDP, (3)SRD on synthetic data, given different noise levels.

As shown in Fig.5-1, when only 10% data are corrupted by noise, the performances of LGC and GFHF with the default KNN graph start to decrease. GGMC performs very well even with few labeled points, as it updates the label $\mathbf{Y}$ in each iteration. When noise samples reach 30%, it can be clearly seen that these GSSL methods with the KNN graph are outperformed by the corresponding method with the RDP graph, while the proposed SRD graph outperforms RDP with a large margin. The improvements by SRD are more significant when half of the data are corrupted, where RDP can only achieve limited improvements over KNN.

By gradually adding noise to the synthetic data, it can be seen that GSSL methods are susceptible to the data noise. The success of these GSSL methods heavily depends on the appropriate construction of the graph, as shown by the significant performance differences with different graphs. The heuristic Gaussian kernel defined on the pairwise Euclidean distance can not reveal the manifold structure given the existence of noise. The diffusion process augments and re-estimates these affinity values by spreading them along with the graph. The obtained pairwise affinities can be considered as an integration of all possible paths between the corresponding data pair on the graph, which is more robust to the partial noise. With the current state-of-the-art diffusion process RDP, the improvements are limited since they attempt to fit the noisy initial affinity matrix ($\mathbf{Y} = \mathbf{W}$), while in the proposed SRD, the fitness term contains the supervised label information, indicating the appropriate manifold neighbors, and the data distribution, reflected by the self-affinity. By iteratively integrating graph smoothness, label consistency, and data distribution, the proposed SRD is capable of learning a graph that adequately captures the underlying data manifolds, resulting in a significant improvement for all GSSL methods.

5.4.2 Real-World Data

The proposed framework is then tested on real-world image data. The first part of the experimental data consists of all the 20 frontal faces with no expression and pose in CMU-PIE database [124] for all the 68 subjects. The images are normalized and cropped to 32×32 resolution. PCA is then used to reduce the dimensionality by keeping 99% of the information, resulting in a 138 dimension vector per image. The neighbor size k is set to be 7. The second part of the data is the MPEG-7 dataset [83], containing 1,400 silhouette images from 70 classes, where each class has 20 different shapes. The IDSC shape descriptor [93] is used to calculate the pairwise distance. k is set to be 10 for this dataset. The last part of the data is the Texture-25 [84], containing 25 classes of texture with 40 samples per class. The features are obtained by a pre-trained VGG-net [125]. For this part k is set to be 10.

As shown in Fig.5-2, the diffusion processes RDP and SRD both show improvements over all the baseline GSSL methods, while the proposed SRD consistently gives the best results. As the labeled points increase, SRD continuously brings in reasonable accuracy gain as the label information is made full use iteratively, while RDP shows limited improvements compared to SRD. More interestingly, it is observed that all three GSSL methods combined

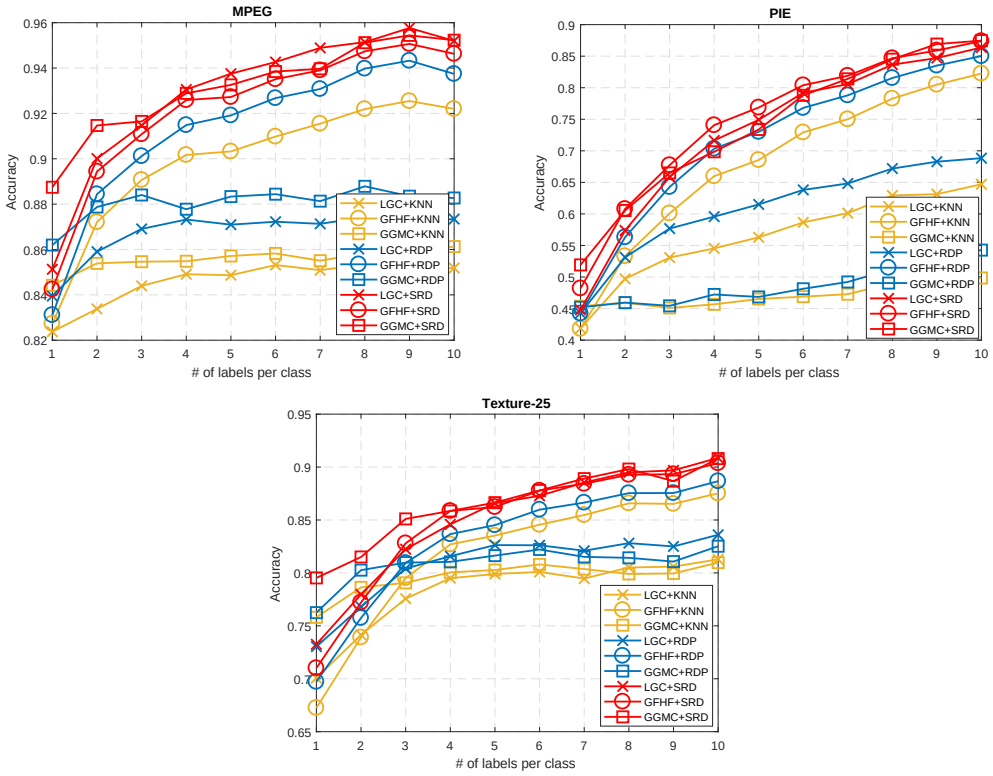

Figure 5-2: Classification performances of graph-based semi-supervised learning methods with the graphs produced by (1)KNN, (2)RDP, (3)SRD on real-world data.

with SRD (red line) perform very similarly, while their differences are significant when they are used with KNN and RDP. It can, therefore, be assumed that the graph obtained by SRD reaches an optimum with respect to the underlying data structure. It also verifies that these GSSL methods are somewhat similar, and it is more important to construct a good graph rather than to choose between different label propagation methods with similar ideas, as claimed in the survey paper [170].

5.4.3 Initialization of Self-Affinity

As shown by the random walk interpretation Eq.(5.15), the self-affinity is implicitly defined even if they are initialized as zero. Due to the iterative formula and affinity spreading property of the diffusion process, the initialization of self-affinity is of great importance. We hypothesis that the capability of the diffusion process in revealing manifold structure is hinged on having an appropriate initialization of the self-affinity.

We conduct three experiments on various types of data, including COIL-20 [103], ORL,

and LANDUSE-21 [156]. COIL-20 contains 1,440 gray-scale images of 20 objects with 72 images per class. The image size is 32×32, and the raw pixel is used as the feature. ORL is a face image database consisting of 40 subjects with 10 images per subject under various conditions of the pose, illumination, and expression. The face is cropped to 32×32 and represented by the raw pixel. LANDUSE contains 2,100 satellite images of 21 classes of land usage, and the features are extracted via off-the-shelf VGG-net. GSSL methods are tested with the proposed SRD, given different initializations of the self-affinity, including 0, 1, and d (node degree), respectively.

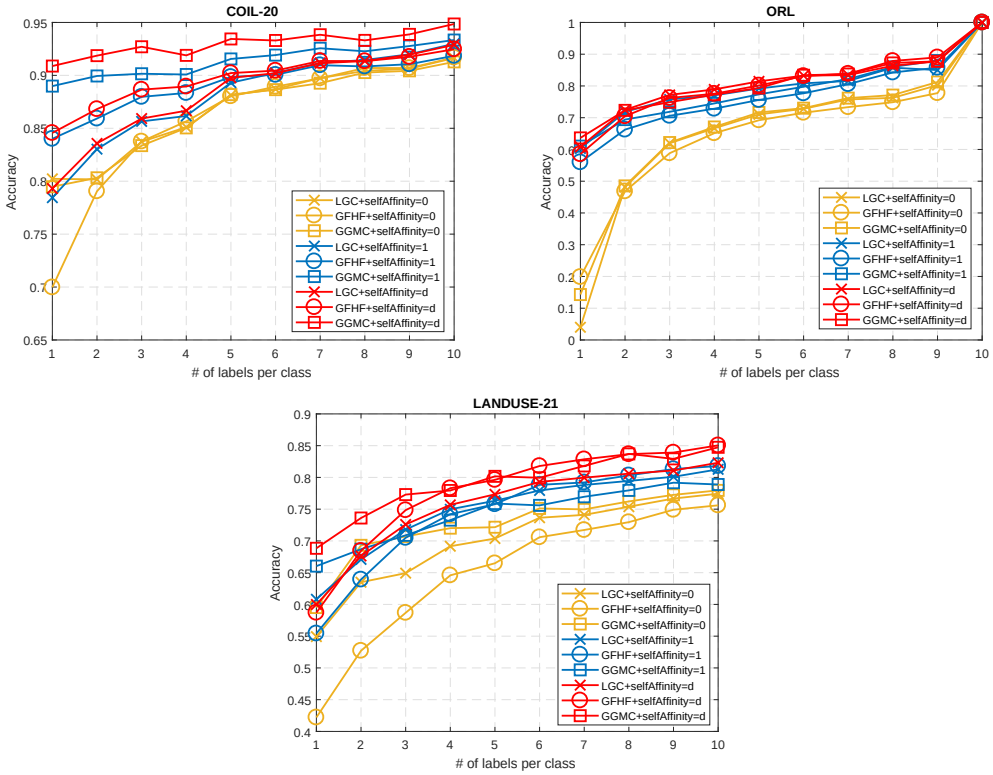

Figure 5-3: Classification performances of SRD-graph based semi-supervised learning methods with different choice of self-affinities (1)self-affinity=0, (2)self-affinity=1, and (3)self-affinity=d.

It is clearly shown in Fig.5-3 that different initialization of self-affinity will impact the diffusion process, and therefore the GSSL methods. Setting self-affinity to zero consistently leads to the worst results, while using the proposed node degree as self-affinity performs the best. If considering the self-affinity as a measurement of data density distribution, setting it to zero represents the distribution is totally ignored during the iterative process, while setting it to 1 means a uniform distribution. On the contrary, setting it to the node degree

is a more appropriate choice, as it attempts to estimate the local density of a particular point by using its affinities to all its neighbors. The significant improvements of setting self-affinity to d over zero self-affinity in the early stage also verify that taking data distribution into consideration is crucial especially when few labeled points are available, as they are combined together to be the fitness target ($\mathbf{Y}$ in Eq.(5.5)). SRD can fit the data density when there is no label to match.

5.5 Summary

Constructing an appropriate graph is the key to the success of most GSSL methods. The widely used KNN graph with the Gaussian kernel is heuristically built and susceptible to data noise. Sensitive kernel parameter often requires extra efforts of practitioners before the model can perform well for a specific problem. Although the diffusion process is promising in learning manifold-aware graphs, data density distribution and label information are totally ignored by the existing diffusion methods, limiting its usage to GSSL methods. Aimed at constructing a robust graph that can deal with data noise and cope with semi-supervised labels, we proposed a variant of the diffusion process, named the Self-Reinforced Diffusion, that can be used as a generic tool to improve all GSSL methods. It is formulated as a regularization framework with two terms: *fitness* and *smoothness*. While the former explicitly models the data density and label information, the later regularizes it on a higher-order tensor graph. An efficient iterative formula that guaranteed to converge to the same closed solution as the regularization framework is also developed. Extensive experiments on noisy synthetic data and various real-world data have demonstrated the effectiveness of the proposed method. Additional experiments are also conducted to verify the importance of self-affinity.

In this chapter, it is demonstrated that the pre-existing labels, incorporated with the diffusion process, can be beneficial for graph construction. Can the predicted labels, or pseudo labels, also be utilized to construct an even better graph? The answer is yes. Pseudo labels, used in a proper way, can reveal the pairwise sample similarity in the "label space", which can further provide weak supervision to the process of affinity learning. In the next chapter, we show how to effectively make use of pseudo labels obtained from label propagation to help learn better affinities.

Chapter 6

Semi-Supervised Learning with Alternating Diffusion Processes

In Chapter 5, the importance of affinity learning in graph-based semi-supervised learning (GSSL) has been demonstrated, where it is shown that the diffusion process, incorporated with label information and self-similarity, can improve the performance of GSSL methods significantly. In this chapter, the discussion on GSSL is continued, where we attempt to integrate the two stages of GSSL, namely affinity learning and label propagation into one unified framework. The motivation is that if the initially given labels can help construct a better affinity graph, it is plausible that the predicted labels of unlabeled points from label propagation are also beneficial for graph construction. Hence, if affinity learning and label propagation can be performed alternately or jointly, it is possible to fully exploit the correlation between these two stages, resulting in a more accurate graph and even better classification accuracy.

6.1 Introduction

Semi-supervised learning (SSL) makes use of unlabeled data to produce a considerable improvement in the learning accuracy of the fully supervised model. The basic idea is that a large amount of unlabeled data can provide additional information about the data distribution that is potentially beneficial for decision making. Graph-based semi-supervised learning (GSSL) models such underlying data distribution under the manifold assumption, which states that the high-dimensional data lie approximately on a manifold of much lower

dimension. Learning the manifold using a graph constructed on both labeled and unlabeled points can avoid the curse of dimensionality, which makes GSSL very attractive for high-dimensional problems [39, 175].

GSSL methods usually consist of two separate stages, namely affinity learning and label propagation. A typical GSSL method first constructs an affinity graph using the Gaussian kernel or sparse representation, and then inference unknown labels by a label propagation method. Despite its significance, the label information is often used for label propagation only, while being ignored during the process of affinity learning. In Chapter 5, it is demonstrated that label information is also valuable for learning better affinity by incorporating them into the diffusion process. Researchers in [176] embedded the label information as constraints to a sparse coding problem and achieved better results.

The observation that class labels can provide helpful supervision to the affinity learning process brings another question: can predicted labels help construct the affinity graph as well? This is a reasonable hypothesis, as it is possible to use "self-training" to boost the performance of a classifier, as discussed in Section 2.3.2. However, it is not clear how to effectively use the predicted labels as feedbacks since naively change the label information means the change of the constraints as in [176], and it leads to diverging of the algorithm. Similarly, if we keep changing the matrix $\mathbf{Y}$ in Eq.(5.5), the Self-Reinforced Diffusion (SRD) algorithm proposed in Chapter 5 will not converge at all. Another common problem for almost all GSSL methods is that they can make predictions only for data points appeared in training, but not for any unseen data, which are referred to as *transductive* methods. The reason is that GSSL makes use of the graph structure to inference unknown labels, which requires a data point to be involved in the stage of graph construction. However, in many applications, it is desirable to have an *inductive* method that can make predictions for unseen data on the fly. The process of making a transductive method to be inductive is sometimes referred to as the out-of-sample extension.

To make use of the predicted labels effectively, we propose to use them to explicitly calculate pairwise *label similarity*. Instead of using the labels as constraints, which causes convergence problems, the label similarity is safely added to the existing affinity values derived from the feature space, as a complementary measurement of pairwise similarity. After adding the label similarity, we formulate affinity learning as a novel regularized function estimation problem, similar to label propagation. We show that these two function estimation

problems can be solved efficiently by two iterative diffusion processes that are fundamentally similar. Most importantly, by alternately or jointly running these two diffusion processes, we are allowed to fully interact with these two stages in an iterative manner, as predicted labels can be effectively fed back to the affinity learning stage. In this way, labels are continuously propagated on a *dynamic* graph updated iteratively with the supervision of predicted labels obtained in the previous iteration. As the proposed method can be interpreted as a process of alternating propagation of both affinities and labels, we name it the Alternating Diffusion process (ADP). Fig.6-1 illustrates a brief workflow of the proposed ADP method. The second challenge is to make ADP inductive. We show that it can be efficiently achieved by using the proposed regularization framework, in which we directly optimize the regularized optimization problem with respect to an unseen data point, as shown in Section 6.3.4.

The main contributions of this chapter can be summarized as:

- We propose to make use of the predicted labels in GSSL by introducing the label similarity, and formulate the affinity learning with label similarity as a regularized optimization problem.
- We propose an iterative diffusion process that can fully exploit the given and predicted labels to solve the optimization problem efficiently.
- We integrate affinity learning and label propagation into one unified iterative diffusion framework, allowing them to be updated simultaneously.
- We propose an out-of-sample extension of ADP, which scales linearly with respect to the number of samples.

6.2 Related Work

State-of-the-art affinity learning methods are usually based on a self-expressive model [36], in which a data point is represented by a linear combination of all other data points. This type of affinity learning method achieve impressive results not only in unsupervised learning, *e.g.*, clustering as shown in Chapter 3, but also in semi-supervised learning (SSL) as shown in Chapter 5. To make best use of valuable label information in SSL, a recent work [176] explicitly added label information as constraints in a low-rank representation problem. It can

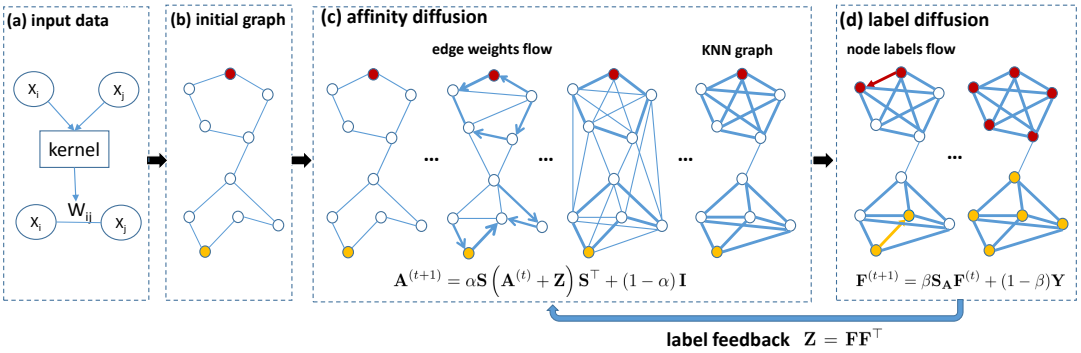

Figure 6-1: The workflow of the proposed ADP method. (a) Obtain pairwise data similarity (*e.g.*, the Gaussian kernel) (b) It first builds an initial KNN graph, and then (c) edge weights spread on the graph with the supervision of the original node labels. (d) Finally, node labels are propagated on the graph weighted by the optimal affinity matrix. Note that steps (c) and (d) are repeated until converging in order to fully exploit the given and predicted labels to guide the affinity learning.

be seen as a semi-supervised variant of the self-expressive model, which can be formulated as:

$$\begin{aligned} \min_{\mathbf{C},\mathbf{E}} \quad & \|\mathbf{C}\|_* + \lambda \|\mathbf{E}\|_{2,1} \\ \text{s.t.} \quad & \mathbf{X} = \mathbf{X}\mathbf{C} + \mathbf{E}, \ Z_{ij} = 0, \ \forall (i,j) \in \Omega, \end{aligned} \tag{6.1}$$

where $\mathbf{C}$ is the target coefficient matrix, and $(i,j) \in \Omega$ represents x_i and x_j have different class labels. The only difference compared to the unsupervised Low-Rank Representation [94] is the additional constraints on the labeled points that prevents the usage of data points from different classes. The final affinity matrix is obtained by $\mathbf{A} = |\mathbf{C}| + |\mathbf{C}|^\top$, resulting in zero affinity between x_i and x_j for any $(i,j) \in \Omega$.

Instead of calculating the affinity matrix by post-processing the coefficient matrix, it is suggested to learn the affinity matrix directly by adding a fitness term to the self-expressive model [39]. They formulated the problem as follow:

$$\begin{aligned} \min_{\mathbf{C},\mathbf{E},\mathbf{A}} \quad & \|\mathbf{C}\|_* + \lambda \|\mathbf{E}\|_{2,1} + \gamma \|\mathbf{C} - \mathbf{A}\|_F^2 \\ \text{s.t.} \quad & \mathbf{X} = \mathbf{X}\mathbf{C} + \mathbf{E},\ \sum_j A_{ij} = 1,\ A_{ij} \geqslant 0,\ \forall (i,j) \end{aligned} \tag{6.2}$$

where only a soft constraint is imposed between the coefficient $\mathbf{C}$ and the affinity matrix $\mathbf{A}$, rather than a hard equality constraint. It is claimed that the soft constraint gives more flexibility of the affinity matrix to fit the underlying data structure. Note that there also proposed other rank constraints to enforce the block-diagonal structure for the affinity matrix $\mathbf{A}$, please refer to [39] for more details.

The self-expressive model was also used in [87] with an additional term that targets at learning the final label matrix directly. The problem is formulated as:

$$\begin{aligned} \min_{\mathbf{C},\mathbf{E},\mathbf{Y}} \quad & \|\mathbf{C}\|_* + \lambda \|\mathbf{E}\|_{2,1} + \gamma \delta(\mathbf{C}, \mathbf{Y}) \\ \text{s.t.} \quad & \mathbf{X} = \mathbf{X}\mathbf{C} + \mathbf{E},\ Y_\ell = Y_\ell^0, \end{aligned} \tag{6.3}$$

where Y_ℓ is the predicted label matrix and Y_ℓ^0 is the given label matrix of the labeled points. $\delta(\mathbf{Z}, \mathbf{Y})$ represents a loss function that quantifies the disagreement between the affinity matrix $\mathbf{A} = |\mathbf{C}| + |\mathbf{C}|^\top$ and the label matrix $\mathbf{Y}$, which can be formulated as follow:

$$\delta(\mathbf{A}, \mathbf{Y}) = \sum_{i,j} \frac{1}{2} A_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 . \tag{6.4}$$

Minimizing this loss will encourage similar data points (large A_{ij}) to have similar predictions as well. More importantly, minimize the overall loss function as in Eq.(6.3) using Alternating Direction Method of Multipliers (ADMM) can learn affinity and label assignment simultaneously.

6.3 Alternating Diffusion Process

As many notations will be involved in the following discussion, we first summarize the most frequently used notations in Table 6.1.

Table 6.1: Some of important notations used in this chapter.

Notation	Description
n	Number of data points
ℓ	Number of labeled data points
c	Number of classes
$\mathbf{W}^{n\times n}$	Initial weight matrix
$\mathbf{D}^{n\times n}$	Diagonal matrix with $D_{ii} = \sum_{j=1}^{n} W_{ij}$
$\mathbf{P}^{n\times n}$	Random walk transition matrix
$\mathbf{S}^{n\times n}$	Symmetrically normalized weight matrix
$\mathbb{S}^{nn\times nn}$	Kronecker product $\mathbf{S}\otimes\mathbf{S}$
$\mathbf{A}^{n\times n}$	Learned affinity matrix
$\mathbf{Y}^{n\times c}$	One-hot encoded label matrix
$\mathbf{F}^{n\times c}$	Classification function
$\mathbf{Z}^{n\times n}$	Label similarity defined as $\mathbf{Z} = \mathbf{F}\mathbf{F}^\top$
$\mathbf{I}$	Identity matrix of appropriate size
$vec(\cdot)$	Operator that stacks matrix $\mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$

Let assume an initial weight matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$, and a one-hot encoded label matrix $\mathbf{Y} \in \mathbb{R}^{n\times c}$, *i.e.*, $Y_{ij} = 1$ if $y_i = j$ and 0 otherwise. The target is to learn a classification function $\mathbf{F} \in \mathbb{R}^{n\times c}$, such that its entry F_{ij} represents the probability of x_i belonging to the j-th class. Given any unlabeled data points x_i, its label can be set to the index of the maximum value in $F_{i\cdot}$, where $F_{i\cdot}$ represents the i-th row of $\mathbf{F}$. The classification function $\mathbf{F}$ is initialized as the given label matrix $\mathbf{Y}$, in which the rows corresponding to unlabeled points are all zero. Then, the proposed Alternating Diffusion Process (ADP) is an alternating process between the optimization of the affinity graph $\mathbf{A} \in \mathbb{R}^{n\times n}$ and the classification $\mathbf{F}$.

6.3.1 Given Classification F, Update Graph A

In this section, the optimization problem to update the affinity graph A given F is first formulated, and its closed-form solution is also derived. We then propose an iterative formula as an efficient alternative solver and show that it converges to the same solution.

Optimization Problem. Given the classification $\mathbf{F}$, we formulate the graph construction as a regularized function estimation problem:

$$\min_{\mathbf{A}} \frac{1}{2}\sum_{i,j,k,l=1}^{n} W_{ij}W_{kl}\left(\frac{A_{ki}+Z_{ki}}{\sqrt{D_{ii}D_{kk}}} - \frac{A_{lj}+Z_{lj}}{\sqrt{D_{jj}D_{ll}}}\right)^2 + \left(-2\sum_{k,i=1}^{n} A_{ki}Z_{ki}\right) + \mu\sum_{k,i=1}^{n}(A_{ki}-I_{ki})^2, \tag{6.5}$$

where $\mu > 0$ is a regularization parameter, $\mathbf{D}$ is a diagonal matrix with its element $D_{ii} = \sum_{j=1}^{n} W_{ij}$, $\mathbf{I}$ is the $n \times n$ identity matrix and $\mathbf{Z} = \mathbf{F}\mathbf{F}^\top$ is the label similarity/affinity matrix. A large element Z_{ij} indicates that the label vector $F_{i\cdot}$ is similar to $F_{j\cdot}$.

As shown in Eq.(6.5), the objective function consists of three terms. The first one is a local smoothness term, where it encourages neighbor points identified by $\mathbf{W}$ (large W_{ij}) to take similar values on the learned affinity $\mathbf{A}$. Instead of considering pairwise similarity independently on the original graph represented by $\mathbf{W}$, ADP attempts to smoothen $\mathbf{A}$ with four nodes at a time by using a higher-order tensor graph [155]. Specifically, if W_{ij} is large (x_i is similar to x_j) and W_{kl} is also large (x_k is similar to x_l), A_{ki} and A_{lj} are encouraged to be similar after adding the label similarity. The second term is a fitness term, which encourages a large A_{ki} when Z_{ki} is large. The assumption is that data points that are similar in the label space (large Z_{ki}) should be similar in the feature space (large A_{ki}) as well. The last term is a regularization term controlling the scale of $\mathbf{A}$. The regularization framework is mainly inspired by [7], where an unsupervised diffusion process was proposed. The main difference is that we add label information $\mathbf{Z}$ (both the given and predicted labels) into the formula, and the labels will provide critical supervision to the diffusion process. We empirically validate in Section 6.4.2 that label information, including the predicted labels, is indeed useful for graph construction.

It can be seen that the problem of constructing a graph $\mathbf{A}$ using Eq.(6.5) is quite similar to the problem of label propagation in Eq.(5.3). They both can be seen as estimating a local smooth function on the graph $\mathbf{W}$ while regularizing its global fitness to the labels. The main difference is that the smooth function is defined on graph nodes (node labels) in label propagation, while the target function here is node affinity defined on the edges of the graph.

Following some similar algebraic derivations as [7], the objective function Eq.(6.5) can be transfered to:

$$\begin{aligned} \mathcal{J} = \; & vec(\mathbf{A}+\mathbf{Z})^\top \mathbb{L} vec(\mathbf{A}+\mathbf{Z}) \\ & - 2vec(\mathbf{A})^\top vec(\mathbf{Z}) + \mu \left\| vec(\mathbf{A}) - vec(\mathbf{I}) \right\|^2, \end{aligned} \tag{6.6}$$

where $\mathbb{L} = \mathbf{I} - \mathbb{S}$ is the normalized graph Laplacian of the tensor product graph, and $\mathbf{I}$ is the identity matrix of appropriate size. $\mathbb{S}^{nn \times nn}$ is the Kronecker product $\mathbf{S} \otimes \mathbf{S}$, where

$\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$. $vec : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ is an operator that stacks the columns of a matrix into a column vector. Its inverse is denoted as vec^{-1}.

Taking the partial derivative of $\mathcal{J}$ with respect to $vec(\mathbf{A})$, we have

$$\begin{aligned}\frac{\partial \mathcal{J}}{\partial vec(\mathbf{A})} = \; & 2\left(\mathbf{I} - \mathbb{S}\right)\left(vec(\mathbf{A}) + vec(\mathbf{Z})\right) \\ & - 2vec(\mathbf{Z}) + 2\mu\left(vec(\mathbf{A}) - vec(\mathbf{I})\right). \end{aligned} \tag{6.7}$$

Setting this derivative to zero, we obtain

$$\begin{aligned} vec(\mathbf{A}) = \; & \frac{1}{\mu+1}\mathbb{S}\left(\mathbf{I} - \frac{1}{\mu+1}\mathbb{S}\right)^{-1} vec(\mathbf{Z}) \\ & + \frac{\mu}{\mu+1}\left(\mathbf{I} - \frac{1}{\mu+1}\mathbb{S}\right)^{-1} vec(\mathbf{I}). \end{aligned} \tag{6.8}$$

After setting $\alpha = \frac{1}{\mu+1} \in (0,1)$ and applying vec^{-1} on both sides of Eq.(6.8), the closed-form solution can be obtained as

$$\begin{aligned} \mathbf{A}^* = \; & vec^{-1}\Big(\Big((\mathbf{I} - \alpha\mathbb{S})^{-1} - \mathbf{I}\Big)\, vec\left(\mathbf{Z}\right) \\ & + \left(1-\alpha\right)\left(\mathbf{I} - \alpha\mathbb{S}\right)^{-1} vec\left(\mathbf{I}\right)\Big). \end{aligned} \tag{6.9}$$

Note that $(\mathbf{I} - \alpha\mathbb{S})^{-1}$ is a diffusion kernel [166]. The solution illustrates that the optimal affinity $\mathbf{A}$ is a balanced combination of diffusions of the label similarity $\mathbf{Z}$ and the prior similarity $\mathbf{I}$. It is shown in [90] that the diagonal matrix $\mathbf{D}$ is a better prior than $\mathbf{I}$ to adapt the underlying data distribution.

Although we can obtain the closed-form solution, it is impractical to use it directly due to its prohibitive computation of the matrix inverse. Hence, we introduce an efficient iterative diffusion process, which converges to the same solution as Eq.(6.9).

Iterative Solver. The closed-form solution for the problem (6.5) can be efficiently obtained by running the following iteration:

$$\mathbf{A}^{(t+1)} = \alpha\mathbf{S}\left(\mathbf{A}^{(t)} + \mathbf{Z}\right)\mathbf{S}^\top + \left(1-\alpha\right)\mathbf{I}. \tag{6.10}$$

As shown in Eq.(6.10), the affinity matrix $\mathbf{A}$ can be learned by iteratively spreading the previous affinity values with the label similarity, while continuously drawing information

from the prior affinity $\mathbf{I}$.

Next, we prove the convergence of the iteration. By applying the operator *vec* on both sides of Eq.(6.10), we obtain

$$\begin{aligned} vec\left(\mathbf{A}^{(t+1)}\right) = \; & vec\left(\alpha \mathbf{S}\mathbf{A}^{(t)}\mathbf{S}^{\top}\right) \\ & + vec\left(\alpha \mathbf{S}\mathbf{Z}\mathbf{S}^{\top}\right) + (1-\alpha)\, vec\left(\mathbf{I}\right). \end{aligned} \tag{6.11}$$

As $vec\left(\mathbf{A}\mathbf{B}\mathbf{C}^{\top}\right) = \left(\mathbf{C} \bigotimes \mathbf{A}\right) vec\left(\mathbf{B}\right)$, Eq.(6.11) can be rewritten as

$$\begin{aligned} vec\left(\mathbf{A}^{(t+1)}\right) = \; & \alpha \mathbb{S} vec\left(\mathbf{A}^{(t)}\right) \\ & + \alpha \mathbb{S} vec\left(\mathbf{Z}\right) + (1-\alpha)\, vec\left(\mathbf{I}\right). \end{aligned} \tag{6.12}$$

By running the iteration t times, we obtain

$$\begin{aligned} vec\left(\mathbf{A}^{(t+1)}\right) = \; & (\alpha \mathbb{S})^{t}\, vec\left(\mathbf{A}^{(1)}\right) + \sum_{i=1}^{t} (\alpha \mathbb{S})^{i}\, vec\left(\mathbf{Z}\right) \\ & + (1-\alpha) \sum_{i=0}^{t-1} (\alpha \mathbb{S})^{i}\, vec\left(\mathbf{I}\right). \end{aligned} \tag{6.13}$$

Since the eigenvalues of $\mathbf{S}$ are bounded in $[-1, 1]$ and $0 < \alpha < 1$, we have

$$\lim_{t\to\infty} (\alpha \mathbb{S})^{t} = 0, \qquad \lim_{t\to\infty} \sum_{i=0}^{t-1} (\alpha \mathbb{S})^{i} = (\mathbf{I} - \alpha \mathbb{S})^{-1}. \tag{6.14}$$

Hence, Eq.(6.13) converges to

$$\begin{aligned} \lim_{t\to\infty} vec\left(\mathbf{A}^{(t+1)}\right) = \; & \alpha \mathbb{S} \left(\mathbf{I} - \alpha \mathbb{S}\right)^{-1} vec\left(\mathbf{Z}\right) \\ & + (1-\alpha) \left(\mathbf{I} - \alpha \mathbb{S}\right)^{-1} vec\left(\mathbf{I}\right). \end{aligned} \tag{6.15}$$

After applying vec^{-1} on both sides of Eq.(6.15), we obtain the solution

$$\begin{aligned} \mathbf{A}^{*} = \; & vec^{-1}\left(\left(\left(\mathbf{I} - \alpha \mathbb{S}\right)^{-1} - \mathbf{I}\right) vec\left(\mathbf{Z}\right) \right. \\ & \left. + (1-\alpha) \left(\mathbf{I} - \alpha \mathbb{S}\right)^{-1} vec\left(\mathbf{I}\right)\right), \end{aligned} \tag{6.16}$$

which is exactly the same as Eq.(6.9) obtained by solving the optimization problem Eq.(6.5).

Note that the solution is independent of the initialization of $\mathbf{A}$. In our experiments, $\mathbf{A}$ is initialized as $\mathbf{S}$ for a faster convergence speed.

6.3.2 Given Graph A, Update Classification F

Once a graph is constructed, the next step is to propagate label on it, where standard label propagation methods can be directly used, such as GFHF [173], LGC [166], and GGMC [146]. We use LGC with the learned affinity matrix $\mathbf{A}$:

$$\min_{\mathbf{F}} \frac{1}{2} \sum_{i,j=1}^{n} A_{ij} \left\| \frac{F_{i\cdot}}{\sqrt{D_{ii}}} - \frac{F_{j\cdot}}{\sqrt{D_{jj}}} \right\|^2 + \frac{\gamma}{2} \sum_{i=1}^{n} \|F_{i\cdot} - Y_{i\cdot}\|^2 , \tag{6.17}$$

where $\gamma > 0$ is a regularization parameter, and $D_{ii} = \sum_{j=1}^{n} A_{ij}$. This is a function estimation problem similar to Eq.(6.5). It encourages similar nodes i and j (with a large A_{ij}) to have similar label predictions F_i and F_j, while regularizing the fitness of the prediction F_i to the groundtrouth Y_i.

It is not surprising that Eq.(6.17) has an equivalent iterative form:

$$\mathbf{F}^{(t+1)} = \beta \mathbf{S}_{\mathbf{A}} \mathbf{F}^{(t)} + (1 - \beta)\mathbf{Y}, \tag{6.18}$$

where $\mathbf{S}_{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. Eqs.(6.17) and (6.18) converge to the same closed-form solution, $F^* = (\mathbf{I} - \beta \mathbf{S}_{\mathbf{A}})^{-1}\mathbf{Y}$, where $(\mathbf{I} - \beta \mathbf{S}_{\mathbf{A}})^{-1}$ is another diffusion kernel [166] defined on the learned affinity graph $\mathbf{A}$ from the previous stage.

6.3.3 The Complete Algorithm and Its Variants

We have proposed a novel GSSL algorithm ADP in a two-stage fashion, where the graph and label are independently optimized by the diffusion process. One of the drawbacks of such two-stage methods is that they do not fully exploit the correlation between the graph and labels, as it has been shown by [87] that the predicted labels can provide "weakly" supervised information for building a better graph and facilitate label propagation. Unlike the variants of LGC [166], where the predicted labels are used as a re-initialization to apply another round of LGC, we feed it back a bit further to the graph construction stage to supply additional supervision for capturing the underlying manifold structure. Hence, the proposed algorithm can be seen as an alternating optimization between the affinity matrix

$\mathbf{A}$, and the classification function $\mathbf{F}$. The complete algorithm is presented in Algorithm 5.

Algorithm 5 Alternating Diffusion Process

Input: weight matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$, label matrix $\mathbf{Y} \in \mathbb{R}^{n \times c}$
Output: classification $\mathbf{F}$, affinity $\mathbf{A}$
Parameter: regularizer α, β, threshold θ

Obtain the diagonal matrix $\mathbf{D}$, where $D_{ii} \leftarrow \sum_{j=1}^{n} W_{ij}$
Normalize $\mathbf{W}$ symmetrically $\mathbf{S} \leftarrow \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$
Initialize $\mathbf{F}^{(0)} \leftarrow \mathbf{Y}$, $\mathbf{A}^{(0)} \leftarrow \mathbf{S}$, $t \leftarrow 0$
while $\left\|\mathbf{F}^{(t+1)} - \mathbf{F}^{(t)}\right\|_F > \theta$ **do**
    Obtain label similarity $\mathbf{Z} \leftarrow \mathbf{F}^{(t)}{\mathbf{F}^{(t)}}^{\top}$
    Obtain $\mathbf{A}^{(t+1)}$ by iterating Eq.(6.10) until converge
    Obtain $\mathbf{F}^{(t+1)}$ by iterating Eq.(6.18) until converge
    $t \leftarrow t + 1$
end while
return

The core of the proposed ADP is that the graph and the classification are optimized *alternately* so that the predicted labels can be fed back to construct a better graph in order to facilitate the label propagation. Motivated by the iterative diffusion processes of these two subproblems, one possible variants of ADP is to optimize $\mathbf{F}$ and $\mathbf{A}$ *jointly* by running only a single iteration for one variable before updating another, resulting in the following update strategy:

$$\begin{cases} \mathbf{F}^{(t+1)} = \beta \mathbf{S}^{(t)} \mathbf{F}^{(t)} + (1-\beta)\mathbf{Y} \\ \mathbf{A}^{(t+1)} = \alpha \mathbf{S}\left(\mathbf{A}^{(t)} + \mathbf{F}^{(t+1)}{\mathbf{F}^{(t+1)}}^{\top}\right)\mathbf{S}^{\top} + (1-\alpha)\,\mathbf{I}. \end{cases}$$

In this formula, instead of running two diffusion processes alternately, *one diffusion process consisting of two steps in each iteration* is used. It can be seen as propagating labels on a dynamic graph smoothed by the initial Gaussian weight matrix under the supervision of both the given and predicted labels. We show empirically this variant, namely *Joint Diffusion Process* (JDP), also works well in terms of semi-supervised classification accuracy.

6.3.4 Out-of-Sample Extension and Scalability

A common problem for graph-based approaches is that they are transductive in nature. There is no doubt that an inductive SSL method is of interest in certain scenarios, where the test samples are given one at a time.

To this end, we need to embed a new data point into the graph without re-running the whole algorithm. Given the training set $\mathcal{X}$ and a new data point x_q, we need to obtain $A_{qi}, \forall x_i \in \mathcal{X}$ directly. Recall that we use the cost function Eq.(6.5) to construct the graph for the training set. It represents a manifold structure with local smoothness and global fitness. The assumption here is that to embed a new point into this structure, it should follow the same manifold. Inspired by [5, 15], we can minimize the same cost function Eq.(6.5) with respect to the new data point only:

$$\min_{A_{qi}} \frac{1}{2} \sum_{j,l=1}^{n} W_{ij} W_{ql} \left(\frac{A_{qi} + Z_{qi}}{\sqrt{D_{ii} D_{qq}}} - \frac{A_{lj} + Z_{lj}}{\sqrt{D_{jj} D_{ll}}} \right)^2 + (-2 A_{qi} Z_{qi}) + \mu \left(A_{qi} - I_{qi} \right)^2, \tag{6.19}$$

where W_{ql} is the kernel weight between x_q and training data x_l that has to be obtained first. As x_q is a new data point, its label similarities and prior similarities with other training samples should be zero, *i.e.*, Z_{qi} and I_{qi} are zero. Thus, Eq.(6.19) reduces to:

$$\min_{A_{qi}} \frac{1}{2} \sum_{j,l=1}^{n} W_{ij} W_{ql} \left(\frac{A_{qi}}{\sqrt{D_{ii} D_{qq}}} - \frac{A_{lj}}{\sqrt{D_{jj} D_{ll}}} \right)^2 + \mu A_{qi}^2. \tag{6.20}$$

The partial derivative of Eq.(6.20) with respect to A_{qi} is:

$$\frac{\sum_{j,l=1}^{n} W_{ij} W_{ql}}{D_{ii} D_{qq}} A_{qi} - \sum_{j,l=1}^{n} \frac{W_{ij} W_{ql}}{\sqrt{D_{ii} D_{jj} D_{qq} D_{ll}}} A_{lj} + \mu A_{qi}. \tag{6.21}$$

By setting the derivative to zero, we obtain the closed form solution for the embedding of the new data point x_q:

$$A_{qi} = \frac{1}{1+\mu} \sum_{j,l=1}^{n} S_{ij} A_{lj} S_{ql}, \tag{6.22}$$

where $\mathbf{S} = \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2}$.

To obtain the final classification $F_{q\cdot}$, where $F_{q\cdot}$ is a row vector and F_{qi} represents the probability of x_q belonging to the i-th class, we can apply the same strategy used to obtain $A_{q\cdot}$, *i.e.*, we can minimize the cost function Eq.(6.17) with respect to $F_{q\cdot}$ only:

$$\min_{F_{q\cdot}} \frac{1}{2} \sum_{i=1}^{n} A_{qi} \left\| \frac{F_{q\cdot}}{\sqrt{D_{qq}}} - \frac{F_{i\cdot}}{\sqrt{D_{ii}}} \right\|^2 + \frac{\gamma}{2} \left\| F_{q\cdot} - Y_{q\cdot} \right\|^2. \tag{6.23}$$

The partial derivative of the cost function in Eq.(6.23) with respect to $F_{q\cdot}$ is:

$$\frac{\sum_{i=1}^{n} A_{qi}}{D_{qq}} F_{q\cdot} - \sum_{i=1}^{n} \frac{A_{qi}}{\sqrt{D_{qq} D_{ii}}} F_{i\cdot} + \gamma F_{q\cdot}. \tag{6.24}$$

Again, setting it to zero gives us the closed solution:

$$F_{q\cdot} = \frac{1}{1+\gamma} \sum_{i=1}^{n} S_{A_{qi}} F_{i\cdot}, \tag{6.25}$$

where $\mathbf{S_A} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ can be seen as a learned manifold-aware kernel, and the prediction Eq.(6.25) is a kernel weighted sum of the neighbors' labels. Interestingly, if we replace $\mathbf{S_A}$ by KNN adjacency matrix, Eq.(6.25) reduces to KNN classification. Similarly, if $\mathbf{S_A}$ is the Gaussian kernel, it is equivalent to the Parzen windows approach [15].

Complexity analysis. The computational cost of the transductive ADP is dominated by the matrix multiplication in Eq.(6.10), which is $O(n^3)$ with a plain implementation. Fortunately, the inductive part of ADP is very efficient. Assuming there are $m (m \ll n)$ training data points, it only involves pre-calculating S_{ql} in Eq.(6.22) which is $O(mn)$, obtaining A_{qi} which is $O(m^3 n)$ and the final classification F_{qi} is $O(m^2 cn)$, where c is the number of class. Overall, the computational cost of inductive inference of ADP is $O(n)$ since $m \ll n$, *i.e.*, it scales linearly with respect to the number of data points. Hence, it is pretty flexible in applying the proposed ADP method in various settings:

- When all data is available, and the cost of running ADP with all of them is acceptable, one can use the transductive version of ADP.
- When all data is available, and the cost of running ADP with all of them is not acceptable, one can use transductive ADP with a subset of data and apply inductive inference on the rest.
- When training data is given as a whole, and test data is given one by one, one can use the transductive formula for off-line training and inductive formula for on-line testing.

6.4 Experiments

Experimental setup. For all GSSL methods, the graph is constructed by the adaptive Gaussian sparsed by KNN with $k = 10$, where the bandwidth σ is set to be the mean distance

of M nearest neighbors with M set to 27 unless stated otherwise. Other hyper-parameters are set according to the corresponding authors. For the proposed ADP, the regularization parameters α and β are set to 0.99, and the threshold θ is set to 0.01. All the experiments are repeated 10 times with randomly chosen labeled points, and the average performances are reported.

6.4.1 Ablation Study on Toy Data

We construct a toy dataset as follows: five circles (classes) are drawn on a 2D plane, 100 evenly distributed points are sampled on each class, and one point per class is randomly chosen as the labeled point (shown as colored dots in Fig.6-2(1)) for each test case. We perform ablation studies on this toy dataset to validate the effectiveness of individual components of the proposed ADP.

As shown in Fig.6-2, none of these methods but the proposed ADP could always achieve perfect label propagation. The base KNN graph produces the worst results, where both the outer circle and the three inner circles mix with the middle green circle, as shown in Fig.6-2(2). Label correction does not provide much help to the base KNN graph, because only one label per class is available, as shown in Fig.6-2(3). Adding affinity diffusion to the KNN graph successfully classified the outer circle, but it still fails to capture the relationship between the middle circle and inner circles without the help of label correction, as shown in Fig.6-2(4). The proposed ADP produces perfect results no matter where the initial labeled points are, which demonstrates the importance of each component to the success of the proposed ADP.

6.4.2 Comparison on Affinity Graph Construction

Graph is the essential component of GSSL. To demonstrate the capability of learning an optimal graph, ADP is compared to the following GSSL methods focused on graph construction: KNN, ℓ_1-graph [26], SPG [60], NNLRS [175], SSNNLRS [176], RDP [6], FAML [39], and SRD proposed in Chapter 5. Most of them use self-expressive models based on sparse representations [36] or low-rank representations [94] to build the graph. Following the same convention, the initial weight matrix $\mathbf{W}$ obtained by sparse representation [36] is used for all the diffusion based methods, including the proposed ADP, SRD, and RDP.

COIL20 and YaleB datasets are used for this experiment. COIL20 [103] contains 1,400

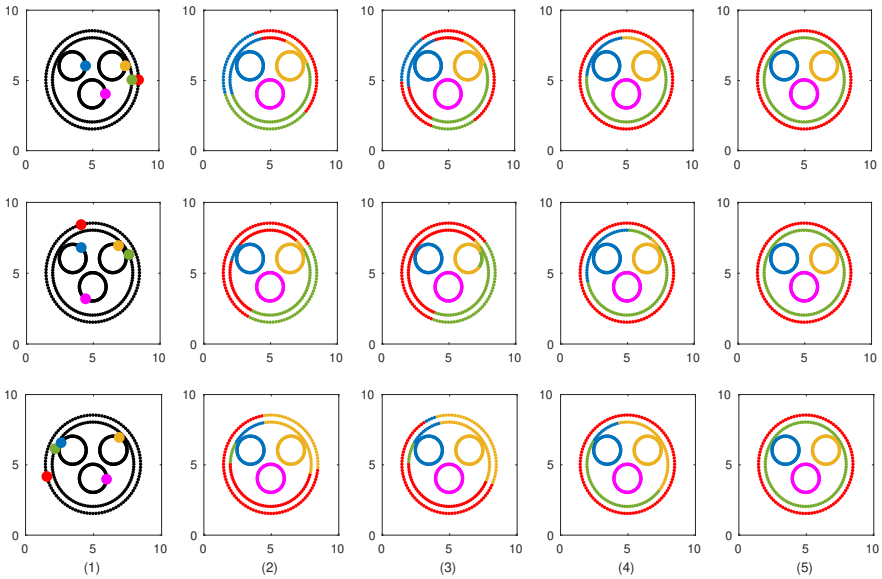

Figure 6-2: Visualization of the classification results on the toy data. (1) The toy dataset with one label per class initialized randomly. (2) The classification results of plain KNN graph + label propagation. (3) The results of KNN + label correction + label propagation. (4) The results of KNN + affinity diffusion + label propagation. (5) The results of the proposed ADP, which can be seen as KNN + label correction + affinity diffusion + label propagation.

gray-scale images of 20 objects with resolution of 128×128. Fig.6-3 shows some example images of this dataset. They are resized to 32×32, and the raw pixel values are used as features. YaleB [47] consists of 2,414 frontal face images of 38 subjects, with 64 images per subject acquired under various illumination conditions. Some example images are given in Fig.6-4. Following the same convention as in [39], the first 15 subjects are used. Each image is cropped, and down-sampled to 32×32, and the raw pixel is used as input. Different numbers of labels per class (δ) are tested.

Classification results are presented in Table 6.2. The proposed ADP and JDP consistently achieve significantly higher accuracies compared to all other graph construction methods. On the COIL20 dataset, both ADP and JDP produce almost perfect results with only three labeled points per class. Similarly, ADP improves 5% accuracy over the second-best method when $\delta = 1$ on the YaleB dataset. Given limited labeled points, ADP outperforms SRD proposed in Chapter 5 significantly, as SRD only makes use of given labels. The clear advantage of the proposed ADP comes from the fact that both the given labels and predicted labels are fully utilized to supervise the graph construction, resulting in a robust graph construction strategy less dependent on the initial labels. As it works slightly

better than its variant JDP, ADP is used in the subsequent experiments.

Figure 6-3: Some example images of the COIL20 dataset [103].

Figure 6-4: Some example images of the YaleB dataset [47].

6.4.3 Comparison on Label Propagation

Another indispensable component of GSSL is label propagation that spreads the labels to unlabeled points according to the learned graph. To validate the effectiveness of ADP on propagating labels, it is compared to several other label propagation methods, including GFHF [173], LGC [166], GGMC [146], TLLT [54], LPGMM [38], on the following datasets.

CMU-PIE [124] contains more than 40,000 facial images of 68 subjects. We use 20 frontal neutral images of all subjects. The images are cropped and resized to 32×32, and the raw pixels are used as features. Texture25 [84] includes 1,000 texture images from 25 classes. The images are pre-processed by the pre-trained VGG-net [125], resulting in a 4,096-dimensional feature vector per image. MPEG7 [83] contains 1,400 silhouette images from 70 classes. Sample images are shown in Fig.6-5. The IDSC shape descriptor [93] is used for distance calculation.

It can be seen from Fig.6-6 that the proposed ADP clearly outperforms other label

Table 6.2: Classification accuracy (%) of different graph construction methods with δ labeled points per class on COIL20 and YaleB datasets.

Dataset (δ)	KNN	ℓ_1-graph	SPG	NNLRS	SSNNLRS	RDP	SRD (ours)	FAML	JDP (ours)	ADP (ours)
COIL20 (1)	84.76±1.75	78.01±3.56	77.15±5.40	83.46±2.62	85.02±3.36	87.00±3.54	89.72±2.80	90.54±2.14	**98.54±1.14**	96.49±1.71
COIL20 (3)	91.86±1.23	87.07±2.32	88.62±4.80	91.20±1.89	92.51±2.11	95.00±1.32	96.12±1.52	94.20±2.18	99.04±0.37	**99.08±0.55**
COIL20 (5)	93.83±1.26	91.11±1.73	91.88±3.02	93.60±2.21	93.91±2.05	96.81±1.04	97.40±1.16	96.80±1.41	**99.72±0.25**	99.44±0.99
COIL20 (7)	94.89±1.38	93.50±1.03	93.56±2.50	94.16±2.00	94.73±1.94	98.55±0.37	99.01±0.49	97.35±1.07	**99.64±0.22**	99.55±0.42
COIL20 (9)	96.13±0.69	94.46±0.88	94.79±2.44	95.32±2.06	95.89±1.76	98.41±0.49	98.90±0.52	98.34±0.66	99.52±0.13	**99.58±0.37**
COIL20 (11)	96.67±0.44	96.00±0.92	95.93±1.81	95.50±1.75	96.21±1.58	98.84±0.46	99.23±0.47	98.49±0.48	99.71±0.09	**99.80±0.10**
Avg.	93.02±1.12	90.02±1.73	90.32±3.34	92.20±2.08	93.05±2.34	95.77±1.20	96.73±1.16	95.95±1.32	**99.36±0.37**	99.00±0.69
YaleB (1)	53.40±3.99	47.34±2.01	50.87±2.69	68.96±3.94	75.42±3.12	72.26±8.46	76.26±8.66	86.22±3.85	87.54±4.14	**91.42±3.89**
YaleB (5)	72.89±4.03	79.76±1.35	78.20±3.21	84.47±2.68	91.51±2.39	93.44±0.85	94.61±1.19	94.55±0.98	94.44±1.02	**96.10±0.74**
YaleB (9)	78.41±2.43	84.90±1.87	88.40±1.19	88.90±1.87	92.91±2.15	95.02±0.58	96.09±0.77	95.88±1.03	95.90±0.89	**97.37±0.55**
YaleB (13)	80.13±1.03	86.97±3.94	90.57±2.55	92.66±2.01	93.73±1.64	95.74±0.37	96.65±0.53	96.78±1.35	96.52±0.71	**97.44±0.43**
YaleB (17)	82.11±0.82	89.98±4.08	91.37±1.50	93.52±2.12	95.77±1.47	96.47±0.52	97.07±0.60	97.37±1.10	96.76±0.73	**97.61±0.45**
YaleB (21)	83.86±1.71	94.43±1.69	93.57±1.09	94.16±1.19	96.21±1.28	96.96±0.73	97.57±0.51	**98.54±0.60**	97.12±0.66	97.85±0.48
Avg.	75.13±2.34	80.56±2.46	82.16±2.09	87.11±2.30	90.93±2.01	91.65±1.92	93.04±2.04	94.89±1.48	94.71±1.35	**96.47±1.09**

Figure 6-5: Some example images of the MPEG7 dataset [83].

propagation methods, given different numbers of labeled points on all three datasets. Similar to the observation in Table 6.2, ADP produces a large winning margin when the labeled points are limited. GGMC has similar effects, but it does not gain much improvement when labeled points increase, as shown by [146] as well. All methods except ADP continuously propagate labels on the initial static graph. This is problematic as the initial graph is suboptimal in the sense that it does not reflect the global labels since it is constructed with few or even none label information. In contrast, ADP propagates labels on a dynamic graph, which is iteratively updated by the label information obtained in the previous iteration. It guarantees that the labels are propagated with respect to the local manifold structure and the global label information in every iteration.

6.4.4 Comparison on Inductive SSL

As shown previously, ADP outperforms state-of-the-art methods in the transductive setting. In this experiment, we are interested to see how it compares to other inductive SSL methods, including LapRLS [12], RELISH [53], LPDGL [51], DLSR [97] and GD [50]. Following the

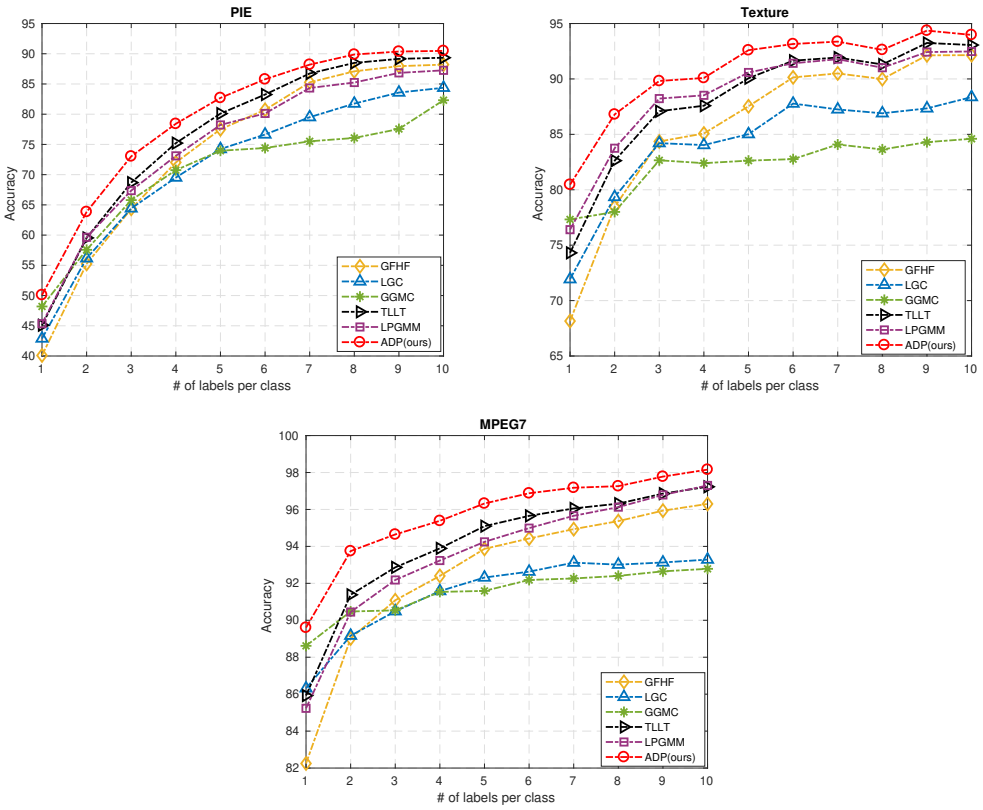

Figure 6-6: Classification accuracy (%) of different label propagation methods with δ labeled points per class on PIE, Texture, and MPEG7 datasets.

setup of [50], we conduct two experiments on the ORL and Wikipedia datasets.

ORL contains 400 face images of 40 subjects with various lighting conditions, facial expressions, and details. Each image is resized to the resolution of 32×32, and the raw pixel value is used as the feature. Wikipedia [111] dataset consists of 2,866 documents, including text and image, from 10 categories. Features are extracted by the bag-of-words encoder for the text and the SIFT descriptor for the image.

For both the ORL and Wikipedia datasets, 5-fold cross-validation is conducted for all the methods to evaluate their performances. In each test case, 20% of the data are randomly chosen to be the test set $\mathcal{T}$, and the remaining 80% serve as the training set $\mathcal{Z}$. Among the training set $\mathcal{Z}$, two samples per class in ORL are randomly picked to form the labeled training set $\mathcal{L}$ and the remaining samples are used as the unlabeled training set $\mathcal{U}$, so $\mathcal{Z} = \mathcal{L} \cup \mathcal{U}$. For the Wikipedia dataset, 20 samples per class are randomly selected to construct $\mathcal{L}$, and others are within set $\mathcal{U}$. In summary, there are 80, 80, 240 samples in $\mathcal{T}$, $\mathcal{L}$, $\mathcal{U}$ respectively for the ORL dataset, and 573, 200, 2093 samples respectively for the Wikipedia dataset, in each test case. Note that the proposed ADP uses the training set $\mathcal{Z}$ to construct the graph, then performs transductive label propagation from the labeled set $\mathcal{L}$ to the unlabeled set $\mathcal{U}$. Finally, inductive label propagation is performed from the training set $\mathcal{Z}$ to the testing set $\mathcal{T}$. The classification accuracy on sets $\mathcal{U}$ and $\mathcal{T}$ are reported as transductive accuracy and inductive accuracy, respectively. The data splits are consistent for all tested methods for a fair comparison. As shown in Table 6.3 and Table 6.4, the proposed ADP outperforms all the competitors in terms of both the transductive accuracy and the inductive accuracy.

Table 6.3: Inductive Semi-Supervised classification accuracy of compared Methods on ORL dataset.

Methods	Transductive Accuracy	Inductive Accuracy
LapRLS [12]	0.713 ± 0.032	0.640 ± 0.066
ReLISH [53]	0.738 ± 0.021	0.700 ± 0.057
LPDGL [51]	0.754 ± 0.019	0.710 ± 0.065
DLSR [97]	0.732 ± 0.012	0.736 ± 0.021
GD [50]	0.776 ± 0.016	0.755 ± 0.049
ADP (ours)	**0.781 ± 0.028**	**0.773 ± 0.040**

Table 6.4: Inductive Semi-Supervised classification accuracy of compared Methods on Wikipedia dataset.

Methods	Transductive Accuracy	Inductive Accuracy
LapRLS [12]	0.643 ± 0.010	0.613 ± 0.013
ReLISH [53]	0.607 ± 0.032	0.604 ± 0.030
LPDGL [51]	0.634 ± 0.018	0.630 ± 0.020
DLSR [97]	-	-
GD [50]	0.663 ± 0.006	0.658 ± 0.018
ADP (ours)	**0.669 ± 0.015**	**0.668 ± 0.020**

6.4.5 Hyper-parameter Robustness and Convergence Analysis

The hyper-parameters of the proposed ADP method are the trade-off parameters α in Eq.(6.10), β in Eq.(6.18), and the number of the nearest neighbor k. We fix β to 0.99 in the LGC label propagation step as suggested in [166], and evaluate the robustness of α and k here. Specifically, we test typical values 7 to 13 for k, and 0.1 to 0.99 for α. The results are shown in Fig.6-7 and it can be seen that the proposed ADP is not very sensitive to its hyper-parameters.

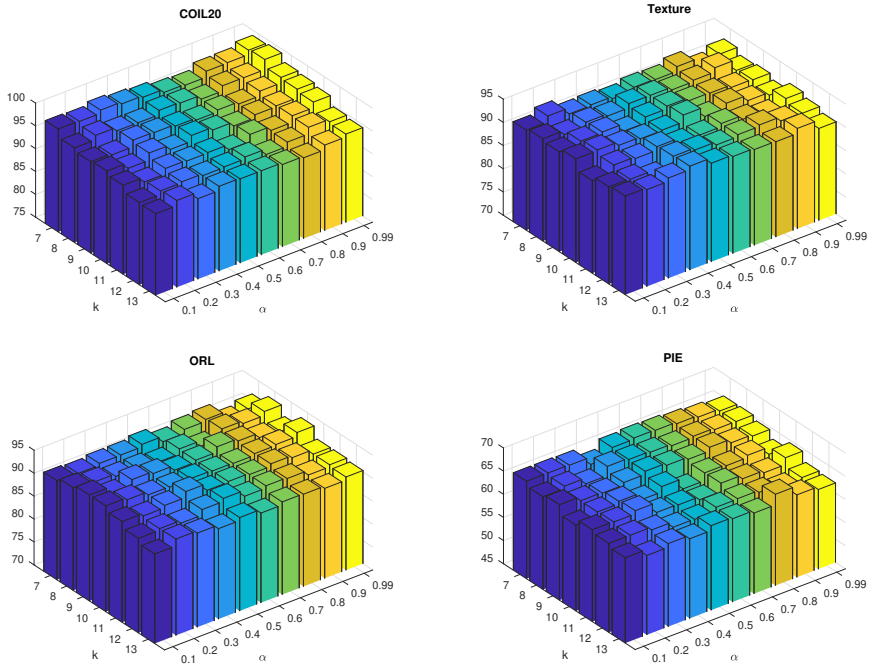

Figure 6-7: Robustness analysis of hyper-parameters on COIL20, Texture, ORL, and PIE datasets. The evaluated parameters are the neighbor size k and the regularization parameter α in Eq.(6.10).

It is also interesting to observe the convergence behavior of the proposed ADP and its variants JDP. As shown in Fig.6-8, both ADP and JDP converge fairly quickly, with ADP having faster convergence speed. It is not surprising, though, as more steps are executed in each iteration of ADP.

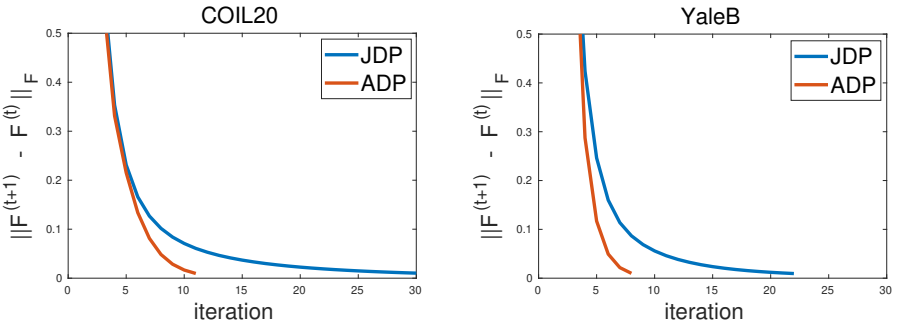

Figure 6-8: Convergence analysis of the proposed ADP and its variant JDP on COIL20 and YaleB datasets.

6.5 Summary

In this chapter, we integrated the two separated components of GSSL, which are graph construction and label propagation, into one unified framework. We show that learning a graph and a classification can both be formulated as a regularized function estimation problem, which attempts to find a trade-off between a global fitness to the labeled points and a local smoothness with respect to the intrinsic manifold structure revealed by the unlabeled points. An alternating diffusion process was proposed to efficiently solve the function estimation problems, allowing simultaneous update of the graph and unknown labels. It is a more plausible solution as we can now fully leverage the given labels and predicted labels to construct a local-and-global consistent graph, and also propagate labels in a more effective way due to the iteratively updated graph. Extensive experiments on both synthetic and real-world data have demonstrated the effectiveness of the proposed method.

So far, we have discussed affinity learning in both unsupervised and semi-supervised learning, where the main task is to obtain an affinity value given a pair of data points represented by a set of features. The problem could be totally ill-posed if the input feature space is not reliable, which is common for computer vision problems. In such scenarios, it is better to find a reasonable feature representation first before any other task is performed.

In the next chapter, we discuss how to learn representation for graphical data.

Chapter 7

Representation Learning with Graph Convolutional Networks

Affinity learning methods proposed in previous chapters work well if the input space is reliable. However, this is not always the case, especially for high-dimensional computer vision problems, in which the raw space is noisy. For such cases, it is better to learn a useful feature or representation first, as it is easier to perform downstream tasks, such as classification, in the specifically learned latent space. In this chapter, we focus on representation learning on graphs with graph convolutional networks.

7.1 Introduction

The success of machine learning methods heavily depends on having good data representation (or features) to which they are applied. Conventional approaches involve large human efforts to produce hand-crafted features, such as LBP, SIFT, and HOG. State-of-the-art representation learning methods are generally based on deep neural networks that learn layer-wise feature representation in an end-to-end fashion.

While deep learning techniques continue on making progress in various fields, researchers start to think about the limitation of these approaches. They find that classic deep learning techniques are designed for dealing with structured data, such as grids or sequences. For example, convolutional neural networks (CNNs), which is the state-of-the-art technique for all visual data related tasks, can only work with images/grids. These images are grid-style data in which the spatial locality and node ordering are fixed so that the convolutional

operation can be directly applied. Recurrent neural networks (RNNs), which is another popular architecture widely used in NLP, can only take texts/sequences as input. These state-of-the-art architectures are powerful as they take advantage of the structured input, but it also limits the application to other non-trivial data structures, such as graphs.

Graphs are a ubiquitous data structure, which is widely employed in computer science, biology, medical science, and many other fields. As a generic topological structure, graphs exist in various forms. Social networks, world wide web, citation networks, molecular structure, biological protein-protein networks are all readily modeled as graphs, including images and texts, as shown in Fig.7-1. In conventional machine learning, graphs are not only useful as a data structure, but also play a critical role in numerous learning models, including spectral graph theory [27], probabilistic graphical models [79], *etc.* Despite the huge potential of replicating the success of deep learning on graphical data, it does not draw much attention until recent years.

The concept of graph neural networks (GNNs) was first proposed in [120], where a basic neural network framework that works on graphical data was created. Instead of using fully connected neural networks, there is a growing interest in applying convolutional operations on graphs, due to the great success of CNN in computer vision. However, it is not straightforward to do so, as nodes on graphs have no obvious order or fixed neighboring structure so that a convolutional filter cannot be readily defined. It was attempted in [57] to implement a spectral convolution in the Fourier domain, where an eigenvector problem needs to be solved for the graph Laplacian matrix. This idea was later modified with a localized constraint that forces neighbors to be within k hops and constructed a CNN on graphs [31]. Researchers in [77] further extended the spectral graph convolutions with a first-order approximation, resulting in a clear layer-wise propagation formula that can be used to stack a multi-layer architecture similar to CNNs. They named their method as graph convolutional networks (GCNs) and demonstrated superior performances on semi-supervised classification. Since then, a large body of research has been presented in this subfield, including GraphSAGE [56], GAT [137], APPNP [78], to name a few.

These approaches generally focus on learning a discriminative representation so that classification can be easily achieved in the feature space. The essential idea of GCNs is to learn the representation of a target node by aggregating the representation of its neighboring nodes. In doing so, the topological graph information is encoded in the local neighborhood

structure and utilized in a convolutional way. Neighbor aggregation, linear transformation, and non-linear mapping are integrated as a single layer operation. A deep GCN can be constructed by stacking multiple layers, and it can be trained with backpropagation by calculating a loss with respect to a particular objective function. For example, in semi-supervised classification, the cross-entropy loss on labeled points are often used, while unlabeled points are only used for modeling the graph structure. The limitations are two folds: 1) overfitting. The model is not well trained and does not generalize well due to the lack of training data. This means we cannot use a deep architecture which is proven to be useful for learning good representations [59], as it makes the problem of overfitting even worse. This brings another problem: 2) the capacity of modeling long-range neighbor relationships. Existing GCNs usually contains two or three layers, which means they can only aggregate neighbor information as far as 2 or 3 hops away. This significantly limits the model capacity of capturing the global graph structure.

The contribution of this chapter is: we propose a novel loss function containing two terms, namely the fitness term and smoothness term. While the fitness term measures the difference between the prediction and ground-truth, the smoothness term regularizes the prediction to be smooth on the underlying manifold represented by the graph. By doing so, the semi-supervised classification can be viewed as a function estimation problem that favors a smooth function with fixed values at certain points. The key point here is that the smoothness term is an unsupervised loss that makes use of both labeled and unlabeled points, which prevents overfitting to the labeled points by the supervised fitness loss only. In addition, we show that the proposed objective function is equivalent to an iterative diffusion process that aggregates neighbor information similar to the layer-wise operation of GCN. Optimizing the objective function is somewhat like going through a GCN with potentially infinite layers, which increases the model capacity significantly. We demonstrate the effectiveness of the proposed loss with semi-supervised classification, where we show that the performance of the standard multi-layer perceptron (MLP) can be significantly boosted when used with the smoothness loss, and also it is ready to be plugged in any state-of-the-art GCNs to improve their performance.

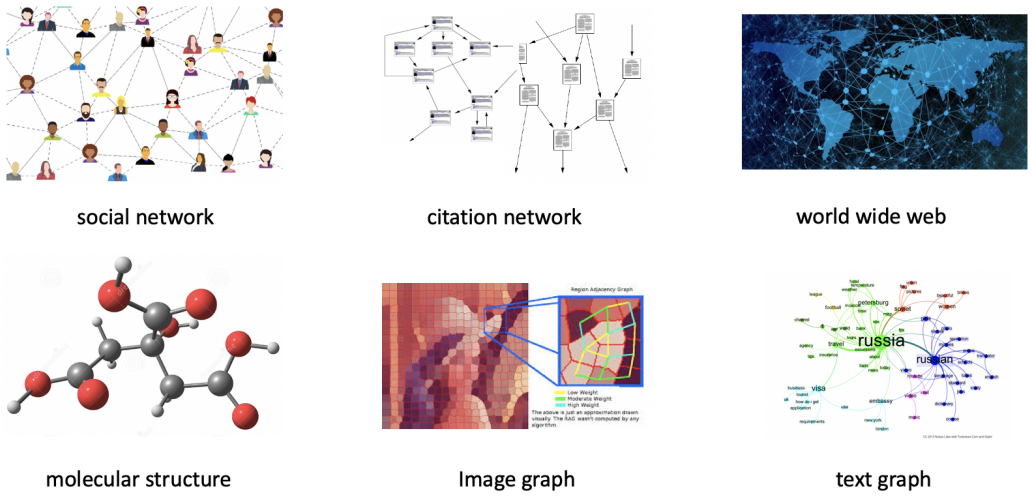

Figure 7-1: Examples of graph structure

7.2 Related Work

We start by reviewing several state-of-the-art GCNs and show that the main difference is on the way of aggregating information of neighbors. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph containing a node set $\mathcal{V}$, an edge set $\mathcal{E}$. Every node has an input feature representation $\mathbf{x}_v \in \mathbb{R}^{d_i}$ for $v \in \mathcal{V}$, and its hidden feature representation learned by l-th layer is denoted by $\mathbf{h}_v^{(l)} \in \mathbb{R}^{d_l}$. We may use $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ as the input representation to simplify derivations. Every node contains a neighbor set $\mathcal{N}(v) = \{u \in \mathcal{V} | (v, u) \in \mathcal{E}\}$ that indicates its adjacency nodes. It is common to add the node itself to its neighbor set in GCNs, and we denote it as $\widetilde{\mathcal{N}}(v) = \{u \in \mathcal{V} | (v, u) \in \mathcal{E}\} \cup \{v\}$.

A typical graph convolution layer can be decomposed into three components, namely neighbor aggregation, linear transformation, and non-linear mapping. In general, the l-th layer of a GCN model updates the hidden representation $\mathbf{h}_v^{(l)}$ as:

$$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)} \cdot \text{AGGREGATE}\left(\{\mathbf{h}_u^{(l-1)},\ \forall u \in \widetilde{\mathcal{N}}(v)\}\right)\right), \tag{7.1}$$

where AGGREGATE is an aggregation function defined by the specific model, $\mathbf{W}^{(l)}$ denotes a trainable weight matrix of l-th layer shared by all the nodes, and σ is a non-linear activation function, such as ReLU.

Graph Convolutional Networks (GCN) [77] is an instance of the general framework with an additional normalization trick, which can be written in the form:

$$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)} \sum_{u \in \widetilde{\mathcal{N}}(v)} \frac{\mathbf{h}_u^{(l-1)}}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(u)|}}\right), \tag{7.2}$$

where $|\mathcal{N}(v)|$ is the cardinality of the neighbor set $\mathcal{N}(v)$, *i.e.*, the number of neighbors of node v. GCN imposes a node-wise normalization in order to balance the effect of each node. The aggregation scheme of GCN can be viewed as a weighted average of all neighbors, and the weighting is proportional to the inverse of the number of neighbors. The update scheme of GCN can be efficiently implemented using sparse batch operations:

$$\mathbf{H}^{(l)} = \sigma\left(\mathbf{D}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}\right), \tag{7.3}$$

where $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is adjacency matrix with self-loop, $\mathbf{D}$ is the diagonal degree matrix with $D_{ii} = \sum_j \widetilde{A}_{ij}$. Indeed, $\mathbf{D}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}$ is the normalized graph Laplacian as discussed in Chapter 2.2.1, and actually GCN has a tight relation with spectral graph theory [77].

Graph Attention Networks (GAN) [137] can be seen as a variant of GCN with trainable aggregation weights. It utilizes the attention mechanism [4] to learn pairwise weights so as to select important neighbors. The update scheme of a single layer in GAN can be written as:

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in \widetilde{\mathcal{N}}(v)} \alpha_{vu}\mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)}\right), \tag{7.4}$$

where α_{vu} denotes the trainable attention weights between node v and u. In GAN, the attention mechanism is implemented as a single-layer feed-forward neural network that can be represented as follows:

$$\alpha_{vu} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_v||\mathbf{W}\mathbf{h}_u]\right)\right)}{\sum_{u \in \widetilde{\mathcal{N}}(v)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_v||\mathbf{W}\mathbf{h}_u]\right)\right)}, \tag{7.5}$$

where $\mathbf{a}$, $\mathbf{W}$ are the trainable weights and $||$ is the concatenation operation.

Approximate Personalized Propagation of Neural Predictions (APPNP) [78] decomposes neighbor aggregation and layer-wise propagation into two steps. It applies the aggregation as a post-processing step after representation learning, which aggregates the neighbor representation in the embedding space obtained from vanilla neural networks. Let

$\mathbf{H}$ represent the hidden feature learned by a standard neural network, APPNP aims at learning a context-aware representation $\mathbf{Z}$ as follows:

$$\mathbf{Z}^{(k)} = (1-\alpha)\widetilde{\mathbf{A}}\mathbf{Z}^{(k-1)} + \alpha\mathbf{H}, \tag{7.6}$$

where $\alpha \in [0, 1]$ controls trade-off between the self-representation and contextual neighbor information. This update scheme has a tight relationship with diffusion processes. In fact, it is derived from the personalized PageRank method as discussed in Chapter 2.6.

There are many other GCN models, such as GraphSAGE [56] which proposes the idea of neighbor sampling, JK-nets [150], which introduces residual connections to GCN. We refer readers to [149, 169] for comprehensive reviews.

7.3 Manifold Smoothness Loss

The central problem of GCNs is how to encode the graph structure, which is high-dimensional and non-Euclidean represented, into feature vectors so that it can be used for learning structure-aware representations. Existing GCNs generally make use of the graph structure in graph convolutional layers, where a graph-dependent neighbor aggregation scheme is integrated with layer-wise operations of vanilla neural networks. It results in a graph convolutional layers with an update formula as in Eq.(7.1), and a typical GCN is constructed by stacking multiple layers one by one, and eventually, the whole model is trained by backpropagation with a specific loss function of choice. We focus on the semi-supervised classification problem, as did by most GCNs, in which the cross-entropy loss $\mathcal{L}_{CE}$ is usually used to measure the difference between the predicted label distribution $\mathbf{z}_i$ and the ground-truth one-hot-encoded label indicator $\mathbf{y}_i$, *i.e.*, $y_ij = 1$ if the i-th data points belong to the j-th class and 0 otherwise. CE loss can be formulated as follows:

$$\mathcal{L}_{CE} = -\sum_{i \in L} \mathbf{y}_i \log \mathbf{z}_i, \tag{7.7}$$

where L is the set of labeled points. This is problematic as there are often limited labeled points in semi-supervised settings. GCNs trained by gradient descent with only the cross-entropy loss will quickly reach perfect training accuracy and zero gradients, resulting in overfitting to the training set of small size. The severe overfitting problem also prevents

GCN models from going deep, resulting in typical GCNs with two or three layers. Not like labeled data, unlabeled data are usually abundant. The problems mentioned above may both be addressed if a GCN can make use of these unlabeled data. We therefore propose a loss function that consists of two terms that explicitly take advantage of both labeled and unlabeled data. It can be formulated as follows:

$$\begin{aligned} &\mathcal{L} = \mathcal{L}_{fit} + \mu\, \mathcal{L}_{smooth}, \\ &\text{s.t.} \quad \mathcal{L}_{fit} = \sum_{i \in L} f(\mathbf{z}_i,\, \mathbf{y}_i), \quad \mathcal{L}_{smooth} = \sum_{i,j \in L \cup U} A_{ij}\, f(\mathbf{z}_i,\, \mathbf{z}_j), \end{aligned} \tag{7.8}$$

where $\mu > 0$ is a trade-off parameter, L and U are the sets of labeled points and unlabeled points respectively, A_{ij} is an element of the adjacency matrix $\mathbf{A}$ that represents whether x_i and x_j are neighbors or not, and f is a specific loss function of choice. While $\mathcal{L}_{fit}$ is a supervised loss that measures the fitness of the prediction $\mathbf{z}_i$ to the ground-truth $\mathbf{y}_i$ as other GCNs, the key innovation here is the unsupervised loss $\mathcal{L}_{smooth}$ that imposes a regularization on the model predictions. Minimizing the smoothness loss encourages the prediction function to be sufficiently smooth with respect to the graph structure, *i.e.*, neighbor nodes on the graph tend to have similar predictions. The overfitting problem of GCNs will be largely alleviated if they are trained with these two loss terms, and it also allows us to use a deeper architecture, as shown empirically in Table 7.2.

7.3.1 The Choice of $\mathcal{L}_{fit}$, $\mathcal{L}_{smooth}$ and their Justification

The loss functions consisting of two terms, namely $\mathcal{L}_{fit}$ and $\mathcal{L}_{smooth}$, which essentially measure the difference between vectors. There is no restriction on the choice of these two terms, and they can be any loss function, such as square error, cross-entropy loss, or KL-divergence, to name a few. We present two special choices here, in which $\mathcal{L}_{fit}$ and $\mathcal{L}_{smooth}$ are both squared loss or cross-entropy loss, and we show that they have a tight relation with the diffusion process and some existing GCNs.

$\mathcal{L}_{fit}$, $\mathcal{L}_{smooth}$ as L_2 Loss

Let the proposed loss $\mathcal{L}$ take the following form, where both $\mathcal{L}_{fit}$ and $\mathcal{L}_{smooth}$ are the squared loss:

$$\mathcal{L} = \sum_{i \in L} (\mathbf{z}_i - \mathbf{y}_i)^2 + \mu \sum_{i,j \in L \cup U} A_{ij} \, (\mathbf{z}_i - \mathbf{z}_j)^2 \, . \tag{7.9}$$

This is closely related to the graph regularization framework proposed in [166] and also used in Chapter 5 (refer Eq.(5.4)). Interestingly, if we treat the loss function Eq.(7.9) as an objective function with a variable $\mathbf{z}$, then we have a convex function whose minimal value can be obtained by setting the derivative with respect to $\mathbf{z}$ to 0 as follows:

$$\left. \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \right|_{\mathbf{z}=\mathbf{z}^*} = \mathbf{z}^* - \mathbf{y} + \mu \, (\mathbf{z}^* - \mathbf{A}\mathbf{z}^*) = 0, \tag{7.10}$$

which can be transformed into

$$\mathbf{z}^* - \frac{1}{1+\mu}\mathbf{y} - \frac{\mu}{1+\mu}\mathbf{A}\mathbf{z}^* = 0. \tag{7.11}$$

Let $\gamma = \frac{1}{1+\mu}$, and obviously $1 - \gamma = \frac{\mu}{1+\mu}$. The Eq.(7.11) can now be rewritten as:

$$\mathbf{z}^* - \gamma\mathbf{y} - (1-\gamma)\mathbf{A}\mathbf{z}^* = 0. \tag{7.12}$$

Putting variable $\mathbf{z}$ together, we have

$$(\mathbf{I} - (1-\gamma)\,\mathbf{A})\,\mathbf{z}^* = \gamma\mathbf{y}. \tag{7.13}$$

where $\mathbf{I}$ is the identity matrix of size n. By the definition of the adjacency matrix $\mathbf{A}$, $(\mathbf{I} - (1-\gamma)\,\mathbf{A})$ is invertible. We therefore have the closed-form solution for $\mathbf{z}$:

$$\mathbf{z}^* = \gamma\big(\mathbf{I} - (1-\gamma)\mathbf{A}\big)^{-1}\mathbf{y}. \tag{7.14}$$

Indeed, $\big(\mathbf{I} - (1-\gamma)\mathbf{A}\big)^{-1}$ is a diffusion kernel, and it was used by another GCN [72] in the layer-wise propagation for feature diffusion, except that they used $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ instead of $\mathbf{A}$. In this sense, the proposed loss function can be viewed as a label diffusion method that propagates labels from the labeled points to unlabeled points on the graph using a diffusion kernel, which is similar to the label propagation stage of the graph-based semi-supervised learning algorithms, *i.e.*, SRD and ADP proposed in Chapter 5 and 6, respectively.

Thanks to the connection between graph regularization framework and iterative diffusion

process [6, 166], it is easy to derive an iterative formula that converges to the same closed-form solution as in Eq.(7.14), and it can be written in matrix form (please refer Chapter 5 or 6 for the transformation between these optimization frameworks and iterative frameworks):

$$\mathbf{Z}^{(k)} = (1-\gamma)\mathbf{A}\mathbf{Z}^{(k-1)} + \gamma\mathbf{Y}, \tag{7.15}$$

where $\mathbf{Z}^{(0)} = \mathbf{Y}$, though the convergence does not depend on the initial condition. This actually brings us to the update scheme of one of the state-of-the-art GCNs, APPNP [78]. The only difference is that they use this iterative process to aggregate the neighbor's feature representation, while we directly aggregate neighbor's prediction. From this iterative formula, the proposed loss can be interpreted as aggregating neighbors' prediction (the first term), with a certain chance of re-initialization (the second term). Optimizing the loss function is equivalent to run the iterative process infinity times until convergence, which significantly improves the model capacity on capturing long-range neighbor relationships. The existence of the re-initialization is also useful for preventing the over-smoothing effect [89]. This effect widely exists in existing GCNs as they aggregate neighbor representations layer by layer. Without any mechanism to keep the distinction, it is very likely for a GCN model to have a similar representation after training, which hinders the performance of downstream discriminative analysis, such as classification.

$\mathcal{L}_{fit}$, $\mathcal{L}_{smooth}$ as Cross-Entropy Loss

Another possible choice is the cross-entropy loss, which is proven to be effective for classification. Instead of considering the total loss on all classes, the cross-entropy loss concentrates on the loss of the true class and imposes a large penalty when the predicted probability of belonging to the true class is low. In the proposed loss function Eq.(7.8), it is straightforward to use cross-entropy loss for $\mathcal{L}_{fit}$. However, directly applying it for $\mathcal{L}_{smooth}$ is problematic as neither predictions $\mathbf{z}_i$ and $\mathbf{z}_j$ is a one-hot-encoded vector. Cross-entropy on such vectors will lose the focus on the true class and lead to poor classification performance. Therefore, we convert one of the predictions to a one-hot-encoded vector before we apply the cross-entropy loss. The conversion can be represented by the following function:

$$\phi(\mathbf{z}_{ij}) = \begin{cases} 1, & \text{if } \mathbf{z}_{ij} = \max(\mathbf{z}_i), \\ 0, & \text{otherwise,} \end{cases} \tag{7.16}$$

where ϕ is an element-wise function. Putting $\mathcal{L}_{fit}$ and $\mathcal{L}_{smooth}$ together, we can now write the proposed loss function as:

$$\mathcal{L} = -\sum_{i \in L} \mathbf{y}_i \log \mathbf{z}_i - \mu \sum_{i \in L \cup U} A_{ij}\, \phi(\mathbf{z}_i) \log \mathbf{z}_j. \tag{7.17}$$

The first term is the supervised cross-entropy loss for the labeled points used by most of the existing GCNs, while the second term is an unsupervised cross-entropy loss that works on both the labeled and unlabeled points. The unsupervised cross-entropy loss enforces the neighbor's predictions to have the maximal value for the same class, *i.e.*, it enforces neighbors to have the same label without knowing what the label is. Note that the loss on the labeled points has been measured in both terms, but they are not redundant as one measures the fitness to the ground-truth, and another regularizes the label smoothness with neighbors on the graph.

7.4 Experiments

To demonstrate the effectiveness of the proposed smoothness loss, we conduct various controlled experiments on both MLP and state-of-the-art GCNs. They are evaluated with and without the smoothness loss, as our concern is on the relative difference rather than the absolute performance. Following most GCNs, we focus on the task of semi-supervised classification on citation networks, and we show that the proposed smoothness loss can significantly improve vanilla MLP as well as bring performance gain for all the state-of-the-art GCNs.

7.4.1 Experimental Settings

We evaluated the proposed loss on MLP, and several popular GCNs, including GCN [77], GAT [137], and APPNP [78]. As the proposed smoothness loss can essentially be viewed as a regularization, we name the corresponding methods used with the proposed loss as R-MLP, R-GCN, R-GAT, R-APPNP for the sake of simplicity and easy comparison. For MLP, we

use the vanilla MLP without any bells and whistles, and each layer is a fully-connected layer with 64 hidden units. For GCNs, we use the same set of hyper-parameters before and after adding the smoothness loss for fair comparisons. Specifically, we use all GCNs with two hidden layers, 64 hidden units, ReLU activation, dropout layers before each hidden layer at rate 0.5, ℓ_2 regularization at $5e^{-4}$, and learning rate 0.01. For the smoothness loss, we use the modified cross-entropy loss as it is better for classification tasks. All the weights are initialized by the method described in [48], and we train all GCNs for a maximum of 1000 epochs using Adam [75] and early stopping with a window size 100, *i.e.*, we stop training if the validation loss does not decrease for 100 consecutive epochs.

Experiments are conducted on the widely used citation datasets, including Cora, Citeseer, and Pubmed [122]. Information of interest is summarized in Table 7.1. These datasets consist of documents and citation links between them. Documents are represented by sparse bag-of-words feature vectors, and the citation links are treated as undirected edges used for constructing the binary adjacency matrix $\mathbf{A}$. For all the citation datasets, we randomly split each of them 50 times, and the average performance is presented. Each split contains ℓ nodes per class, 500 validation nodes, and 1000 testing nodes, where ℓ is tested for 5, 10, or 20 in our experiments.

Table 7.1: Summary of datasets used in the experiments.

Dataset	**Nodes**	**Edges**	**Classes**	**Features**	**Validation Nodes**	**Testing Nodes**
Cora	2,708	5,429	7	1,433	500	1,000
Citeseer	3,327	4,732	6	3,703	500	1,000
Pubmed	19,717	44,338	3	500	500	1,000

7.4.2 Qualitative Comparison of Embedding Visualization

The core idea of GCNs is to make use of graph structure to learn a good representation that can be used for downstream machine learning tasks, such as classification. To this end, existing GCNs usually integrate an aggregation scheme to the layer-wise operations of neural networks so as to explicitly learn context-aware representation. We propose to regularize the loss function with a graph smoothness loss defined on both labeled points and unlabeled points. Although node representations are not specifically changed, GCNs with the proposed loss tend to produce more compact representations that are better for discriminative analysis. As we explicitly regularize neighbor node's predictions to be close,

there is an implicit effect to enforce the neighbor node's representations become closer as well, which results in compact classes that are ready to be classified.

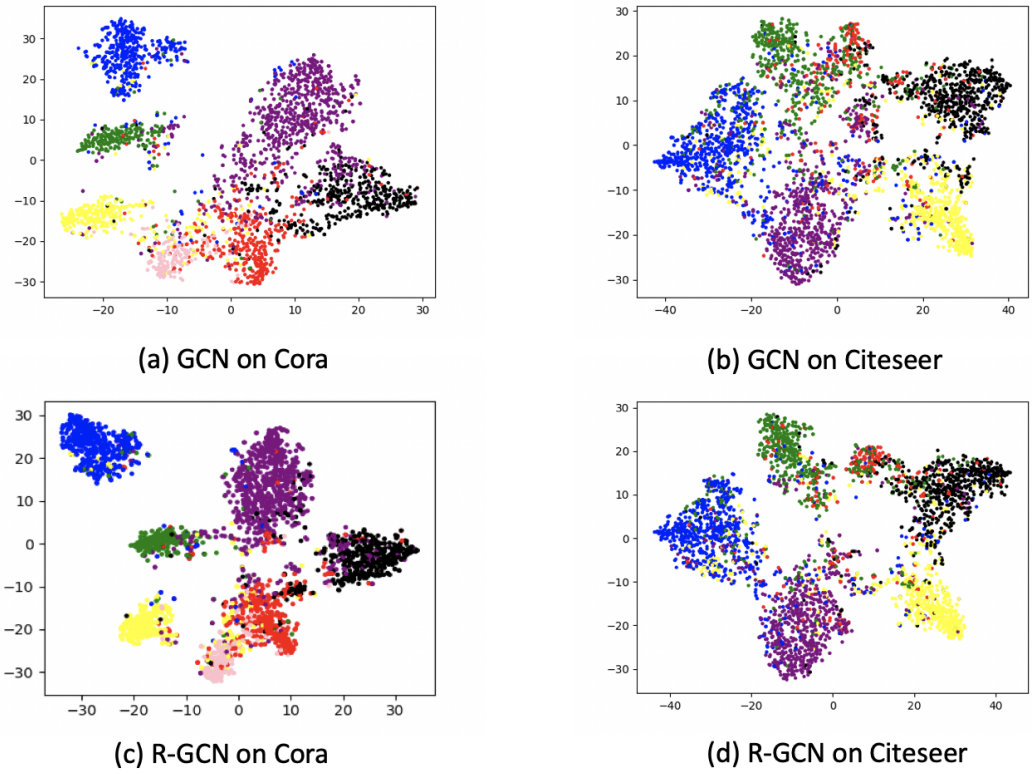

Figure 7-2: T-sne plots of the learned representations of GCN and R-GCN on Cora and Citeseer datasets.

We train a two-layer GCN model with and without the regularization loss, denoted by R-GCN and GCN, respectively, on the Cora and Citeseer datasets. We extract the output of the last hidden layer as the learned representation (embedding) and visualize it in two-dimensional space using t-sne [99]. As shown in Fig.7-2, the representations obtained by R-GCN are clearly more compact than those of GCN, which demonstrates the effectiveness of the proposed smoothness loss on learning useful representation. This further explains why it can improve classification accuracy.

7.4.3 Quantitative Comparison of Classification Accuracy

Baseline Multi-Layer Perceptrons

Next, we conduct quantitative comparisons between models and their regularized counterparts on semi-supervised classification. We hypothesized that limited labeled points could

not train a deep model, and proposed an unsupervised loss to address it. To demonstrate this, we first compare the vanilla MLP with R-MLP using a different number of hidden layers. The classification accuracy on the Cora, Citeseer, Pubmed datasets are reported.

As shown in Table 7.2, the performance of vanilla MLP is significantly improved by adding the proposed smoothness loss, which also shows the importance of making use of graph structure when it is available. The performance of R-MLP also demonstrates the potential capability of the smoothness loss on training deep architectures. While the performance of vanilla MLP reaches its optimal with 1 or 2 layers and drops dramatically when the number of layers increases further, adding the smoothness loss makes its performance much more stable even with five hidden layers.

Table 7.2: Semi-Supervised Classification Accuracy ± Standard Deviation (%) of MLPs.

Datasets	**# Layers**	1	2	3	4	5
Cora	MLP	57.5 ± 2.4	57.7 ± 2.0	54.3 ± 3.5	33.2 ± 8.2	26.6 ± 7.3
	R-MLP	**58.6 ± 2.1**	**76.0 ± 2.3**	**77.4 ± 3.1**	**77.6± 4.2**	**73.3± 4.7**
Citeseer	MLP	58.8 ± 2.3	57.0 ± 2.0	52.1 ± 2.9	36.5 ± 7.9	26.5 ± 5.6
	R-MLP	**60.6 ± 2.1**	**62.5 ± 2.3**	**62.7 ± 2.5**	**57.4 ± 2.8**	**52.2 ± 4.8**
Pubmed	MLP	69.9 ± 2.7	69.6 ± 2.5	68.0 ± 2.5	65.9 ± 5.3	56.7 ± 7.9
	R-MLP	**70.6 ± 2.4**	**70.7± 2.7**	**69.3± 3.2**	**66.5± 5.1**	**61.1± 6.5**

State-of-the-art Graph Convolutional Netowrks

The proposed smoothness is ready to be plugged into any state-of-the-art GCN models to improve its performance. We compare the classification performance of GCN, GAT, APPNP to their regularized version, namely R-GCN, R-GAT, R-APPNP, in the semi-supervised setting. In addition, to study the effect of the number of labeled points on the model accuracy, we use a different number of labeled points per class, as denoted by ℓ. ℓ is tested for 20, 10, 5, and the respective classification accuracies are reported.

As shown in Table 7.3, adding the smoothness loss improves the classification accuracy of all GCN models consistently, including GCN, GAT, and APPNP. This may seem counter-intuitive, as the graph structure used by these GCNs implies layer-wise propagations. Using label propagation again in the loss function seems redundant. We believe that graph information has not been extensively exploited by existing GCNs due to their inability to capturing long-range neighbor relationships. Coupling neighbor aggregation and the

Table 7.3: Semi-Supervised Classification Accuracy ± Standard Deviation (%) of GCNs.

Datasets	ℓ	GCN	R-GCN	GAT	R-GAT	APPNP	R-APPNP
Cora	20	79.2 ± 1.7	**81.9 ± 1.8**	80.8 ± 1.6	**81.9 ± 1.3**	82.7 ± 1.3	**83.4 ± 1.4**
	10	75.8 ± 2.2	**78.6 ± 2.2**	77.6 ± 2.2	**78.2 ± 2.1**	80.1 ± 1.5	**81.2 ± 2.1**
	5	68.1 ± 4.1	**71.9 ± 4.2**	71.8 ± 3.4	**72.3 ± 3.5**	75.1 ± 3.5	**77.2 ± 2.9**
Citeseer	20	68.3 ± 2.0	**70.9 ± 2.3**	68.5 ± 1.6	**69.4 ± 1.4**	69.9 ± 1.6	**71.4 ± 1.5**
	10	64.8 ± 2.6	**67.3 ± 2.7**	66.2 ± 2.5	**67.1 ± 2.2**	67.0 ± 2.7	**68.6 ± 2.6**
	5	57.4 ± 4.2	**58.7 ± 3.6**	59.4 ± 4.6	**61.4 ± 4.3**	61.6 ± 3.8	**62.8 ± 3.6**
Pubmed	20	77.5 ± 2.5	**78.6 ± 2.5**	77.5 ± 2.5	**78.3 ± 2.0**	79.1 ± 2.5	**80.9 ± 2.2**
	10	73.8 ± 3.6	**75.1 ± 3.4**	73.7 ± 4.1	**75.3 ± 2.5**	76.2 ± 3.4	**77.8 ± 2.9**
	5	68.9 ± 4.5	**69.7 ± 4.5**	67.8 ± 5.0	**70.3 ± 4.8**	72.2 ± 5.5	**73.4 ± 4.5**

layer-wise operation takes existing GCNs to a dilemma between strict localization (shadow model) and over-smoothing (deep model), while the proposed smoothness loss can model global graph structure without going deep.

Performances of using different values of ℓ demonstrate that the proposed smoothness loss is robust with respect to the number of labeled points, as its improvement is recorded even with only five labeled points per class.

7.5 Summary

In this chapter, we discuss the representation learning on graphs with graph convolutional networks (GCNs). While most existing GCNs focus on encoding graph structure into layer-wise neural operations by a neighbor aggregation scheme, we propose a simple yet effective manifold smoothness loss to regularize these GCNs. It addresses two problems of existing GCNs, which are overfitting to few labeled points and incapacity of capturing long-range neighbor relationships. We justify the proposed loss by drawing its connection to the iterative diffusion process and show that minimizing the proposed loss can be viewed as aggregating neighbor predictions with infinity layers. The overfitting problem is alleviated as the smoothness loss is unsupervised so that we can take advantage of the unlabeled points. The long-range neighbor relationships are captured without adding more layers explicitly, which reduces the risk of over-smoothing.

We conduct qualitative comparisons to show that GCNs trained with the smoothness loss learn more compact representation than their vanilla counterparts. Quantitative analyses demonstrate that such representations can yield better performances on downstream

discriminative machine learning tasks, such as semi-supervised classification.

Chapter 8

Conclusion and Future works

In this thesis, we aim at designing machine learning algorithms to analyze data, identify patterns, and make predictions. This is achieved by learning accurate pairwise affinities between data points with the node-edge graph. To go beyond the pairwise relationship, we make use of the diffusion process to propagate neighboring affinities according to the local graph structure, resulting in context-aware affinity that is smooth with respect to the local data manifold. Non-local affinity is also studied by incorporating the global label information into the diffusion process. Learning affinities and class labels simultaneously leads to the local-global consistent affinity and improved classification accuracy. Several affinity learning algorithms are proposed and applied to various machine learning problems, such as clustering, dimensionality reduction, and semi-supervised classification. When the given input space of data is not reliable, we utilize data and task driven approach to learn latent feature representations. Graph convolutional networks based on a smoothness loss are proposed to learn representation from both labeled and unlabeled data in a semi-supervised setting. Extensive experimental results demonstrate the effectiveness of these proposed algorithms.

The main contributions of the thesis are summarized as follows:

- Unsupervised learning of context-aware affinity. In Chapter 3, we discuss the important role of affinity learning in the unsupervised setting. Due to the lack of label information, the mapping from data points to labels cannot be established, but the pairwise relationships between data points can be utilized to identify a potential pattern of interest. We propose a Diffusion-based Sparse Subspace Clustering (DSSC)

algorithm, in which the key idea is to construct an affinity graph by sparse coding and post-process the affinity graph by a diffusion process. The diffusion process propagates the initial affinity values with respect to the graph structure and re-evaluates these values in an iterative manner. It results in accurate affinity values with contextual information. We demonstrated superior performances on various visual tasks, including handwritten digits recognition, facial image clustering, and motion segmentation.

- Solutions to solve large-scale spectral problems. Spectral problems referred to those problems that involve the spectral eigendecomposition. Many spectral-based approaches, including DSSC and several non-linear dimensionality reduction techniques, are scale limited due to the cubic computational complexity of eigendecomposition. In Chapter 4, the classic Nyström approach is rephrased using the idea of Markov random walk, and a generic diffusion framework is proposed that can generalize most existing Nyström-like solutions. Within this framework, a Diffusion-based Variational Nyström (DVN) algorithm is then proposed, and it approximately solves the general spectral problem using higher-order Markov transition matrices. We evaluate DVN on spectral clustering and Laplacian Eigenmaps for dimensionality reduction, and show that the proposed DVN can achieve the best trade-off between accuracy and efficiency, compared to existing approximate approaches.

- Semi-supervised learning of non-local affinity. In Chapter 5, we discuss the importance of affinity learning in the graph-based semi-supervised learning (GSSL), where the goal is to propagate labels from labeled points to unlabeled ones. Unsupervised affinity learning approaches usually use neighborhood information, resulting in affinities that are only aware of the local structure. We propose a variant of the diffusion process called Self-Reinforced Diffusion (SRD) to learn non-local affinity in GSSL, where the label information is incorporated into the diffusion process as a global prior. Self-similarity is also proposed to encode data density distribution to guide the diffusion process. Extensive experiments are conducted to show the benefits of learning non-local affinity with label information and validate the choice of self-affinity. While in Chapter 6, we propose an extension for the work in Chapter 5, where we integrated the two separate stages, affinity learning, and label propagation, into one unified framework by feeding the predicted labels back to the affinity learning process.

This is achieved by formulating both of them as similar function estimation problems that can be characterized by the graph regularization framework. By establishing the connection between the regularization framework and the diffusion process, we then propose another variant of the diffusion process, named Alternating Diffusion Process (ADP), to alternatively or jointly propagate the affinities and labels. It enables us to learn affinity graph and class labels simultaneously so that the intermediate results of classification can be feedback to affinity learning iteratively, resulting in an optimal affinity graph that can be used for accurate label propagation. We conduct semi-supervised classification on various visual tasks and demonstrate the proposed ADP could achieve even better results than SRD.

- Representation learning with graph convolutional networks (GCNs). In Chapter 7, we study the application of GCNs in semi-supervised learning, where we identify two limitations of existing GCNs, which are overfitting and the inability to capture long-range neighbor relationships. We propose an unsupervised manifold smoothness loss that takes advantage of both labeled points and unlabeled points to model data manifold so that the overfitting problem is alleviated. By connecting this loss function to the graph regularization framework and the diffusion process, we observe that minimizing the proposed loss can be viewed as propagating contextual information using a GCN with a potentially infinite number of layers. It significantly improves the capacity of modeling long-range neighbor structures without explicitly constructing a very deep architecture. We conduct extensive experiments on citation network datasets, and it is shown that the proposed smoothness loss can lead to a more compact representation of graph nodes, as well as consistent performance gain compared to state-of-the-art GCN models.

8.1 Future Works

Despite the satisfactory performances achieved by all of the proposed algorithms, there are still possible extensions and unaddressed problems that can be studied for future works:

- As shown by the DSSC algorithm proposed in Chapter 3, the diffusion process with graphs obtained by sparse representation models produces significantly better performances compared to those using the Gaussian kernel graph. We hypothesize that the

key is the sparsity nature of ℓ_1 graph, where we have to manually enforce kNN sparsity to constrain a local diffusion process in the Gaussian kernel graph. It would be interesting to investigate the relationships between sparse representation models and the diffusion process, *e.g.*, replace the ℓ_1 minimization to the elastic net regularization.

- Semi-supervised learning (SSL) if of great interest in both theoretical research and real application. In this thesis, we have proposed two frameworks for learning from both labeled points and unlabeled points, namely SRD and ADP. They both show much better performance than their unsupervised counterparts. However, there are still important questions that remain totally open: do unlabeled points always help? To what extent can they help? How can we make use of unlabeled data so that they can guarantee to improve model accuracy? If these questions can be answered, SSL will become a much powerful tool.

- Representation learning is the key to solve very complicated visual tasks, as it is sometimes very difficult or impossible to learn accurate similarity in the raw input space. In this thesis, we studied the representation learning on graphs with graph convolutional networks (GCNs). We advocate this line of research as we think the success of deep learning should not be limited to the structured data, such as images and sequences, but to more complex data structures, such as graphs. We conducted research of GCNs on small citation networks. Similar researches could be done to scale current GCNs to large-scale graphical data, such as knowledge graphs, social networks, which is of huge significance for real-world applications.

Bibliography

[1] D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003.

[2] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[3] K. E. Atkinson. *The numerical solution of integral equations of the second kind*, volume 4. Cambridge university press, 1997.

[4] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] S. Bai, X. Bai, and Q. Tian. Scalable person re-identification on supervised smoothed manifold. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2530–2539, 2017.

[6] S. Bai, X. Bai, Q. Tian, and L. J. Latecki. Regularized diffusion process for visual retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3967–3973, 2017.

[7] S. Bai, X. Bai, Q. Tian, and L. J. Latecki. Regularized diffusion process on bidirectional context for object retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(5):1213–1226, 2019.

[8] P. Baldi and G. W. Hatfield. *DNA microarrays and gene expression: from experiments to data analysis and modeling*. Cambridge university press, 2011.

[9] R. Basri and D. W. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):218–233, 2003.

[10] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[11] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[12] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(11):2399–2434, 2006.

[13] R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[14] S. Belongie, C. Fowlkes, F. Chung, and J. Malik. Spectral partitioning with indefinite kernels using the nyström extension. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 531–542, 2002.

[15] Y. Bengio, O. Delalleau, and N. Le Roux. Label propagation and quadratic criterion. In O. Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-supervised learning*, chapter 11, pages 193–216. MIT Press, 2004.

[16] S. K. Bhatia and J. S. Deogun. Conceptual clustering in information retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28(3):427–436, 1998.

[17] T. D. Bie and N. Cristianini. Convex methods for transduction. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 73–80, 2004.

[18] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. 2001.

[19] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Annual Conference on Computational Learning Theory*, pages 92–100. ACM, 1998.

[20] V. Castelli and T. M. Cover. The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *IEEE Transactions on Information Theory*, 42(6):2102–2117, 1996.

[21] O. Chapelle, V. Sindhwani, and S. S. Keerthi. Branch and bound for semi-supervised support vector machines. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 217–224, 2007.

[22] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 2005, pages 57–64, 2005.

[23] G. Chen and G. Lerman. Spectral curvature clustering (SCC). *International Journal of Computer Vision*, 81(3):317–330, 2009.

[24] M. Chen, K. Q. Weinberger, and J. Blitzer. Co-training for domain adaptation. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 2456–2464, 2011.

[25] X. Chen and D. Cai. Large scale spectral clustering with landmark-based representation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 313–318, 2011.

[26] B. Cheng, J. Yang, S. Yan, Y. Fu, and T. S. Huang. Learning with l1-graph for image analysis. *IEEE Transactions on Image Processing*, 19(4):858–866, 2009.

[27] F. R. Chung and F. C. Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[28] C. Cortes, M. Mohri, and A. Talwalkar. On the impact of kernel approximation on learning accuracy. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 113–120, 2010.

[29] J. P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998.

[30] D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977.

[31] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 3844–3852, 2016.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[33] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[34] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 551–556, 2004.

[35] M. Donoser and H. Bischof. Diffusion processes for retrieval revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1320–1327, 2013.

[36] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013.

[37] V. Estivill-Castro. Why so many clustering algorithms: a position paper. *SIGKDD explorations*, 4(1):65–75, 2002.

[38] M. Fan, X. Zhang, L. Du, L. Chen, and D. Tao. Semi-supervised learning through label propagation on geodesics. *IEEE Transactions on Cybernetics*, 48(5):1486–1499, 2018.

[39] X. Fang, N. Han, W. K. Wong, S. Teng, J. Wu, S. Xie, and X. Li. Flexible affinity matrix learning for unsupervised and semisupervised classification. *IEEE Transactions on Neural Networks and Learning Systems*, 30(4):1133–1149, 2019.

[40] P. Favaro, R. Vidal, and A. Ravichandran. A closed form solution to robust subspace estimation and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1801–1807, 2011.

[41] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1):59–70, 2007.

[42] J. Feng, Z. Lin, H. Xu, and S. Yan. Robust subspace segmentation with block-diagonal prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3818–3825, 2014.

[43] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190, 2008.

[44] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

[45] P. Fox-Roberts and E. Rosten. Unbiased generative semi-supervised learning. *Journal of Machine Learning Research*, 15(1):367–443, 2014.

[46] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

[47] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

[48] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256, 2010.

[49] A. Goh and R. Vidal. Segmenting motions of different types by unsupervised manifold clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–6, 2007.

[50] C. Gong, X. Chang, M. Fang, and J. Yang. Teaching semi-supervised classifier via generalized distillation. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 2156–2162, 2018.

[51] C. Gong, T. Liu, D. Tao, K. Fu, E. Tu, and J. Yang. Deformed graph laplacian for semisupervised learning. *IEEE Transactions on Neural Networks and Learning Systems*, 26(10):2261–2274, 2015.

[52] C. Gong, D. Tao, X. Chang, and J. Yang. Ensemble teaching for hybrid label propagation. *IEEE Transactions on Cybernetics*, 49(2):388–402, 2019.

[53] C. Gong, D. Tao, K. Fu, and J. Yang. Relish: Reliable label inference via smoothness hypothesis. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

[54] C. Gong, D. Tao, W. Liu, L. Liu, and J. Yang. Label propagation via teaching-to-learn and learning-to-teach. *IEEE Transactions on Neural Networks and Learning Systems*, 28(6):1452–1465, 2017.

[55] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[56] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 1024–1034, 2017.

[57] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[58] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[59] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[60] R. He, W.-S. Zheng, B.-G. Hu, and X.-W. Kong. Nonnegative sparse coding for discriminative semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2849–2856, 2011.

[61] X. He and P. Niyogi. Locality preserving projections. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 153–160, 2004.

[62] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

[63] G. Hinton and S. Roweis. Stochastic neighbor embedding. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 833–840, 2002.

[64] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–11, 2003.

[65] W. Hong, J. Wright, K. Huang, and Y. Ma. Multiscale hybrid linear models for lossy image representation. *IEEE Transactions on Image Processing*, 15(12):3655–3671, 2006.

[66] J. Hu, B. K. Ray, and M. Singh. Statistical methods for automated generation of service engagement staffing plans. *IBM Journal of Research and Development*, 51(3.4):281–293, 2007.

[67] B. Huang and T. Jebara. Loopy belief propagation for bipartite maximum weight b-matching. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 195–202, 2007.

[68] Y. Huang, Q. Liu, and D. Metaxas. Video object segmentation by hypergraph cut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1738–1745, 2009.

[69] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

[70] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015.

[71] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[72] B. Jiang, D. Lin, J. Tang, and B. Luo. Data representation and learning with graph diffusion-embedding networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10414–10423, 2019.

[73] T. Joachims. Transductive inference for text classification using support vector machines. In *Icml*, volume 99, pages 200–209, 1999.

[74] Z. Kalal, J. Matas, and K. Mikolajczyk. Pn learning: Bootstrapping binary classifiers by structural constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 49–56, 2010.

[75] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[76] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 3581–3589, 2014.

[77] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representation (ICLR)*, 2017.

[78] J. Klicpera, A. Bojchevski, and S. Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[79] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[80] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 1097–1105, 2012.

[81] A. Kumar and H. Daumé. A co-training approach for multi-view spectral clustering. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 393–400, 2011.

[82] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 331–339, 1995.

[83] L. J. Latecki, R. Lakamper, and T. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 424–429, 2000.

[84] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, 2005.

[85] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[86] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[87] C.-G. Li, Z. Lin, H. Zhang, and J. Guo. Learning semi-supervised representation towards a unified optimization framework for semi-supervised learning. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2767–2775, 2015.

[88] C.-G. Li, C. You, and R. Vidal. Structured sparse subspace clustering: A joint affinity learning and subspace clustering framework. *IEEE Transactions on Image Processing*, 26(6):2988–3001, 2017.

[89] Q. Li, Z. Han, and X.-M. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[90] Q. Li, W. Liu, and L. Li. Self-reinforced diffusion for graph-based semi-supervised learning. *Pattern Recognition Letters*, 125:439–445, 2019.

[91] Q. Li, Y. Ren, L. Li, and W. Liu. Fuzzy based affinity learning for spectral clustering. *Pattern Recognition*, 60:531–542, 2016.

[92] Y. Lin and Y. Jeon. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590, 2006.

[93] H. Ling and D. W. Jacobs. Shape classification using the inner-distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):286–299, 2007.

[94] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2012.

[95] C.-Y. Lu, H. Min, Z.-Q. Zhao, L. Zhu, D.-S. Huang, and S. Yan. Robust and efficient subspace segmentation via least squares regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 347–360, 2012.

[96] L. Lu and R. Vidal. Combined central and subspace clustering for computer vision applications. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 593–600, 2006.

[97] M. Luo, L. Zhang, F. Nie, X. Chang, B. Qian, and Q. Zheng. Adaptive semi-supervised learning with discriminative least squares regression. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 2421–2427, 2017.

[98] Y. Ma, H. Derksen, W. Hong, and J. Wright. Segmentation of multivariate mixed data via lossy data coding and compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1546–1562, 2007.

[99] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[100] J. MacQueen. Some methods for classification and analysis of multivariate observations, 1967.

[101] M. Meila and J. Shi. A random walks view of spectral segmentation. 2001.

[102] B. Nasihatkon and R. Hartley. Graph connectivity in sparse subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2137–2144, 2011.

[103] S. A. Nene, S. K. Nayar, H. Murase, et al. Columbia object image library (COIL-20). Technical report, 1996.

[104] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 849–856, 2002.

[105] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, volume 5, page 3, 2000.

[106] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2-3):103–134, 2000.

[107] N. J. Nilsson. Learning machines. 1965.

[108] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[109] Y. Panagakis and C. Kotropoulos. Elastic net subspace clustering applied to pop/rock music structure analysis. *Pattern Recognition Letters*, 38:46–53, 2014.

[110] X. Peng, L. Zhang, and Z. Yi. Scalable sparse subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 430–437, 2013.

[111] J. C. Pereira, E. Coviello, G. Doyle, N. Rasiwasia, G. R. Lanckriet, R. Levy, and N. Vasconcelos. On the role of correlation and abstraction in cross-modal multimedia retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):521–535, 2013.

[112] V. Premachandran and R. Kakarala. Consensus of k-nns for robust neighborhood selection on graph-based manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1594–1601, 2013.

[113] G. Punj and D. W. Stewart. Cluster analysis in marketing research: Review and suggestions for application. *Journal of Marketing Research*, 20(2):134–148, 1983.

[114] S. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1832–1845, 2010.

[115] S. R. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[116] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. *WACV/MOTION*, 2, 2005.

[117] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[118] M. Sahami and D. Koller. *Using machine learning to improve information access*. PhD thesis, Stanford University, Department of Computer Science, 1998.

[119] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, and C.-T. Lin. A review of clustering techniques and developments. *Neurocomputing*, 267:664–681, 2017.

[120] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[121] H. Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

[122] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–93, 2008.

[123] J. Shi and J. Malik. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, page 107, 2000.

[124] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination, and expression (pie) database. In *Proceedings of the IEEE Conference on Automatic Face and Gesture Recognition*, pages 53–58, 2002.

[125] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[126] R. J. Solomonoff. A formal theory of inductive inference. part i. *Information and control*, 7(1):1–22, 1964.

[127] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 945–952, 2002.

[128] A. Talwalkar, S. Kumar, M. Mohri, and H. A. Rowley. Large-scale svd and manifold learning. *Journal of Machine Learning Research*, 14(1):3129–3152, 2013.

[129] J. Tanha, M. van Someren, and H. Afsarmanesh. Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics*, 8(1):355–370, 2017.

[130] K. Taşdemir. Vector quantization based approximate spectral clustering of large datasets. *Pattern Recognition*, 45(8):3034–3044, 2012.

[131] K. Taşdemir, B. Yalçin, and I. Yildirim. Approximate spectral clustering with utilized similarity information using geodesic based hybrid distance measures. *Pattern Recognition*, 48(4):1465–1477, 2015.

[132] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[133] I. Triguero, S. García, and F. Herrera. Seg-ssc: A framework based on synthetic examples generation for self-labeled semi-supervised classification. *IEEE Transactions on Cybernetics*, 45(4):622–634, 2014.

[134] R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[135] P. Tseng. Nearest q-flat to m points. *Journal of Optimization Theory and Applications*, 105(1):249–252, 2000.

[136] V. Vapnik and V. Vapnik. Statistical learning theory, 1998.

[137] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[138] R. Vidal. A tutorial on subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2010.

[139] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1945–1959, 2005.

[140] M. Vladymyrov and M. Carreira-Perpinán. The variational nystrom method for large-scale spectral problems. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

[141] M. Vladymyrov and M. A. Carreira-Perpinán. Entropic affinities: Properties and efficient numerical computation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 477–485, 2013.

[142] M. Vladymyrov and M. A. Carreira-Perpinán. Locally linear landmarks for large-scale manifold learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 256–271, 2013.

[143] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[144] B. Wang and Z. Tu. Affinity learning via self-diffusion for image segmentation and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2312–2319, 2012.

[145] J. Wang, S.-F. Chang, X. Zhou, and S. T. Wong. Active microscopic cellular image annotation by superposable graph transduction with imbalanced labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[146] J. Wang, T. Jebara, and S.-F. Chang. Semi-supervised learning using greedy max-cut. *Journal of Machine Learning Research*, 14(Mar):771–800, 2013.

[147] Y.-X. Wang, H. Xu, and C. Leng. Provable subspace clustering: When lrr meets ssc. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 64–72, 2013.

[148] C. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 682–688, 2001.

[149] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[150] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. *arXiv preprint arXiv:1806.03536*, 2018.

[151] L. Xu and D. Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 5, page 13, 2005.

[152] Z. Xu, R. Jin, J. Zhu, I. King, and M. Lyu. Efficient convex relaxation for transductive support vector machine. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 1641–1648, 2008.

[153] D. Yan, L. Huang, and M. I. Jordan. Fast approximate spectral clustering. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 907–916, 2009.

[154] X. Yang, S. Koknar-Tezel, and L. J. Latecki. Locally constrained diffusion process on locally densified distance spaces with applications to shape retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 357–364, 2009.

[155] X. Yang, L. Prasad, and L. J. Latecki. Affinity learning with diffusion on tensor product graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):28–38, 2012.

[156] Y. Yang and S. Newsam. Bag-of-visual-words and spatial extensions for land-use classification. In *Proceedings of theSIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 270–279, 2010.

[157] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[158] C. You, C.-G. Li, D. P. Robinson, and R. Vidal. Oracle based active set algorithm for scalable elastic net subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3928–3937, 2016.

[159] C. You and R. Vidal. Geometric conditions for subspace-sparse recovery. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1585–1593, 2015.

[160] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 1601–1608, 2005.

[161] W. Zhan and M.-L. Zhang. Inductive semi-supervised multi-label learning with co-training. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1305–1314, 2017.

[162] K. Zhang and J. T. Kwok. Clustered nyström method for large scale manifold learning and dimension reduction. *IEEE Transactions on Neural Networks*, 21(10):1576–1587, 2010.

[163] K. Zhang, I. W. Tsang, and J. T. Kwok. Improved nyström low-rank approximation and error analysis. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1232–1239, 2008.

[164] R. Zhang, F. Nie, and X. Li. Semisupervised learning with parameter-free similarity of label and side information. *IEEE Transactions on Neural Networks and Learning Systems*, 30(2):405–414, 2019.

[165] T. Zhang, A. Szlam, Y. Wang, and G. Lerman. Hybrid linear modeling via local best-fit flats. *International Journal of Computer Vision*, 100(3):217–240, 2012.

[166] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 321–328, 2004.

[167] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 1601–1608, 2007.

[168] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 169–176, 2004.

[169] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[170] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[171] X. Zhu, C. Change Loy, and S. Gong. Constructing robust affinity graphs for spectral clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1450–1457, 2014.

[172] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Citeseer, 2002.

[173] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine learning (ICML)*, pages 912–919, 2003.

[174] X. J. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.

[175] L. Zhuang, S. Gao, J. Tang, J. Wang, Z. Lin, Y. Ma, and N. Yu. Constructing a non-negative low-rank and sparse graph with data-adaptive features. *IEEE Transactions on Image Processing*, 24(11):3717–3728, 2015.

[176] L. Zhuang, Z. Zhou, S. Gao, J. Yin, Z. Lin, and Y. Ma. Label information guided graph construction for semi-supervised learning. *IEEE Transactions on Image Processing*, 26(9):4182–4192, 2017.