Robust Graph Transduction

Chen Gong

Centre for Quantum Computation & Intelligent Systems

University of Technology, Sydney

A thesis submitted for the degree of

Doctor of Philosophy

May, 2016

I would like to dedicate this thesis to my loving parents and wife.

Acknowledgements

I would like to express my gratitude to all those who helped me during the writing of this thesis.

My deepest gratitude goes first and foremost to my supervisor Prof. Dacheng Tao, for his patient and valuable guidance. I knew little about research when I began my PhD study in UTS. It was Prof. Dacheng Tao who taught me how to find interesting ideas, how to develop solid algorithms, and how to write technical papers all from scratch. He was always ready to help, and was very willing to teach everything he knows to me. Without his illuminating instructions, insightful inspiration, consistent encouragement, and expert guidance, I would not have published papers on the leading journals or conferences in my research field. Therefore, I feel extremely grateful for Prof. Dacheng Tao's effort.

I am also greatly indebted to the Centre for Quantum Computation & Intelligent Systems (QCIS) directed by Prof. Chengqi Zhang. In QCIS, I got the opportunities to learn from many world-famous experts; I met many colleagues with great enthusiasm for scientific research; and I was also permitted to attend many prestigious international conferences related to my research. Studying in QCIS and also UTS will be a fantastic memory that I will never forget.

I would also like to thank Prof. Jie Yang from Shanghai Jiao Tong University. Without his introduction, I would not have the opportunity to study in UTS under the supervision of Prof. Dacheng Tao.

Last but not the least, my gratitude also extends to my family who have been consistently supporting, encouraging and caring for me all of my life!

Abstract

Given a weighted graph, graph transduction aims to assign unlabeled examples explicit class labels rather than build a general decision function based on the available labeled examples. Practically, a dataset usually contains many noisy data, such as the "bridge points" located across different classes, and the "outliers" that incur abnormal distances from the normal examples of their classes. The labels of these examples are usually ambiguous and also difficult to decide. Labeling them incorrectly may further bring about erroneous classifications on the remaining unlabeled examples. Therefore, their accurate classifications are critical to obtaining satisfactory final performance.

Unfortunately, current graph transduction algorithms usually fall short of tackling the noisy but critical examples, so they may become fragile and produce imperfect results sometimes. Therefore, in this thesis we aim to develop a series of robust graph transduction methodologies via iterative or non-iterative way, so that they can perfectly handle the difficult noisy data points. Our works are summarized as follows:

In Chapter 2, we propose a robust non-iterative algorithm named "**L**abel **P**rediction via **D**eformed **G**raph **L**aplacian" (LPDGL). Different from the existing methods that usually employ a traditional graph Laplacian to achieve label smoothness among pairs of examples, in LPDGL we introduce a deformed graph Laplacian, which not only induces the existing pairwise smoothness term, but also leads to a novel local smoothness term. This local smoothness term detects the ambiguity of each example by exploring the associated degree, and assigns confident labels to the examples with large degree, as well as allocates "weak labels" to the uncertain examples with small degree. As a result, the negative effects of outliers and bridge points are suppressed, leading to more robust transduction performance than some existing representative algorithms. Although LPDGL is designed for transduction purpose, we show that it can be easily extended to inductive settings.

In Chapter 3, we develop an iterative label propagation approach, called "**F**ick's **L**aw **A**ssisted **P**ropagation" (FLAP), for robust graph transduction.

To be specific, we regard label propagation on the graph as the practical fluid diffusion on a plane, and develop a novel label propagation algorithm by utilizing a well-known physical theory called Fick's Law of Diffusion. Different from existing machine learning models that are based on some heuristic principles, FLAP conducts label propagation in a "natural" way, namely when and how much label information is received or transferred by an example, or where these labels should be propagated to, are naturally governed. As a consequence, FLAP not only yields more robust propagation results, but also requires less computational time than the existing iterative methods.

In Chapter 4, we propose a propagation framework called "**T**eaching-to-**L**earn and **L**earning-to-**T**each" (TLLT), in which a "teacher" (*i.e.* a teaching algorithm) is introduced to guide the label propagation. Different from existing methods that equally treat all the unlabeled examples, in TLLT we assume that different examples have different classification difficulties, and their propagations should follow a simple-to-difficult sequence. As such, the previously "learned" simple examples can ease the learning for the subsequent more difficult examples, and thus these difficult examples can be correctly classified. In each iteration of propagation, the teacher will designate the simplest examples to the "learner" (*i.e.* a propagation algorithm). After "learning" these simplest examples, the learner will deliver a learning feedback to the teacher to assist it in choosing the next simplest examples. Due to the collaborative teaching and learning process, all the unlabeled examples are propagated in a well-organized sequence, which contributes to the improved performance over existing methods.

In Chapter 5, we apply the TLLT framework proposed in Chapter 4 to accomplish saliency detection, so that the saliency values of all the superpixels are decided from simple superpixels to more difficult ones. The difficulty of a superpixel is judged by its informativity, individuality, inhomogeneity, and connectivity. As a result, our saliency detector generates manifest saliency maps, and outperforms baseline methods on the typical public datasets.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The notion of transduction was originally proposed by Gammerman et al. [1998], which means that we are interested in the classification of a particular set of examples rather than a general decision function for classifying the future unseen examples. In other words, if we know the examples to be classified in advance, and do not care about the explicit decision function, then transduction is an ideal mathematical tool to solve the related problem.

A critical issue in transductive algorithms is how to exploit the relationship among different examples to aid the classifications on the unlabeled examples. One feasible solution suggested by Zhu et al. [2003a] is to use graph to model the similarity between pairs of examples, and the corresponding technique is called graph transduction.

In this chapter, we briefly introduce the background of graph transduction, thoroughly review the related literatures, clearly elaborate our motivations and contributions, and finally present the organization of the entire thesis.

1.1 Background

Formally, we use the notations $\mathcal{X}$ and $\mathcal{Y}$ to denote the example space and label space, respectively. Given a set of examples $\Psi = \{\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d, i = 1, 2, \cdots, n, n = l + u\}$, in which the first l elements are examples with the labels $\{y_i\}_{i=1}^{l} \in \mathcal{Y} \in \{1, -1\}$, and the rest are u unlabeled examples. We use $\mathcal{L} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_l, y_l)\}$ to denote the labeled set drawn from the joint distribution P defined on $\mathcal{X} \times \mathcal{Y}$, and $\mathcal{U} = \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \cdots, \mathbf{x}_{l+u}\}$ to represent the unlabeled set drawn from the unknown marginal distribution $P_{\mathcal{X}}$ of P. Then the target of a transductive algorithm is to find the labels $y_{l+1}, y_{l+2}, \cdots, y_{l+u}$ of every unlabeled examples $\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \cdots, \mathbf{x}_{l+u}$ in $\mathcal{U}$ based on Ψ. For graph transduction, a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is usually built where $\mathcal{V}$ is the vertex set composed of all the examples in Ψ, and $\mathcal{E}$ is the edge set recording

the relationship among all the vertices[1]. $\mathbf{W}_{n\times n}$ is the adjacency matrix of graph $\mathcal{G}$, in which the element ω_{ij} encodes the similarity between vertices $\mathbf{x}_i$ and $\mathbf{x}_j$. The degree of the i-th vertex is defined by $d_{ii} = \sum_{j=1}^{n} \omega_{ij}$, and $\mathbf{D}$ is a diagonal matrix with $(\mathbf{D})_{ii} = d_{ii}$ for $1 \leq i \leq n$. Therefore, the volume of graph $\mathcal{G}$ can be further formulated as $v = \sum_{i=1}^{n} d_{ii}$. Based on the adjacency matrix $\mathbf{W}$ and the degree matrix $\mathbf{D}$, we further define the graph Laplacian as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, which will play an important role in the following chapters.

An illustration of graph $\mathcal{G}$ consisted of seven vertices is presented in Fig. 1.1, of which the associated adjacency matrix $\mathbf{W}$, degree matrix $\mathbf{D}$, and graph Laplacian $\mathbf{L}$ are

$$\mathbf{W} = \begin{array}{c} \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \\ \mathbf{x}_6 \\ \mathbf{x}_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 & \mathbf{x}_5 & \mathbf{x}_6 & \mathbf{x}_7 \end{array} \\ \begin{pmatrix} 0 & 0.7 & 0.8 & 0 & 0 & 0 & 0 \\ 0.7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.8 & 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0 & 0.5 & 0.6 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0.9 \\ 0 & 0 & 0 & 0 & 0.6 & 0.9 & 0 \end{pmatrix} \end{array},$$

$$\mathbf{D} = \begin{pmatrix} 1.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.5 \end{pmatrix}, \tag{1.1}$$

and

$$\mathbf{L} = \begin{pmatrix} 1.5 & -0.7 & -0.8 & 0 & 0 & 0 & 0 \\ -0.7 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ -0.8 & 0 & 0.9 & -0.1 & 0 & 0 & 0 \\ 0 & 0 & -0.1 & 0.7 & -0.6 & 0 & 0 \\ 0 & 0 & 0 & -0.6 & 1.7 & -0.5 & -0.6 \\ 0 & 0 & 0 & 0 & -0.5 & 1.4 & -0.9 \\ 0 & 0 & 0 & 0 & -0.6 & -0.9 & 1.5 \end{pmatrix}.$$

From this example, we know that two questions should be answered to establish a graph, namely: 1) how to decide the existence or absence of an edge between two vertices, and 2) how to compute the edge weight (*i.e.* the similarity between the two examples) given an edge. Based on the different ways for generating the edges, we list some commonly adopted graphs as follows:

[1]We use "examples" and "vertices" interchangeably in this thesis for different explanation purpose.

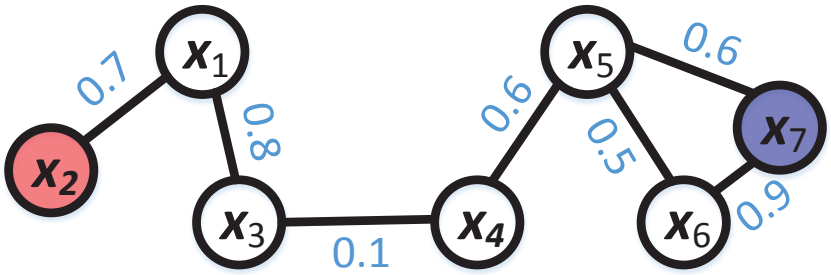

Figure 1.1: The illustration of graph, where the circles represent the vertices $\mathbf{x}_1 \sim \mathbf{x}_7$ and the lines are edges. The red circle denotes the positive example with label 1, while blue circle denotes the negative example with label -1. The numbers near the edges are weights evaluating the similarity between the two connected vertices.

- **Fully connected graph.** In this graph, all pairs of vertices are connected by an edge, so there are totally $n(n-1)/2$ edges in the graph.
- **ε-neighborhood graph.** Two vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ is connected by an edge if and only if the Euclidean distance between them satisfies $\|\mathbf{x}_i - \mathbf{x}_j\| \leq \varepsilon$ where ε is a predefined threshold.
- **K-nearest neighbourhood (KNN) graph.** Two vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ should be linked if $\mathbf{x}_i$ is among the K nearest neighbours of $\mathbf{x}_j$ or $\mathbf{x}_j$ belongs to the K nearest neighbours of $\mathbf{x}_i$. This is the most popular graph widely adopted by massive existing algorithms.
- **Mutual KNN graph.** Two vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ should be connected by an edge if $\mathbf{x}_i$ is among the K nearest neighbours of $\mathbf{x}_j$, and simultaneously $\mathbf{x}_j$ is one of the K neighbours of $\mathbf{x}_i$. Note that mutual KNN graph slightly differs from KNN graph in that an edge exists when both $\mathbf{x}_i$ and $\mathbf{x}_j$ are neighbors of each other, therefore mutual KNN graph is usually sparser than then KNN graph under the same choice of K.

After edge establishment, the next step is to compute the weights of these edges. There are three traditional approaches to determine the edge weights:

- **Binary 0-1 weight.** The weight is 1 as long as there is an edge between two vertices, and 0 otherwise.
- **Gaussian kernel function.** The weight ω_{ij} between $\mathbf{x}_i$ and $\mathbf{x}_j$ is computed by the Gaussian kernel function, namely

$$\omega_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right), \tag{1.2}$$

where σ is the kernel width. The weight ω_{ij} is large if $\mathbf{x}_i$ and $\mathbf{x}_j$ are close in the Euclidean space.

- **Locally linear embedding.** By assuming that the central vertex $\mathbf{x}_i$ can be linearly reconstructed by its K neighbors $\mathcal{N}(\mathbf{x}_i)$, the weights are decided by solving the following optimization problem:

$$\begin{array}{ll} \min_{\omega_{ij}} & \left\| \mathbf{x}_i - \sum_{j:\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} \omega_{ij} \mathbf{x}_j \right\|^2 \\ s.t. & \sum_j \omega_{ij} = 1, \quad \omega_{ij} \geq 0 \end{array} . \tag{1.3}$$

Though above weighting methods are different, they produce the weights that are in the range $[0, 1]$. Apart from above traditional graph construction techniques, more advanced approaches have been proposed in recent years to improve the scalability Liu et al. [2010]; Wang and Xia [2012]; Zhang and Wang [2010]; Zhu and Lafferty [2005] or robustness Chen et al. [2013]; Jebara et al. [2009]; Karasuyama and Mamitsuka [2013a,b]; Li and Fu [2013]. We refer the readers to D. Sousa et al. [2013] for more insightful studies and comparisons on various graph construction strategies.

Based on the established graph $\mathcal{G}$, the labeled examples in $\mathcal{L}$ are regarded as seed vertices, and they will spread their label information to the remaining unlabeled vertices via the edges in $\mathcal{G}$. For example, in Fig. 1.1 the vertices $\mathbf{x}_2$ and $\mathbf{x}_7$ are labeled examples with labels 1 and -1, respectively, while the remaining vertices are originally unlabeled. Then graph transduction aims to diffuse the label information carried by $\mathbf{x}_2$ and $\mathbf{x}_7$ to the unlabeled $\mathbf{x}_1$, $\mathbf{x}_3$, $\mathbf{x}_4$, $\mathbf{x}_5$, and $\mathbf{x}_6$, so that they are accurately classified as positive or negative.

Graph transduction belongs to the scope of Semi-Supervised Learning (SSL) Chapelle et al. [2006]; Zhu and Goldberg [2009], which aims to predict the labels of a large amount of unlabeled examples given only a few labeled examples (namely $l \ll u$). The key motivation of SSL is to elaborately exploit the data structure revealed by both scarce labeled examples and the massive unlabeled examples. Though these unlabeled examples do not have explicit labels, they often reflect the structure of the entire data distribution. Apart from the transductive semi-supervised methods Azran [2007]; Blum et al. [2004]; Chang and Yeung [2004]; Liu et al. [2010]; Wang et al. [2008b, 2009b]; Wu and Schölkopf [2007]; Zhou and Bousquet [2003]; Zhou and Schölkopf [2004]; Zhu et al. [2003a], there also exist a variety of inductive SSL methods, such as Belkin et al. [2006]; Ji et al. [2012]; Li and Zhou [2011, 2015]; Li et al. [2009]; Quang et al. [2013]; Vapnik [1998]; Wang and Chen [2013]; Wang et al. [2012]. Different from transduction, an inductive algorithm takes Ψ as the training set to train a suitable $f : \mathcal{X} \to \mathcal{Y}$, which is able to predict the label $f(\mathbf{x}_t) \in \mathcal{Y} \in \mathbb{R}$ of an unseen test example $\mathbf{x}_t \in \mathcal{X} \in \mathbb{R}^d$. In this thesis, we mainly focus on developing the transductive algorithms based on an undirected graph as shown by Fig. 1.1.

As an important branch of SSL, graph transduction inherits many ideal properties of SSL, such as the capability of handling insufficient labeled examples. Therefore, graph transduction can be utilized for solving various practical problems such as:

- **Interactive image segmentation.** Interactive image segmentation requires the user to annotate a small number of "seed pixels" as foreground or background, and then the segmentation algorithm will automatically segment the foreground region out of the irrelevant background. In this application, it is intractable to manually annotate a large amount of seed pixels, because such annotation is time-consuming when the input image contains hundreds of thousands of pixels.

- **Web-scale image/text annotation.** Recent years have witnessed the vigourous development of the industry of Internet. There are numerous new image/text data uploaded to the Internet everyday. Current search engines such as Google and Baidu largely depend on the annotated tags to provide users useful search results. However, annotating all the new image/text data manually is infeasible because of the unacceptable human labor cost. Therefore, graph transduction can be employed to annotate the massive unlabeled new data based on the small amount of previously annotated data.

- **Protein structure prediction.** In protein 3D structure prediction, a DNA sequence is an example and its label is the 3D protein folding structure. It often takes months of laboratory work for researchers to identify a single protein's 3D structure, so it is impossible to prepare sufficient labeled examples for predicting the structure of an unseen DNA sequence. By building a graph over all the labeled and unlabeled protein examples, we may fully exploit the limited labeled examples and the relationship among examples, making it possible to infer the structures of new DNA sequences.

- **Social network analysis.** In recent years, various social websites such as Twitter and Facebook have gained much popularity among the people all over the world. The social network is a natural graph in which each user is a vertex and their interactions are edges reflecting the closeness of their interpersonal relationship. Therefore, graph transduction can be used to discover the implicit social behaviours, such as community establishment, user influence, and rumor spreading, etc.

1.2 Related Work

Due to the extensive applications and solid mathematical foundations, graph transduction has been investigated by many researchers and various transductive algorithms

have been developed as a result. The existing graph transduction methodologies can be divided into iterative methods and non-iterative methods. This section will review the related literatures according to this taxonomy.

1.2.1 Non-iterative Methods

A non-iterative method usually fit into an optimization framework. The unlabeled examples are then classified by directly minimizing an objective function on the graph.

For example, Joachims [2003] classifies unlabeled examples by finding the best graph partition to minimize a pre-defined energy function. They leverage the theory of graph cut Shi and Malik [2000] and solve the problem via utilizing the spectral property of the graph. Zhu et al. [2003a] regard the graph as a Gaussian random field, where the mean is characterized by harmonic functions. The most important contribution of this work is that they propose a smoothness regularizer based on the graph Laplacian $\mathbf{L}$, which enforces the nearby examples to obtain similar labels on a graph. The authors also elegantly related their algorithm with random walks, electric networks, and spectral graph theory. This work was extended to large-scale situations by incorporating the generative mixture models to construct a much smaller "backbone graph" with vertices induced from the mixture components Zhu and Lafferty [2005]. Another method that is applicable to large datasets is proposed by Fergus et al. [2009], in which the spectral graph theory is adopted to efficiently construct accurate numerical approximations to the eigenvectors of the normalized graph Laplacian. As a result, their method achieves linear complexity w.r.t. the number of examples.

Inspired by Zhu et al. [2003a], a variety of regularizers were developed afterwards according to different heuristics. Belkin et al. [2004a,b] deployed the Tikhonov regularization for graph transduction which requires that the sum of assigned labels equal to 1. Belkin et al. [2006] also proposed the manifold regularization to regulate the variations of examples' labels to vary smoothly along the manifold. They explore the geometry of the data distribution by postulating that its support has the geometric structure of a Riemannian manifold. They extend the traditional Regularized Least Squares (RLS) method and Support Vector Machines (SVM) by utilizing the properties of Reproducing Kernel Hilbert Spaces (RKHS). The proved new Representer theorem provides both solutions and theoretical basis to their algorithms. Furthermore, Belkin and Niyogi [2005] and Belkin and Niyogi [2008] showed the theoretical foundation about manifold regularization and proved that the manifold assumption can reliably tackle some situations that the fully supervised learning would fail. The scalability of Belkin et al. [2006] was improved by Chen et al. [2012b] via an introduced intermediate decision variable, which avoids the computation of the inverse of a large matrix appeared in Belkin et al. [2006]. Karlen et al. [2008] also proposed a large-scale manifold regularization method based on SVM, which is solved via stochastic gradient descend. The effectiveness of manifold regularization has been demonstrated

in feature selection Xu et al. [2010] and dimensionality reduction Huang et al. [2012]. Different from Belkin et al. [2006] that assumes single manifold is embedded in the dataset, Goldberg et al. [2009] consider that there are multiple manifolds hidden in the dataset and present a "cluster-then-label" method to solve the transduction problem.

Apart from the regularizers mentioned above, Wu and Schölkopf [2007] developed the local learning regularizer to predict each example's label from those of its neighbors. Wang et al. [2008a] extends this work by further incorporating a global regularizer. Orbach and Crammer [2012] utilized the degree information to evaluate the quality of different vertices, based on which they cast transduction as a convex optimization problem that is able to assign confident labels to the unlabeled examples. The importance of local regularizer is also observed by Xiang et al. [2010], which developed the local splines in Sobolev space so that the examples can be directly mapped to their class labels. They also prove that the designed objective function has a globally optimal solution.

Another trend for non-iterative graph transduction is to use the information theory to model the relationship among different examples. Szummer and Jaakkola [2002b] propose an information theoretic regularization framework for combining conditional and marginal densities in a semi-supervised estimation setting. The framework admits both discrete and continuous densities. Subramanya and Bilmes [2011] minimize the Kullback-Leibler divergence (also known as "relative entropy") between discrete probability measures that encode class membership probabilities. They show that the adopted alternating optimization process has a closed-form solution for each subproblem and converges to the correct optima. Grandvalet and Bengio [2004] discover that the unlabeled examples are mostly beneficial when classes have small overlap, and they use the conditional entropy to assess the usefulness of unlabeled examples. By introducing the entropy regularization, Grandvalet and Bengio [2004] transform the graph transduction problem into a standard supervised learning problem.

There also exist numerous algorithms that extend the conventional graph transduction to different settings. For example, Goldberg et al. [2007], Ma and Jan [2013] and Gong et al. [2014a] develop various transductive algorithms when the "must-links" and "cannot-links" between examples are available. Ding et al. [2009] generalize the Laplacian embedding to K-partite graph rather than the traditional graphs introduced in Section 1.1. Cai et al. [2013] and Karasuyama and Mamitsuka [2013b] associate each view with a graph, and build multiple graphs to deal with multi-view graph transduction. Liu and Chang [2009] develop multi-class graph transduction by incorporating class priors, and obtain a simple closed-form solution. For multi-label situations, Kong et al. [2013] build a graph on all the examples and conduct multi-label transduction by assuming that the labels of a central example can be linearly reconstructed by its neighbors' labels. Differently, Wang et al. [2009a] and Chen et al. [2008] establish two graphs in example space and label space, and use Green's function and Sylvester equation respectively to capture the dependencies between different

examples and labels.

Other representative non-iterative methods include Erdem and Pelillo [2012]; Goldberg et al. [2010]; Kim and Theobalt [2015]; Li and Zemel [2014]; Niu et al. [2014]; Sinha and Belkin [2009]; Zhu et al. [2005].

1.2.2 Iterative Methods

Iterative methods conduct graph transduction by gradually propagating the labels of seed vertices to the unlabeled vertices. The algorithms of this type usually establish an iterative expression, based on which the label information is diffused on the graph in an iterative way. Therefore, the iterative graph transduction is also known as label propagation. In each propagation, the labels of all the examples are updated by considering both their previous states and the influence of other examples. The entire iteration process can be proved to converge to a stationary state, which conveys the labels of originally unlabeled examples.

The notion of "label propagation" was introduced by Zhu and Ghahramani [2002], which proposed to iteratively propagate class labels on a weighted graph by executing random walks with clamping operations. This algorithm was successfully applied to shape retrieval by Yang et al. [2008] and Bai et al. [2010]. Similarly to Zhu and Ghahramani [2002], Szummer and Jaakkola [2002a] combine a limited number of labeled examples with a Markov random walk representation over the unlabeled examples. The random walk process exploits low-dimensional structure of the dataset in a robust and probabilistic manner. Azran [2007] associates each vertex with a particle that moves on the graph according to a transition probability matrix. By treating the labeled vertices as absorbing states of the Markov random walk, the probability of each particle to be absorbed by the different labeled points is then employed to derive a distribution over the labels of unlabeled examples. Furthermore, Wu et al. [2012] proposed partially absorbing random walks, in which a random walk is with probability p_i being absorbed at current state i, and with probability $1 - p_i$ follows a random edge out of the state i. When the walking process is completed, the probability of each particle to be absorbed by the labeled examples can be determined, which helps to estimate the labels of all the examples.

Unlike Szummer and Jaakkola [2002a]; Wu et al. [2012]; Zhu and Ghahramani [2002] which work on asymmetric normalized graph Laplacians $\mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$ ($\mathbf{I}$ denotes the identity matrix in this thesis), Zhou and Bousquet [2003]; Zhou et al. [2004b] deployed a symmetric normalized graph Laplacian $\mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ to implement propagation. However, the computational burden of this method is heavy, so Fujiwara et al. Fujiwara and Irie [2014] proposed the efficient label propagation by iteratively computing the lower and upper bounds of labeling values to prune unnecessary label computations. Lin and Cohen [2011] also proposed a "path-folding trick" to adapt Zhou and Bousquet [2003] to large-scale text data. In order to address the problem

that Zhou and Bousquet [2003] is sensitive to the initial set of labels provided by the user, Wang et al. [2008b] introduced a node normalization terms to resist the label imbalance. They provide an alternating minimization scheme that incrementally adjusts the objective function and the labels towards a reliable local minimum. Besides, inspired by Zhou and Bousquet [2003], Wang and Zhang [2006, 2008]; Wang et al. [2009b] established a graph by assuming that an example can be linearly reconstructed by its neighbours, and adopted the similar iterative expression to Zhou and Bousquet [2003] for label propagation.

Considering that the fixed adjacency matrix of a graph cannot always faithfully reflect the similarities between examples during propagation, Wang et al. [2013] developed dynamic label propagation to update the edge weights dynamically by fusing available multi-label and multi-class information. Wang and Tu [2012] proposed to learn an accurate adjacency matrix via self-diffusion, in which the optimal iteration number $t^\star$ is heuristically determined based on the defined "degree of freedom".

Other representative iterative methods include the mixed label propagation for handling the pair-wise constraints Tong and Jin [2007], label propagation on directed graph Zhou et al. [2004a], graph-based propagation under probabilistic point-wise smoothness Fang et al. [2014], and Wasserstein propagation for diffusing the probability distributions and histograms Solomon et al. [2014].

It is worth emphasizing that some iterative methods also have a closed convergent result, so they can also be attributed to non-iterative category, such as Wang and Zhang [2006]; Zhou and Bousquet [2003]; Zhou et al. [2004b]; Zhu and Ghahramani [2002]. We classify them into iterative category because they are originally derived for iterative label propagation, even though each of them also has a closed-form solution that can be understood as the optimizer of a regularization framework.

1.3 Motivations and Contributions

Although the existing graph transduction algorithms mentioned above have obtained encouraging results to some extent, they may become fragile under certain circumstances. Specifically, the practical data are usually "dirty", which means that there are a considerable amount of "bridge points" located across different classes, and the outliers that incur abnormal distances from the normal examples of their classes. The labels of these examples are usually ambiguous and also difficult to decided. As a result, they are very likely to mislead the transduction and result in error-prone classifications. Therefore, the robustness of current transductive methods should be improved to better handle such difficult or abnormal examples.

Fig. 1.2 shows an example that some of the existing algorithms are misled by the difficult bridge point located between the two classes. Fig. 1.2(a) presents the adopted

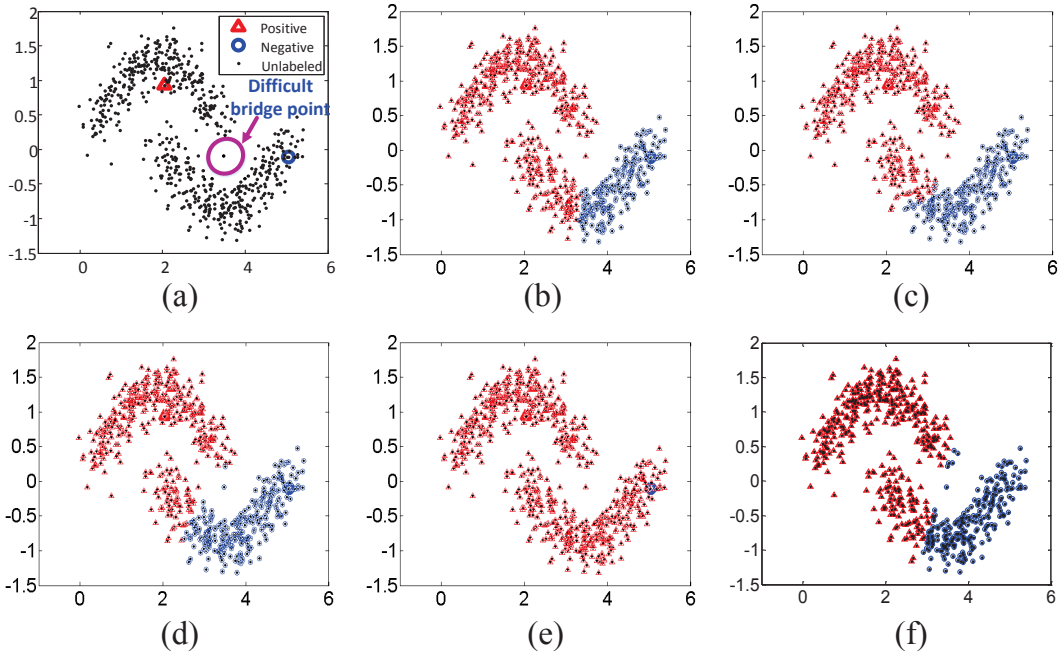

Figure 1.2: The transductive results of some representative methods on the *DoubleMoon* dataset. (a) is the initial state with marked labeled examples and difficult bridge points. (b), (c), (d), (e), (f) are incorrect results produced by Zhu and Ghahramani [2002], Zhou and Bousquet [2003], Wang et al. [2009b], Wang et al. [2013] and Wang et al. [2008a], respectively.

two-dimensional *DoubleMoon* dataset, which consists of 640 examples that are equally divided into two moons. This dataset is contaminated by the Gaussian noise with standard deviation 0.15, and each class has only one initial labeled example. The classification results produced by Zhu and Ghahramani [2002], Zhou and Bousquet [2003], Wang et al. [2009b], Wang et al. [2013] and Wang et al. [2008a] are illustrated in (b), (c), (d), (e), (f), respectively. From Fig. 1.2(a), we observe that the distance between the two classes is very small, and thus a difficult bridge point is distributed among the intersection region between the two moons. As a result, some representative existing methods cannot perfectly deal with this ambiguous bridge point and thus generate unsatisfactory results.

The main motivation of this thesis is to design novel robust graph transduction algorithms so that they can resist the adverse impact of the ambiguous "dirty examples" such as outliers and bridge points. Specifically, three robust transduction algorithms are proposed in this thesis, which are **L**abel **P**rediction via **D**eformed **G**raph **L**aplacian (LPDGL), **F**ick's **L**aw **A**ssisted **P**ropagation (FLAP), and label propagation via **T**eaching-to-**L**earn and **L**earning-to-**T**each (TLLT). Of these, the first one is non-

iterative, and the rest are iterative methods.

In LPDGL, we extend the traditional graph Laplacian $\mathbf{L}$ to the deformed graph Laplacian to define a novel smoothness term, and propose an optimization framework based on the new smoothness term. The unlabeled examples are then assigned accurate labels by solving this optimization problem. We prove that the proposed optimization problem has a closed-form solution, which is also globally optimal. Compared with existing popular transduction methods which simply exploit the smoothness between pairs of examples (*i.e.* pairwise smoothness), a novel local smoothness term is introduced "naturally", which is critical for our model to better deal with ambiguous examples. This local smoothness term considers the examples and their neighbours as a whole, and regulates the labels of uncertain examples to small values in order to suppress their negative influence on other examples. Furthermore, we show that the proposed LPDGL can be easily extended to inductive cases even though it is originally designed for the transductive purpose. Theoretical studies reveal that the incorporated free parameters are easy to tune because the result of LPDGL is not sensitive to the variations of these free parameters. We also theoretically analyse the robustness of LPDGL, based on which the generalization bound is derived. Experiments on a variety of real-world datasets demonstrate that LPDGL achieves top level performance on both transductive and inductive settings by comparing it with popular graph-based algorithms.

In FLAP, we utilize the well-known physical theory, Fick's First Law of Diffusion, to guide the label propagation. As a result, FLAP is more lifelike because it is straightforwardly derived from statistical physics. Therefore, when and how much label information is received or transferred by an example, or where these labels should be propagated to, are directly governed by the well-known Fick's law, which is better than decided via some heuristic and ad hoc requirements or criteria exploited in conventional machine learning algorithms. This is beneficial for FLAP to obtain robust performance. In particular, FLAP simulates the diffusion of fluid for label propagation, thus the labeled examples can be regarded as the diffusive sources with high concentration of label information. When the diffusion process starts, the flux of label information will be transferred from the labeled examples to the remaining unlabeled examples. When the diffusion process is completed, all the examples on the graph will receive a certain concentration of label information, providing the foundation for final classification. Another merit by using Fick's First Law of Diffusion is that FLAP makes eigenvalues of the iteration matrix distributed regularly, leading to faster convergence rate than the traditional propagation algorithms such as Zhu and Ghahramani [2002], Zhou and Bousquet [2003] and Wang et al. [2009b]. We conduct the experiments on several computer vision and pattern recognition repositories, including handwritten digit recognition, face recognition and teapot image classification. Comprehensive experimental evaluations on synthetic and practical datasets reveal that FLAP obtains encouraging results in terms of both

accuracy and efficiency.

In TLLT, we propose a novel teaching-to-learn and learning-to-teach framework for robust label propagation. Existing graph-based propagation algorithms usually treat unlabeled examples equally, and transmit seed labels to the unlabeled examples that are directly connected to the labeled examples in a graph. Such a popular propagation scheme is very likely to yield inaccurate propagation, because it falls short of tackling ambiguous but critical data points (*e.g.*, outliers). To this end, TLLT treats the unlabeled examples in different levels of difficulties by assessing their reliability and discriminability, and explicitly optimizes the propagation quality by manipulating the propagation sequence to move from simple to difficult examples. Specifically, we employ the method of Zhu and Ghahramani [2002] as a "learner", and introduce a "teacher" to guide the entire propagation process. In each propagation, the proposed method alternates between two paradigms, teaching-to-learn and learning-to-teach. In the teaching-to-learn step, the learner conducts the propagation on the simplest unlabeled examples designated by the teacher. In the learning-to-teach step, the teacher incorporates the learner's feedback to adjust the choice of the subsequent simplest examples. The TLLT strategy critically improves the accuracy of label propagation, making our algorithm substantially robust to the values of tuning parameters such as the Gaussian kernel width used in graph construction. To the best of our knowledge, this method is the first work to model label propagation as a teaching and learning framework, so that abundant unlabeled examples are activated to receive the propagated labels in a well-organized sequence. Furthermore, we show that TLLT is related to FLAP if the difficult unlabeled examples and simple examples are respectively regarded as the highlands and lowlands in the diffusive simulation. The merits of our algorithm are theoretically justified and empirically demonstrated through experiments performed on both synthetic and real-world datasets.

Moreover, since label propagation has achieved very encouraging performance on salient object detection (or "saliency detection") in recent years, it is natural to apply the TLLT framework to saliency detection for obtaining improved results. A saliency detection algorithm aims to identify the most attractive object in an image, and meanwhile outputs a grey-scale saliency map to indicate the saliency degree of different regions. Saliency propagation refers to the detection method that is based on the label propagation strategy. The propagation sequence generated by existing saliency detection methods is governed by the spatial relationships of image regions, *i.e.*, the saliency value is transmitted between two adjacent regions. However, for the inhomogeneous difficult adjacent regions, such a sequence may incur wrong propagations. Therefore, we attempt to manipulate the propagation sequence for optimizing the propagation quality by using the TLLT algorithm. Intuitively, we postpone the propagations to difficult regions and meanwhile advance the propagations to simple regions. The difficulty of a region is evaluated by its informativity, individuality, inhomogeneity, and connectivity. In the teaching-to-learn step, a teacher

is designed to arrange the regions from simple to difficult and then assign the simplest regions to the learner. In the learning-to-teach step, the learner delivers its learning confidence to the teacher to assist the teacher to choose the subsequent simple regions. Due to the interactions between the teacher and learner, the uncertainty of original difficult regions is gradually reduced, yielding manifest salient objects with optimized background suppression. Extensive experimental results on benchmark saliency datasets demonstrate the superiority of the proposed algorithm over several existing representative saliency detectors.

In summary, the main contributions of this thesis lie in the following four aspects:

1. We propose a novel non-iterative graph transduction algorithm called "LPDGL", which deploys the deformed graph Laplacian to generate a local smoothness term. As a result, the ambiguous data points with uncertain labels are assigned small soft labels, and the examples with definite labels are assigned trustable and confident labels.

2. We propose a novel label propagation algorithm termed "FLAP" from the perspective of physical theory, so that the propagation follows the practical fluid-spreading way. As a consequence, the unlabeled examples are propagated naturally and thus receive more precise label information than using other ad hoc methods. Therefore, the robustness of the entire propagation process is guaranteed.

3. We design a novel framework for accurate label propagation called "TLLT", which requires the interactions between a teacher and a learner. We assume that different examples have different levels of difficulty, and invoke the unlabeled examples to be propagated from a simple-to-difficult sequence. Consequently, the previously learned simple knowledge eases the learning burden of the difficult examples afterwards, leading to the improved accuracy and robustness of our propagation approach.

4. We apply the proposed TLLT methodology to saliency detection in the natural image. By comprehensively evaluating the difficulty scores of different regions, we decide the saliency values of simple regions ahead of more difficult ones, so that all the regions in an image can get confident and accurate saliency values.

1.4 Thesis Structure

To achieve robust graph transduction, this thesis proposes three algorithms LPDGL, FLAP and TLLT based on different motivations, and then applies TLLT to saliency detection tasks. The remaining parts of this thesis are organized as follows:

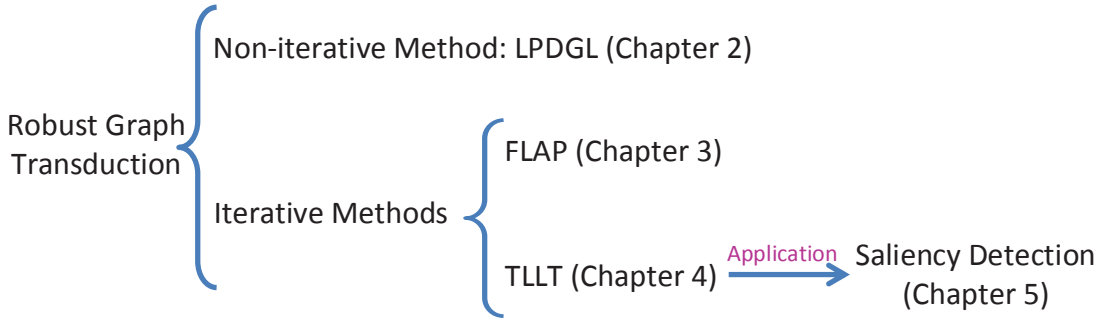

Figure 1.3: The structure of this thesis.

Chapter 2 introduces the LPDGL algorithm for both transductive and inductive purposes.

Chapter 3 introduces the FLAP algorithm for iterative label propagation.

Chapter 4 introduces the TLLT framework for robust label propagation.

Chapter 5 introduces an application of TLLT to saliency detection.

Chapter 6 concludes this thesis and also presents the possible future works.

The structure of the entire thesis is illustrate in Fig. 1.3.

1.5 Publications during PhD Study

1. **Chen Gong**, Dacheng Tao, Keren Fu, Jie Yang. ReLISH: Reliable Label Inference via Smoothness Hypothesis. AAAI, 2014. (oral)

2. **Chen Gong**, Dacheng Tao, Keren Fu, Jie Yang. Signed Laplacian Embedding for Supervised Dimension Reduction. AAAI, 2014.

3. **Chen Gong**, Keren Fu, Artur Loza, Qiang Wu, Jia Liu, Jie Yang. PageRank Tracker: From Ranking To Tracking. IEEE Transactions on Cybernetics (TCYB), 2014, 44(6): 882-893.

4. **Chen Gong**, Dacheng Tao, Keren Fu, Jie Yang. Fick's Law Assisted Propagation for Semisupervised Learning. IEEE Transactions on Neural Networks and Learning Systems (TNNLS), 2015, 26(9): 2148-2162.

5. **Chen Gong**, Dacheng Tao, Keren Fu, Jie Yang. Deformed Graph Laplacian for Semisupervised Learning. IEEE Transactions on Neural Networks and Learning Systems (TNNLS), 2015, 26(10): 2261-2274.

6. **Chen Gong**, Dacheng Tao, Wei Liu, S.J. Maybank, Meng Fang, Keren Fu, Jie Yang. Saliency Propagation From Simple To Difficult. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

7. **Chen Gong**, Dacheng Tao, Jie Yang. Teaching-to-Learn and Learning-to-Teach for Multi-label Propagation. AAAI, 2016. (oral)

8. **Chen Gong**, Dacheng Tao, Wei Liu, Liu Liu, Jie Yang. Label Propagation via Teaching-to-Learn and Learning-to-Teach. IEEE Transactions on Neural Networks and Learning Systems (TNNLS), 2016. (accepted)

Chapter 2

Label Prediction Via Deformed Graph Laplacian

This chapter aims to develop a graph transduction algorithm under the manifold assumption, which assumes that there exists a $\mathbb{C}^\infty$ smooth manifold $\mathcal{M}$ without boundary and with an infinitely differentiable embedding in the ambient example space $\mathcal{X}$. Specifically, we aim to use the limited number of labeled examples $\{\mathbf{x}_i\}_{i=1}^{l} \in \mathbb{R}^d$ and the abundant unlabeled examples $\{\mathbf{x}_i\}_{i=l+1}^{l+u} \in \mathbb{R}^d$ to approximate the embedded manifold. This discovered manifold carries critical information for the distribution of the dataset, which can be utilized to accurately classify the unlabeled examples.

As mentioned in the Chapter 1, graph transduction methods usually deploy a smoothness term to penalize the variation of labels along the manifold. To design such a smoothness term, existing methods usually adopt a standard graph Laplacian to constrain the labels of every pair of examples according to their similarities. The smoothness term defined by the standard graph Laplacian in this case is called a *pairwise smoothness term*. Although this pairwise smoothness term achieves promising performance in both transductive and inductive learning, it is not effective for handling ambiguous examples (shown in Fig. 2.1). Therefore, different from the traditional methods, here we use the deformed graph Laplacian Morbidi [2013] to define a novel smoothness term, and propose an algorithm called Label Prediction via Deformed Graph Laplacian (LPDGL). Compared with other popular methods, LPDGL has the following three advantages as a result of the deformed graph Laplacian:

1. A novel *local smoothness term* is introduced "naturally", which is critical for our model to better deal with ambiguous examples;

2. LPDGL is able to achieve higher classification accuracy than some state-of-the-art methods for both transductive and inductive settings;

3. LPDGL can be regarded as a unified framework of many popular graph

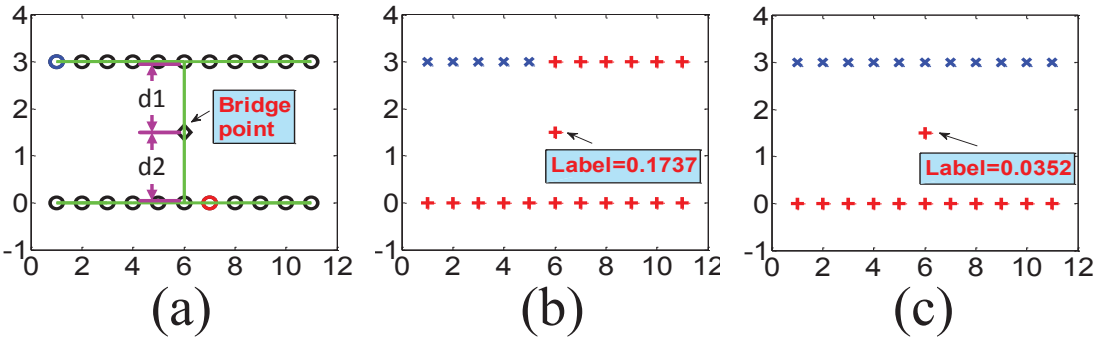

Figure 2.1: The illustration of local smoothness constraint on *DoubleLine* dataset. A k-NN graph with $k = 2$ is built and the edges are shown as green lines in (a). (b) shows the result without incorporating the local smoothness, and (c) is the result produced by the proposed LPDGL. The labels of "bridge point" under two different simulations are highlighted in (b) and (c), respectively.

transduction algorithms.

The local smoothness term mentioned in 1 considers the label smoothness of examples with their neighbors as a whole, and heavily regularizes the example that corresponds to a low degree. This is because an example that has weak edges with its neighbors often confuses the classifier significantly. Such examples can be outliers, or points that are located very close to the decision boundary. These ambiguous examples cannot be reliably classified because there is very little information provided by other examples. Similar idea can be found in Li and Guo [2013], which uses the informative examples in dense regions to conduct active learning. The incorrect classification of ambiguous examples is likely to bring about disastrous results. Taking the *DoubleLine* dataset for example (Fig. 2.1), the red, blue and black circles in (a) represent positive examples, negative examples, and unlabeled examples, respectively. The examples with y-coordinate 3 form the negative class and the points with y-coordinate 0 correspond to the positive class. The point at (6, 1.5) lies exactly in the middle of the two classes ($d_1 = d_2$), and can be attributed to an arbitrary class. We call this point "bridge point" because it will probably serve as a bridge for the mutual transmission of positive and negative labels. In Fig. 2.1 (b), which does not incorporate the local smoothness term, the positive label is mistakenly propagated to the negative class through the "bridge point". This is because the labeled positive example (the red circle in Fig. 2.1 (a)) is closer to the "bridge point" than the labeled negative example (blue circle), so it imposes more effects on the "bridge point". As a consequence, the label of the "bridge point" is +0.1737 (see Fig. 2.1 (b)), which strongly influences the point at (6, 3), and leads to the incorrect classification of more

than half of the negative examples. By comparison, Fig. 2.1 (c) shows that the proposed LPDGL equipped with the local smoothness constraint successfully prohibits the label information from passing through it, and achieves a reasonable result. We observe that the label of "bridge point" is "suppressed" to a very small number (+0.0352), significantly weakening the "strength" of the positive label propagating to the negative points.

LPDGL is formulated as a regularization framework, through which the globally optimal solution is obtained. LPDGL deals with the transductive situations in Euclidean space, and handles the inductive tasks in reproducing kernel Hilbert space (RKHS). Theoretical analyses illustrate that LPDGL is very robust to the choice of training examples, and the probability of the generalization risk being larger than any positive constant is bounded. Therefore, LPDGL performs accurately and reliably. Moreover, the parametric sensitivity is investigated based on the stability theory of solution of equations, from which we find that the classification performance is very robust to a wide choice of parameters. Therefore, the parameters in LPDGL are easy to tune.

LPDGL is demonstrated to be effective in many tough real-world applications, such as handwritten digit recognition, unconstrained face recognition and the detection of violent behaviors. Therefore, the proposed algorithm has high practical value.

2.1 Transduction In Euclidean Space

Given a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ as introduced in the Chapter 1, the existing graph transduction algorithms Belkin et al. [2006]; Zhou and Bousquet [2003]; Zhu et al. [2003a] usually adopt the traditional graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$, to model the smoothness relationship between examples. Specifically, if we use the vector $\mathbf{f} = (f_1, f_2, \cdots, f_n)^\top$ to record the determined soft labels of all the examples $\{\mathbf{x}_i\}_{i=1}^n$ in Ψ, then the smoothness term is formulated as

$$\frac{1}{2}\sum\nolimits_{i=1}^{n}\sum\nolimits_{j=1}^{n}\omega_{ij}(f_i - f_j)^2 = \mathbf{f}^\top\mathbf{L}\mathbf{f}. \tag{2.1}$$

However, the pairwise smoothness (2.1) cannot effectively handle the ambiguous "bridge point" as shown by Fig. 2.1, so we proposes a novel smoothness regularizer defined as

$$\Omega(\mathbf{f}) = \beta\mathbf{f}^\top\mathbf{L}\mathbf{f} + \gamma\mathbf{f}^\top(\mathbf{I} - \mathbf{D}/v)\mathbf{f}, \tag{2.2}$$

in which β and γ are non-negative parameters balancing the weights of the above two terms. The first term $\mathbf{f}^\top\mathbf{L}\mathbf{f}$ is the traditional pairwise smoothness defined by (2.1). It evaluates the smoothness between pairs of $\mathbf{x}_i$ and $\mathbf{x}_j$ over the entire dataset. The second term is the local smoothness term mentioned at the beginning of this chapter,

which can be reformulated as

$$\mathbf{f}^\top(\mathbf{I}-\mathbf{D}/v)\mathbf{f}=\sum\nolimits_{i=1}^{n}(1-d_{ii}/v)f_i^2. \tag{2.3}$$

On a k-NN graph $\mathcal{G}$, d_{ii} records the connective strength among $\mathbf{x}_i$ and its neighbors, so minimizing (2.3) enforces the example with large d_{ii} to obtain a confident soft label f_i, while the example with low degree d_{ii} to receive a relatively weak label. Therefore, minimizing (2.3) requires that the label of each example satisfies a sufficient smoothness with its neighbors.

Actually, a deformed graph Laplacian formulated as $\hat{\mathbf{L}}=\mathbf{I}-\kappa\mathbf{W}-\kappa^2(\mathbf{I}-\mathbf{D})$ has been shown in Morbidi [2013], where κ is a free parameter and $\mathbf{I}$ is an $n\times n$ identity matrix. This deformed graph Laplacian is an instance of a more general theory of deformed differential operators developed in mathematical physics Hislop and Sigal [1996]. The deformation technique was initially proposed for the dilation group, and was applied to many situations afterwards such as Schrödinger operation theory, quantum field theory, and plasma stability theory. Note that the deformed Laplacian $\hat{\mathbf{L}}$ will degenerate to the standard graph Laplacian $\mathbf{L}$ if κ is set to 1. Next we will shed light upon that the proposed smoothness term (2.2) is related to $\hat{\mathbf{L}}$. By denoting $\tilde{\mathbf{L}}=\beta\mathbf{L}+\gamma(\mathbf{I}-\mathbf{D}/v)$, Eq. (2.2) can be expressed as $\Omega(\mathbf{f})=\mathbf{f}^\top\tilde{\mathbf{L}}\mathbf{f}$ and $\tilde{\mathbf{L}}$ here plays an equivalent role as $\mathbf{L}$ in (2.1). Considering that $\mathbf{L}=\mathbf{D}-\mathbf{W}$, we have

$$\begin{aligned}\tilde{\mathbf{L}}&=\gamma\mathbf{I}-\beta\mathbf{W}+(\beta-\gamma/v)\mathbf{D}\\&=(\gamma+\beta-\gamma/v)\left[\mathbf{I}-\frac{\beta v}{\gamma v+\beta v-\gamma}\mathbf{W}-\frac{\beta v-\gamma}{\gamma v+\beta v-\gamma}(\mathbf{I}-\mathbf{D})\right],\end{aligned} \tag{2.4}$$

which equals to $\hat{\mathbf{L}}$ when $\frac{\gamma}{\beta}=\frac{v(v-2)}{v-1}$, and followed by the division by the coefficient $\gamma+\beta-\gamma/v$.

Based on the novel smoothness term, we derive the transductive model of LPDGL in the Euclidean space. Suppose $\mathbf{y}=(y_1,y_2,\cdots y_n)^\top$ is a vector indicating the initial states of all examples, in which $y_i=1,-1,0$ when $\mathbf{x}_i$ is a positive example, negative example and unlabeled example, respectively. Moreover, we define a diagonal matrix $\mathbf{J}_{n\times n}$ with the i-th ($1\leq i\leq n$) diagonal element 1 if $\mathbf{x}_i$ is labeled and 0 otherwise, then the regularization framework of transductive LPDGL is

$$\min_{\mathbf{f}}\ Q(\mathbf{f})=\frac{1}{2}\left[\beta\mathbf{f}^\top\mathbf{L}\mathbf{f}+\gamma\mathbf{f}^\top(\mathbf{I}-\mathbf{D}/v)\mathbf{f}+\|\mathbf{J}(\mathbf{f}-\mathbf{y})\|_2^2\right]. \tag{2.5}$$

The first term in the bracket of (2.5) is the pairwise smoothness term, which indicates that if two examples $\mathbf{x}_1,\mathbf{x}_2\in\mathcal{X}$ distribute nearby in the example space $\mathcal{X}$, then their labels y_1 and y_2 should be also very similar in the label space $\mathcal{Y}$. Compared to the first term that simply evaluates the smoothness between two examples simultaneously, the second local smoothness term, which has been introduced above, considers the

smoothness of examples and their k neighbors in a local region at the same time. If an example $\mathbf{x}_i$ has very small edge weights ω_{ij} $(j = 1, 2, \cdots, k)$ with its k neighbors, it should not receive a confident label f_i because it corresponds to a low degree d_{ii}. As already revealed by Fig. 2.1(c), this manipulation makes the "bridge point" obtain a less reliable soft label, which effectively prevents the mutual transmission of labels belonging to different classes. The third term is a fidelity function which guarantees that the labels of initially labeled examples $\{\mathbf{x}_i\}_{i=1}^{l}$ keep consistent with its initial conditions $\{y_i\}_{i=1}^{l}$ after transduction. To find the minimizer of (2.5), we set the derivative of $Q(\mathbf{f})$ w.r.t $\mathbf{f}$ to $\mathbf{0}$, and obtain

$$\beta\mathbf{L}\mathbf{f} + \gamma(\mathbf{I} - \mathbf{D}/v)\mathbf{f} + \mathbf{J}\mathbf{f} - \mathbf{J}\mathbf{y} = \mathbf{0}. \tag{2.6}$$

Therefore, the optimal $\mathbf{f}$ is expressed as

$$\mathbf{f} = \left[\mathbf{J} + \beta\mathbf{L} + \gamma\left(\mathbf{I} - \mathbf{D}/v\right)\right]^{-1}\mathbf{y}. \tag{2.7}$$

Based on (2.7), the label of $\mathbf{x}_i \in \mathcal{U}$ is further determined as 1 if $f_i > 0$, and -1 otherwise.

Theorem 2.1. *The optimization problem* (2.5) *is convex and the solution is globally optimal.*

Proof. The Hessian matrix of (2.5) is formulated as

$$\mathbf{B} = \mathbf{J} + \beta\mathbf{L} + \gamma(\mathbf{I} - \mathbf{D}/v), \tag{2.8}$$

which is diagonally dominant, so it is a positive definite matrix. Therefore, (2.5) defines a convex optimization problem and the decision function derived from (2.7) is globally optimal. □

Next we investigate the parametric sensitivity of the proposed LPDGL. Parametric sensitivity evaluates the impact of a parameter on the final output of the model. If the output remains substantially unchanged with the wide range of a parameter, we say that the output is insensitive to the choice of this parameter.

In the proposed LPDGL, β and γ are two critical parameters to be tuned. This section aims to verify that the classification results of LPDGL are insensitive to the variation of either of them. The theoretical results provided here will be empirically demonstrated in the experimental Section 2.4.3. Since $\mathbf{B} = \mathbf{J} + \beta\mathbf{L} + \gamma\left(\mathbf{I} - \mathbf{D}/v\right)$, then the impacts of β and γ on $\mathbf{f}$ are studied by investigating the equations $\mathbf{B}\mathbf{f} = \mathbf{y}$ about how $\mathbf{f}$ is affected when the coefficient matrix $\mathbf{B}$ is slightly disturbed. Before discussing the parametric sensitivity of β and γ, we first provide a useful lemma:

Lemma 2.2. *Lancaster and Tismenetsky [1969] Given a set of linear equations* $\mathbf{Bf} = \mathbf{y}$*, where* $\mathbf{B} \in C^{n\times n}$ *is the coefficient matrix and* $\mathbf{f}$ *is the solution. Suppose* $\mathbf{y}$ *at the right-hand side of equations is accurate and* $\mathbf{B}$ *is slightly disturbed by* $\delta\mathbf{B}$*, then the deviation* $\delta\mathbf{f}$ *from the accurate* $\mathbf{f}$ *satisfies*

$$\frac{\|\delta\mathbf{f}\|}{\|\mathbf{f}\|} \leq \frac{\textit{Cond}(\mathbf{B})\,(\|\delta\mathbf{B}\|/\|\mathbf{B}\|\,)}{\textit{1-Cond}(\mathbf{B})\,(\|\delta\mathbf{B}\|/\|\mathbf{B}\|)}, \tag{2.9}$$

where $\|\cdot\|$ *denotes the Euclidean norm and* $\textit{Cond}(\mathbf{B}) = \|\mathbf{B}\|\,\|\mathbf{B}^{-1}\|$ *is the associated condition number.*

2.1.1 Sensitivity of γ

Suppose a small deviation $\delta\gamma$ is added to the parameter γ, then $\delta\mathbf{B}$ in (2.9) is $\delta\mathbf{B} = \delta\gamma\,(\mathbf{I} - \mathbf{D}/v)$, which leads to the departure $\delta\mathbf{f}$ from the accurate solution $\mathbf{f}$. Therefore, we have

$$\begin{aligned}
\frac{\|\delta\mathbf{B}\|}{\|\mathbf{B}\|} &= \frac{\delta\gamma\,\|\mathbf{I} - \mathbf{D}/v\,\|}{\|\mathbf{J} + \beta\mathbf{L} + \gamma\,(\mathbf{I} - \mathbf{D}/v)\|} \\
&= \frac{\delta\gamma\sqrt{\sum_{i=1}^{n}\left(1 - d_{ii}/v\,\right)^2}}{\sqrt{\beta^2\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\omega_{ij}^2 + \sum_{i=1}^{n}\left[\beta d_{ii} + \gamma\,(1 - d_{ii}/v\,)\right]^2 + \xi}} \\
&= \frac{\delta\gamma\sqrt{\sum_{i=1}^{n}\left(1 - d_{ii}/v\,\right)^2}}{\sqrt{\beta^2\sum_{i}\sum_{j}\omega_{ij}^2 + \beta\sum_{i} d_{ii}\left[(\beta - 2\gamma/v)d_{ii} + 2\gamma\right] + \xi + \gamma^2\sum_{i}\left(1 - d_{ii}/v\right)^2}},
\end{aligned} \tag{2.10}$$

where $\xi = 2\sum_{i=1}^{l}\left[\beta d_{ii} + \gamma\,(1 - d_{ii}/v)\right] + l > 0$. Note that the numerator in (2.10) is the same as the last term of denominator except the coefficient, and it is a small number compared to the denominator if γ is slightly disturbed, so $\|\delta\mathbf{B}\|/\|\mathbf{B}\|$ in (2.10) is very close to 0. Moreover, it is clear that $\mathbf{B}$ is a positive definite matrix which is invertible, so $\mathrm{Cond}(\mathbf{B})$ will not be overly large. Therefore, the value of the right-hand side of (2.9) is small, which suggests that the performance of LPDGL is not sensitive to the choice of γ.

2.1.2 Sensitivity of β

Suppose a small bias $\delta\beta$ is added to β, then $\delta\mathbf{B}$ in (2.9) is $\delta\mathbf{B} = \delta\beta\mathbf{L}$. Therefore, we compute the value of $\|\delta\mathbf{B}\|/\|\mathbf{B}\|$, and obtain

$$\begin{aligned}\frac{\|\delta\mathbf{B}\|}{\|\mathbf{B}\|} &= \frac{\delta\beta\,\|\mathbf{L}\|}{\|\mathbf{J}+\beta\mathbf{L}+\gamma\,(\mathbf{I}-\mathbf{D}/v)\|} \\ &= \frac{\delta\beta\sqrt{\sum_{i=1}^{n} d_{ii}^2 + \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\omega_{ij}^2}}{\sqrt{\beta^2\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\omega_{ij}^2 + \sum_{i=1}^{n}\left[\beta d_{ii}+\gamma\,(1-d_{ii}/v\,)\right]^2+\xi}} \\ &= \frac{\delta\beta\sqrt{\sum_{i=1}^{n} d_{ii}^2 + \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\omega_{ij}^2}}{\sqrt{\gamma^2\sum_{i}\left(1-\frac{d_{ii}}{v}\right)^2+\gamma\sum_{i}\left\{\left(1-\frac{d_{ii}}{v}\right)\left[(2\beta-\frac{\gamma}{v})d_{ii}+\gamma\right]\right\}+\xi+\beta^2\left(\sum_{i} d_{ii}^2+\sum_{i}\sum_{j}\omega_{ij}^2\right)}}.\end{aligned} \tag{2.11}$$

Similar to (2.10), the numerator in (2.11) is very small compared with the denominator, so $\|\delta\mathbf{B}\|/\|\mathbf{B}\|$ is very close to 0. As a result, according to (2.9) we know that $\|\delta\mathbf{f}\|$ is negligible in the presence of $\|\mathbf{f}\|$, which indicates that the result of LPDGL is also very robust to the variation of β.

2.2 Induction In RKHS

Note that $\mathbf{f} = (f_1, f_2, \cdots, f_n)^\top$ in (2.7) only encodes the soft labels of examples that are used to construct the graph $\mathcal{G}$ during the training phase, so it cannot predict the labels of test examples that are unseen in the training phase. Therefore, this section adapts the proposed LPDGL to inductive settings, which requires the decision function f trained on $\Psi = \mathcal{L} \cup \mathcal{U}$ to perfectly handle the out-of-sample data, and the predicted label for example $\mathbf{x}$ is $f(\mathbf{x}) \in \mathbb{R}$.

In this work, we build the LPDGL model for prediction in the reproducing kernel Hilbert space (RKHS). An RKHS $\mathcal{H}_K$ is a Hilbert space $\mathcal{H}$ of functions on a set $\mathcal{X}$ with the property that for all $x \in \mathcal{X}$ and $f \in \mathcal{H}$, the point evaluations $f \to f(x)$ are continuous linear functionals Hofmann et al. [2005]. The Moore-Aronszajn theorem Aronszajn [1950] indicates that for every RKHS, there exists a unique positive definite kernel on $\mathcal{X} \times \mathcal{X}$. Therefore, by adopting the Riesz representation theorem, the unique reproducing kernel can always be constructed as $K(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which has an

important property that $\forall x_1, x_2 \in \mathcal{X}$, $K(x_1, x_2) = \langle K(\cdot, x_1), K(\cdot, x_2) \rangle_{\mathcal{H}}$, from the point evaluation functional.

Suppose $K(\cdot, \cdot)$ is a Mercer kernel associated with RKHS, and the corresponding norm is $\|\cdot\|_{\mathcal{H}}$, then we have the following regularization framework of LPDGL defined in RKHS:

$$\min_{f \in \mathcal{H}_K} Q(f) = \frac{1}{2} \left[\alpha \|f\|_{\mathcal{H}}^2 + \beta \mathbf{f}^\top \mathbf{L} \mathbf{f} + \gamma \mathbf{f}^\top (\mathbf{I} - \mathbf{D}/v) \mathbf{f} + \sum\nolimits_{i=1}^{l} \left(f(\mathbf{x}_i) - y_i \right)^2 \right]. \tag{2.12}$$

In (2.12), the trade-off among the four terms is captured by three non-negative parameters α, β and γ. Compared with the expression (2.5) for transduction, Eq. (2.12) contains one more induction term $\|f\|_{\mathcal{H}}^2$ that controls the complexity of f. This term enhances the generalizability of LPDGL by effectively preventing the overfitting problem.

The extended representer theorem Belkin et al. [2006] states that the minimizer of (2.12) can be decomposed as an expansion of kernel functions over both labeled and unlabeled examples, *i.e.*,

$$f(\mathbf{x}) = \sum\nolimits_{i=1}^{n} s_i K(\mathbf{x}, \mathbf{x}_i). \tag{2.13}$$

Therefore, by plugging (2.13) into (2.12) we obtain a novel objective function with respect to $\mathbf{S} = (s_1, \cdots, s_n)^\top$:

$$\min_{\mathbf{S} \in \mathbb{R}^n} \widetilde{Q}(\mathbf{S}) = \frac{1}{2} \left[\alpha \mathbf{S}^\top \mathbf{K} \mathbf{S} + \beta \mathbf{S}^\top \mathbf{K} \mathbf{L} \mathbf{K} \mathbf{S} + \gamma \mathbf{S}^\top \mathbf{K} (\mathbf{I} - \mathbf{D}/v) \mathbf{K} \mathbf{S} + \|\mathbf{y} - \mathbf{J} \mathbf{K} \mathbf{S}\|^2 \right], \tag{2.14}$$

where $\mathbf{K}$ is an $n \times n$ Gram matrix over all the training examples, with elements $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ for $1 \leq i, j \leq n$. It can be easily proved that the objective function in (2.14) is convex, so we can find the globally optimal $\mathbf{S}$ by calculating the derivative of $\widetilde{Q}(\mathbf{S})$ to $\mathbf{S}$, and then setting the result to $\mathbf{0}$, which is expressed as

$$\mathbf{S} = [\alpha \mathbf{I} + \beta \mathbf{L} \mathbf{K} + \gamma (\mathbf{I} - \mathbf{D}/v) \mathbf{K} + \mathbf{J} \mathbf{K}]^{-1} \mathbf{y}. \tag{2.15}$$

Finally, we substitute (2.15) into (2.13), and obtain the function f for predicting the label of $\mathbf{x}$.

2.2.1 Robustness Analysis

Robustness is a desirable property for a learning algorithm because it reflects the sensitivity of the algorithm to the disturbances of training data. Xu and Mannor [2012] state that an algorithm is robust if its solution achieves "similar" performances on a test set and a training set that are "close". Based on the notion introduced by Xu and Mannor [2012], this section studies the robustness of LPDGL. The whole sample space

is represented by $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the input example space and $\mathcal{Y}$ is the output label space. Furthermore, we use $z_i = (\mathbf{x}_i, y_i) \in \mathcal{Z}$ to denote the example-label pair, where $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y} = \{-1, 1\}$. Therefore, the task of LPDGL is to learn a function $f : \mathcal{X} \to \mathcal{Y}$ that maps the elements in the input space $\mathcal{X}$ to the output space $\mathcal{Y}$.

Definition 2.3. ***(covering number, Xu and Mannor [2012])*** *For a metric space S_ρ with a metric ρ, where T, $\hat{T} \subset S_\rho$ are two sets in S_ρ, we say that $\hat{T}$ is an ε-cover of T, if $\forall t \in T$, $\exists \hat{t} \in \hat{T}$, such that $\rho(t, \hat{t}) \leq \varepsilon$. The ε-covering number of T is*

$$N(\varepsilon, T, \rho) = \min\{\left|\hat{T}\right| : \hat{T} \text{ is an } \varepsilon - \text{cover of } T\}. \tag{2.16}$$

Definition 2.4. ***(robustness, Xu and Mannor [2012])*** *Let Ψ, $L(\cdot)$ denote the training set and loss function of an algorithm $\mathcal{A}$, respectively, then $\mathcal{A}$ is $(\theta, \varepsilon(\Psi))$-robust if $\mathcal{Z}$ can be partitioned into θ disjoint sets, denoted as $\{C_i\}_{i=1}^{\theta}$, such that $\forall \mathbf{x}_1, \mathbf{x}_2 \in \Psi$,*

$$z_1, z_2 \in C_i \Rightarrow \ |L(\mathcal{A}_\Psi, z_1) - L(\mathcal{A}_\Psi, z_2)| \leq \varepsilon(\Psi). \tag{2.17}$$

Based on the Definitions 2.3 and 2.4 provided above, we have the following theorem:

Theorem 2.5. *Let $\mathcal{X}$ denote the input space, and $\forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, $\|\mathbf{x}_i - \mathbf{x}_j\| \leq \varepsilon$. A k-NN graph is built with the edge weights represented by Gaussian kernel $\omega_{ij} = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2)\right)$. Under $N(\varepsilon/2, \mathcal{X}, \|\cdot\|_2) < \infty$, the proposed LPDGL is $\sqrt{\frac{8l}{\alpha}}\left(1 + \sqrt{\frac{l}{\alpha}}\right)\sqrt{1 - \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)}$-robust.*

Proof. Suppose $\mathbf{S}$ in (2.14) is set to $\mathbf{S}_0 = (0, \cdots, 0)^\top$, then we have $\widetilde{Q}(\mathbf{S}_0) = \|\mathbf{y}\|^2/2 = l/2$. Moreover, note that all the terms in the bracket in (2.14) are non-negative, so we obtain $\frac{1}{2}\alpha\mathbf{S}^\top\mathbf{K}\mathbf{S} \leq Q(\mathbf{S}) \leq Q(\mathbf{S}_0) = l/2$, which reveals that

$$\mathbf{S}^\top\mathbf{K}\mathbf{S} \leq l/\alpha. \tag{2.18}$$

For binary classification, we can partition $\mathcal{Z}$ into $\theta = 2N(\varepsilon/2, \mathcal{X}, \|\cdot\|_2)$ disjoint sets with a margin ε Xu and Mannor [2012]. Therefore, according to Definition 2.3 we know that if z_1 and z_2 belong to the same set C_i ($1 \leq i \leq \theta$), then $\|\mathbf{x}_1 - \mathbf{x}_2\| \leq \varepsilon$ and $\|y_1 - y_2\| = 0$ Xu and Mannor [2012]. We also know that the loss function of LPDGL is

$$L(f) = (f(\mathbf{x}) - y)^2, \tag{2.19}$$

so according to Definition 2.4 the difference between the losses of f on z_1 and z_2 is

$$|L(f, z_1) - L(f, z_2)| = \left|(y_1 - f(\mathbf{x}_1))^2 - (y_2 - f(\mathbf{x}_2))^2\right|. \tag{2.20}$$

By plugging (2.13) into (2.20), we obtain

$$
\begin{aligned}
&|L(f,z_1)-L(f,z_2)| \\
&= \left| \left[y_1 - \sum\nolimits_{i=1}^{n} s_i K(\mathbf{x}_1,\mathbf{x}_i) \right]^2 - \left[y_2 - \sum\nolimits_{i=1}^{n} s_i K(\mathbf{x}_2,\mathbf{x}_i) \right]^2 \right| \\
&\leq \left| y_1 + y_2 - \sum\nolimits_{i=1}^{n} s_i \left(K(\mathbf{x}_1,\mathbf{x}_i) + K(\mathbf{x}_2,\mathbf{x}_i) \right) \right| \\
&\quad \left| y_1 - y_2 - \sum\nolimits_{i=1}^{n} s_i \left(K(\mathbf{x}_1,\mathbf{x}_i) - K(\mathbf{x}_2,\mathbf{x}_i) \right) \right| \\
&= |B_1|\,|B_2|\,,
\end{aligned}
\tag{2.21}
$$

in which $B_1 = y_1 + y_2 - \sum_{i=1}^{n} s_i \left(K(\mathbf{x}_1,\mathbf{x}_i) + K(\mathbf{x}_2,\mathbf{x}_i) \right)$ and $B_2 = y_1 - y_2 - \sum_{i=1}^{n} s_i \left(K(\mathbf{x}_1,\mathbf{x}_i) - K(\mathbf{x}_2,\mathbf{x}_i) \right)$. In the following derivations, we aim to find the upper bounds of $|B_1|$ and $|B_2|$, respectively. It is easy to show that

$$
\begin{aligned}
|B_1| &\leq |y_1| + |y_2| + |f(\mathbf{x}_1) + f(\mathbf{x}_2)| \\
&\leq 2 + 2 \max_{\mathbf{x}\in\{\mathbf{x}_1,\mathbf{x}_2\}} \langle f, K(\mathbf{x},\cdot)\rangle_{\mathcal{H}} \\
&\leq 2 + 2 \max_{\mathbf{x}\in\{\mathbf{x}_1,\mathbf{x}_2\}} \|f\|_{\mathcal{H}} \sqrt{K(\mathbf{x},\cdot)} \\
&\leq 2 + 2 \max_{\mathbf{x}\in\{\mathbf{x}_1,\mathbf{x}_2\}} \left\| \sum\nolimits_{i=1}^{n} s_i K(\mathbf{x}_i,\cdot) \right\|_{\mathcal{H}} \sqrt{K(\mathbf{x},\cdot)} \\
&\leq 2 + 2\sqrt{\left\langle \sum\nolimits_{i=1}^{n} s_i K(\mathbf{x}_i,\cdot), \sum\nolimits_{i=1}^{n} s_j K(\mathbf{x}_j,\cdot) \right\rangle_{\mathcal{H}}} \\
&= 2 + 2\sqrt{\sum\nolimits_{i,j=1}^{n} s_i s_j K(\mathbf{x}_i,\cdot) K(\mathbf{x}_j,\cdot)} \\
&= 2 + 2\sqrt{\sum\nolimits_{i,j=1}^{n} s_i K(\mathbf{x}_i,\mathbf{x}_j) s_j} \\
&= 2 + 2\sqrt{\mathbf{S}^\top \mathbf{K} \mathbf{S}} \\
&\leq 2 + 2\sqrt{\frac{l}{\alpha}},
\end{aligned}
\tag{2.22}
$$

in which the notation $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ denotes the inner product defined in $\mathcal{H}$. Note that in the derivation of (2.22), we employed the reproducing property of RKHS $f(\mathbf{x}) = \langle f, K(\mathbf{x},\cdot)\rangle_{\mathcal{H}}$ in the second line, the Cauchy-Schwarz inequality $\langle f, K(\mathbf{x},\cdot)\rangle_{\mathcal{H}} \leq \|f\|_{\mathcal{H}}\sqrt{K(\mathbf{x},\cdot)}$ in the third line, and the results of (2.18) in the last line.

Moreover, since $\|f\|_{\mathcal{H}}^2 = \mathbf{S}^\top \mathbf{K} \mathbf{S} \leq \frac{l}{\alpha}$, we have $\|f\|_{\mathcal{H}} \leq \sqrt{\frac{l}{\alpha}}$. By further considering

that $\|y_1 - y_2\| = 0$ and $\|\mathbf{x}_1 - \mathbf{x}_2\| \leq \varepsilon$, we immediately obtain

$$\begin{aligned}
|B_2| &= |f(\mathbf{x}_1) - f(\mathbf{x}_2)| \\
&= |\langle f, K(\mathbf{x}_1, \cdot) - K(\mathbf{x}_2, \cdot)\rangle_{\mathcal{H}}| \\
&\leq \|f\|_{\mathcal{H}} \|K(\mathbf{x}_1, \cdot) - K(\mathbf{x}_2, \cdot)\|_{\mathcal{H}} \\
&= \|f\|_{\mathcal{H}} \sqrt{K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2)} \\
&\leq \sqrt{l/\alpha}\sqrt{K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2)} \\
&\leq \sqrt{l/\alpha}\sqrt{2 - 2\exp\left[-\|\mathbf{x}_1 - \mathbf{x}_2\|^2/(2\sigma^2)\right]} \\
&= \sqrt{l/\alpha}\sqrt{2 - 2\exp\left[-\varepsilon^2/(2\sigma^2)\right]}.
\end{aligned} \tag{2.23}$$

Finally, we substitute (2.22) and (2.23) into (2.21), and have

$$|L(f, z_1) - L(f, z_2)| \leq \sqrt{\frac{8l}{\alpha}}\left(1 + \sqrt{\frac{l}{\alpha}}\right)\sqrt{1 - \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)}, \tag{2.24}$$

which indicates that LPDGL is $\left(\theta, \sqrt{\frac{8l}{\alpha}}\left(1 + \sqrt{\frac{l}{\alpha}}\right)\sqrt{1 - \exp\left(\frac{-\varepsilon^2}{2\sigma^2}\right)}\right)$ -robust. □

2.2.2 Generalization Risk

Based on the robustness analysis in Section 2.2.1, we derive the generalization bound for LPDGL. The empirical error $L_{emp}(\mathcal{A}_\Psi)$ is the error of algorithm $\mathcal{A}$ on the training set Ψ. The generalization error $\tilde{L}(\cdot)$ is the expectation of error rate produced by f on the whole sample space $\mathcal{Z}$. Suppose all the examples are i.i.d, and are generated from an unknown distribution P, then the above two errors are defined by $\tilde{L}(\mathcal{A}_\Psi) = E_{\mathbf{x}\sim P}[L(\mathcal{A}_\Psi, \mathbf{x})]$ and $L_{emp}(\mathcal{A}_\Psi) = \frac{1}{n}\sum_{\mathbf{x}_i \in \Psi} L(\mathcal{A}_\Psi, \mathbf{x}_i)$, respectively.

Theorem 2.6. *(generalization bound, Xu and Mannor [2012]): If the training set Ψ consists of n i.i.d. samples, and the algorithm $\mathcal{A}$ is $(\theta, \varepsilon(\Psi))$-robust, then for any $\delta > 0$, with probability at least $1 - \delta$*

$$\left|\tilde{L}(\mathcal{A}_\Psi) - L_{emp}(\mathcal{A}_\Psi)\right| \leq \varepsilon(\Psi) + M\sqrt{\frac{2\theta \ln 2 + 2\ln(1/\delta)}{n}}, \tag{2.25}$$

where M is the upper bound of loss function $L(\cdot, \cdot)$.

According to Theorem 2.6, the generalization bound of inductive LPDGL is provided in Theorem 2.7:

Theorem 2.7. *Let $L(f,\Psi)=(f(\mathbf{x})-y)^2$ be the loss function of LPDGL, than for any $\delta>0$, with probability at least $1-\delta$, the generalization error of LPDGL is*

$$\begin{aligned}\left|\tilde{L}(A_\psi)-L_{emp}(A_\psi)\right| \leq \sqrt{\frac{8l}{\alpha}}\left(1+\sqrt{\frac{l}{\alpha}}\right)\sqrt{1-\exp\left(\frac{-\varepsilon^2}{2\sigma^2}\right)} \\ +2\left(1+\frac{l}{\alpha}\right)\sqrt{\frac{2\theta\ln 2+2\ln(1/\delta)}{n}}.\end{aligned} \tag{2.26}$$

Proof. To obtain the generalization bound of LPDGL, we need to compute $\varepsilon(\Psi)$, θ and M that appear in (2.25). Note that $\varepsilon(\Psi)$ and θ have been already worked out in Section 2.2.1, so our target is to find the upper bound M of the loss function $L(f,\Psi)$. Therefore, we compute

$$\begin{aligned}L(f,\Psi)=(y-f(\mathbf{x}))^2=y^2-2yf(\mathbf{x})+f^2(\mathbf{x}) \\ \leq 2y^2+2f^2(\mathbf{x})\leq 2+2l/\alpha.\end{aligned} \tag{2.27}$$

As a result, the upper bound of the adopted loss function is

$$M=2+2l/\alpha. \tag{2.28}$$

Finally, by putting (2.24) and (2.28) into (2.25), we complete the proof. □

Theorem 2.7 reveals that the our LPDGL has a profound generalizability with convergence rate of order $\mathcal{O}(\sqrt{\frac{1}{n}})$, which means that the more training examples are available, the lower generalization bound of LPDGL we have, so LPDGL can predict the label of a test example reliably.

2.2.3 Linearization of Kernelized LPDGL

Although the inductive LPDGL is designed in RKHS, its linear counterpart can be easily derived by using the linear prediction function $f(\mathbf{x})=\omega^T\mathbf{x}$, and then substitute $\mathbf{f}=\mathbf{X}^T\omega$ into (2.5), where ω is the weight vector to be optimized and $\mathbf{X}=(\mathbf{x}_1,\mathbf{x}_2,\cdots\mathbf{x}_n)$ is the data matrix with each column representing an example. Next we show that LPDGL in RKHS includes the linear LPDGL as a special case, and in particular, LPDGL in RKHS degenerates to the linear LPDGL when the linear kernel $\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^\top\mathbf{x}_j$ is adopted.

LPDGL in RKHS is Eq. (2.14), and the corresponding minimizer is Eq. (2.15). According to the representer theorem Eq. (2.13), the soft label of a test example $\mathbf{x}_0$ is given by

$$f_1(\mathbf{x}_0)=\bar{\mathbf{k}}\mathbf{S}=\bar{\mathbf{k}}\left[\alpha\mathbf{I}+\beta\mathbf{L}\mathbf{K}+\gamma(\mathbf{I}-\frac{1}{v}\mathbf{D})\mathbf{K}+\mathbf{J}\mathbf{K}\right]^{-1}\mathbf{y}, \tag{2.29}$$

where $\mathbf{K}$ is an $n \times n$ kernel matrix with elements $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, and $\bar{\mathbf{k}} = (K(\mathbf{x}_0, \mathbf{x}_1), K(\mathbf{x}_0, \mathbf{x}_2), \cdots, K(\mathbf{x}_0, \mathbf{x}_n))$ is an n-dimensional row vector.

Given the linear prediction function $f(\mathbf{x}_0) = \omega^\top \mathbf{x}_0$ and the data matrix $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_n)$, the linear LPDGL is given by

$$\begin{aligned} &\min_{\omega} \ Q(\omega) \\ &= \frac{1}{2}\left[\alpha\|\omega\|^2 + \beta\omega^\top \mathbf{X}\mathbf{L}\mathbf{X}^\top\omega + \gamma\omega^\top \mathbf{X}(\mathbf{I} - \frac{1}{v}\mathbf{D})\mathbf{X}^\top\omega + \left\|\mathbf{y} - \mathbf{J}\mathbf{X}^\top\omega\right\|_2^2\right], \end{aligned} \tag{2.30}$$

of which the optimal solution is

$$\omega = \left[\alpha\mathbf{I} + \beta\mathbf{X}\mathbf{L}\mathbf{X}^\top + \gamma\mathbf{X}\left(\mathbf{I} - \frac{1}{v}\mathbf{D}\right)\mathbf{X}^\top + \mathbf{X}\mathbf{J}\mathbf{X}^\top\right]^{-1}\mathbf{X}\mathbf{y}. \tag{2.31}$$

Then, the soft label of a test example $\mathbf{x}_0$ is

$$f_2(\mathbf{x}_0) = \mathbf{x}_0^\top\omega = \mathbf{x}_0^\top\left[\alpha\mathbf{I} + \beta\mathbf{X}\mathbf{L}\mathbf{X}^\top + \gamma\mathbf{X}\left(\mathbf{I} - \frac{1}{v}\mathbf{D}\right)\mathbf{X}^\top + \mathbf{X}\mathbf{J}\mathbf{X}^\top\right]^{-1}\mathbf{X}\mathbf{y}. \tag{2.32}$$

We then need to prove $f_1(\mathbf{x}_0) = f_2(\mathbf{x}_0)$ if the linear kernel $K(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^\top\mathbf{x}_2$ is applied to Eq. (2.29).

Given the linear kernel $K(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^\top\mathbf{x}_2$, it is straightforward that $\mathbf{K} = \mathbf{X}^\top\mathbf{X}$. Then, after plugging $\mathbf{K} = \mathbf{X}^\top\mathbf{X}$ into Eq. (2.29) and denoting $\mathbf{M} = \beta\mathbf{L} + \gamma\left(\mathbf{I} - \frac{1}{v}\mathbf{D}\right) + \mathbf{J}$, we have

$$f_1(\mathbf{x}_0) = \mathbf{x}_0^\top\mathbf{X}\left(\alpha\mathbf{I} + \mathbf{M}\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{y}. \tag{2.33}$$

By using the Woodbury matrix identity Petersen and Pedersen, $f_1(\mathbf{x}_0)$ can be further derived as

$$\begin{aligned} f_1(\mathbf{x}_0) &= \mathbf{x}_0^\top\mathbf{X}\left(\alpha\mathbf{I} + \mathbf{M}\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{y} \\ &= \mathbf{x}_0^\top\mathbf{X}\left[\frac{1}{\alpha}\mathbf{I} - \frac{1}{\alpha^2}\mathbf{M}\left(\mathbf{I} + \frac{1}{\alpha}\mathbf{X}^\top\mathbf{X}\mathbf{M}\right)^{-1}\mathbf{X}^\top\mathbf{X}\right]\mathbf{y} \\ &= \mathbf{x}_0^\top\mathbf{X}\left[\frac{1}{\alpha}\mathbf{I} - \frac{1}{\alpha^2}\left(\mathbf{M}^{-1} + \frac{1}{\alpha}\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{X}\right]\mathbf{y}. \end{aligned} \tag{2.34}$$

Similarly, we rewrite $f_2(\mathbf{x}_0)$ in Eq. (2.32) as

$$\begin{aligned} f_2(\mathbf{x}_0) &= \mathbf{x}_0^\top\left(\alpha\mathbf{I} + \mathbf{X}\mathbf{M}\mathbf{X}^\top\right)^{-1}\mathbf{X}\mathbf{y} \\ &= \mathbf{x}_0^\top\left[\frac{1}{\alpha}\mathbf{I} - \frac{1}{\alpha^2}\mathbf{X}\left(\mathbf{M}^{-1} + \frac{1}{\alpha}\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\right]\mathbf{X}\mathbf{y} \\ &= \mathbf{x}_0^\top\mathbf{X}\left[\frac{1}{\alpha}\mathbf{I} - \frac{1}{\alpha^2}\left(\mathbf{M}^{-1} + \frac{1}{\alpha}\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{X}\right]\mathbf{y}, \end{aligned} \tag{2.35}$$

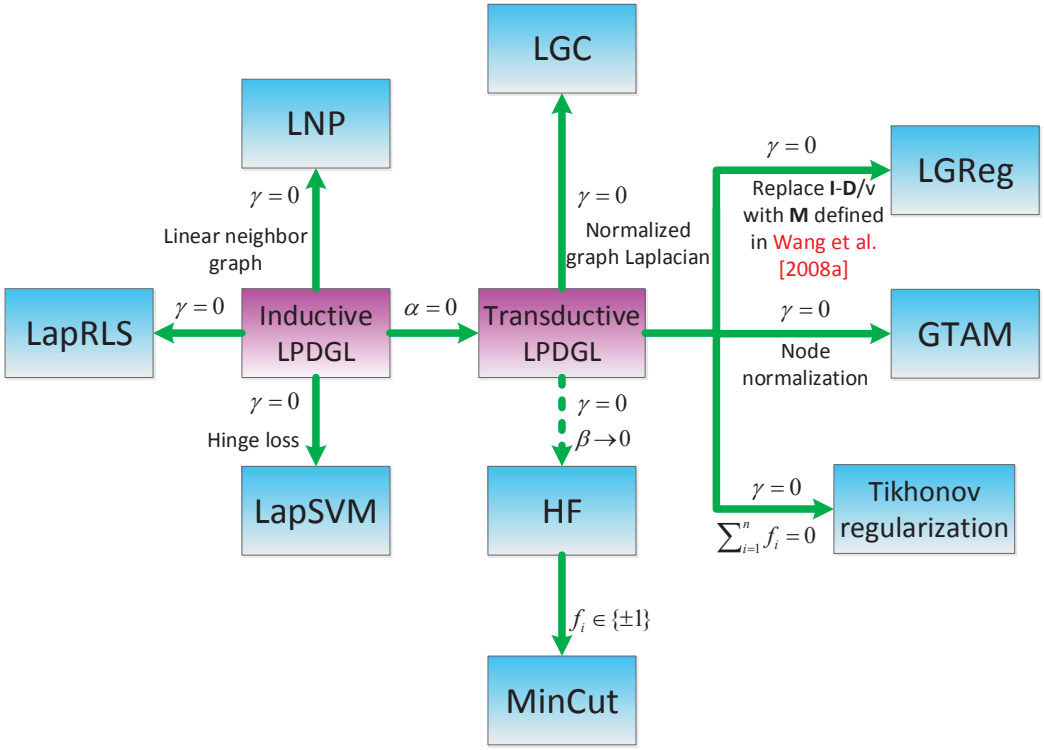

Figure 2.2: The evolutionary process from LPDGL to other typical SSL methods. The dashed line means "infinitely approach to". Note that our LPDGL is located in the central position and other algorithms are derived from LPDGL by satisfying the conditions alongside the arrows.

which is exactly same as Eq. (2.34). Therefore, LPDGL in RKHS with linear kernel is equivalent to linear LPDGL.

2.3 Relationship Between LPDGL and Existing Methods

This section relates the proposed LPDGL to several typical semi-supervised learning (SSL) algorithms. SSL has attracted considerable interest since it was developed. Various SSL algorithms have been proposed for different purposes and applications. As mentioned in the Chapter 1, existing SSL algorithms can be divided into a transductive approach or inductive approach.

Typical transductive methods include Tikhonov regularization Belkin et al. [2004b], harmonic functions (HF, Zhu et al. [2003a]), local and global consistency (LGC, Zhou and Bousquet [2003]), minimum cut (MinCut, Joachims [2003]), local learning

regularization (LLReg) Wu and Schölkopf [2007], local and global regularization (LGReg) Wang et al. [2008a], path-based SSL (PBSSL) Chang and Yeung [2004], transductive SVMs (S3VM, Vapnik [1998]), safe semi-supervised learning (S4VM, Li and Zhou [2011]), AnchorGraph regularization (AGR, Liu et al. [2010]), graph transduction via alternating minimization (GTAM, Wang et al. [2008b]), Laplacian embedded support vector regression (LapESVR, Chen et al. [2012a]), semi-supervised classification based on class membership (SSCCM, Wang et al. [2012]), and safety-aware SSCCM (SA-SSCCM, Wang and Chen [2013]).

Representative inductive SSL algorithms include harmonic mixtures Zhu and Lafferty [2005], Laplacian Support Vector Machines (LapSVM, Belkin et al. [2006]), Laplacian Regularized Least Squares (LapRLS, Belkin et al. [2006]), linear neighborhood propagation (LNP, Wang et al. [2009b]), simple semi-supervised learning (SSSL, Ji et al. [2012]), and vector-valued manifold regularization Quang et al. [2013]. For more detailed explanations about SSL algorithms, the reader is referred to the surveys Zhu and Goldberg [2009] and Chapelle et al. [2006].

All the above methods are formulated as a regularization framework, the same as the proposed LPDGL. The main difference between them is how to design the regularizer. In this sense, most of the above SSL algorithms can be derived from LPDGL by choosing different regularizers or incorporating other constraints, as illustrated in Fig. 2.2. For example, if the local smoothness term of LPDGL is removed (*i.e.* γ is set to 0) and let $\beta \to 0$, the result of LPDGL will get arbitrarily close to HF[1]. If we further require the obtained discrete labels belong to $\{\pm 1\}$, we reach the MinCut algorithm. In addition, LGC can be derived from LPDGL by adopting the normalized graph Laplacian. LGReg, GTAM and Tikhonov regularization can also be easily derived by employing the techniques alongside the arrows. Some inductive algorithms, including LNP, LapRLS and LapSVM are also related to inductive LPDGL. If we set $\gamma = 0$ and adopt the hinge loss instead of the squared loss incorporated by LPDGL, we immediately obtain LapSVM. Similarly, if $\gamma = 0$ and a linear neighborhood graph in Wang et al. [2009b] is constructed, the proposed LPDGL will have the same formation as LNP. Of particular note is that the only difference between LapRLS and inductive LPDGL lies in the local smoothness term, of which the significance for boosting the accuracy will be demonstrated in the experimental section (Section 2.4). Therefore, the proposed LPDGL has a strong relationship with other popular SSL methodologies, and they can be viewed as special cases of LPDGL.

[1]The precise solution of HF can be obtained by further relaxing the HF model derived from LPDGL as $\min_{\mathbf{f}} \mathbf{f}^T \mathbf{L} \mathbf{f}$, s.t. $f_i = y_i$ for $i = 1, 2, \cdots, l$. The detailed relaxation process is referred to Zhu and Goldberg [2009].

2.4 Experiments

In this section, we validate the proposed LPDGL on several synthetic toy datasets, and compare LPDGL with some state-of-the-art graph transduction algorithms on a number of real-world collections. HF Zhu et al. [2003a], LGC Zhou and Bousquet [2003], AGR Liu et al. [2010], LNP Wang et al. [2009b], LapRLS Belkin et al. [2006], LapSVM Belkin et al. [2006], LLReg Wu and Schölkopf [2007], PBSSL Chang and Yeung [2004], S4VM (RBF kernel) Li and Zhou [2011] and S4VM (linear kernel) Li and Zhou [2011] were adopted as baselines to evaluate the transductive ability of LPDGL. LNP Wang et al. [2009b], LapRLS Belkin et al. [2006] and LapSVM Belkin et al. [2006] were used for the inductive performance comparison because other algorithms do not have inductive ability. For fair comparison, HF, LGC, LLReg, PBSSL, LapRLS, LapSVM and LPDGL were trained by the same k-NN graph[1] for each of the datasets, and all the algorithms were conducted 10 times independently under each l (l represents the size of the labeled set) with randomly selected labeled set $\mathcal{L}$. However, at least one labeled example was selected in each class when $\mathcal{L}$ was generated. The reported accuracies and standard deviations of algorithms were calculated as the mean value of the outputs of these runs. To demonstrate the superiority of the proposed LPDGL over linear LPDGL mentioned in Section 2.2, we also compared the performances of these two models on various datasets.

2.4.1 Toy Data

Synthetic 2D and 3D data was adopted in this section to visualize the transductive and inductive performance of LPDGL.

2.4.1.1 Transduction on 3D Data

Two 3D datasets, *Cylinder&Ring* and *Knot*, were used to test the transductive ability of LPDGL. The *Cylinder&Ring* dataset (see Figs. 2.3 (a), (b) and (c)) forms like a cylinder surrounded by a ring, in which the cylinder with radius 0.2 represents the positive class and the ring with radius 0.8 constitutes the negative class. The *Knot* dataset is shaped like a knot composed of two crossing rings with radiuses 0.8, and each ring represents a class (see Figs. 2.3 (d), (e) and (f)). Both datasets are contaminated by the Gaussian noise of variance 0.1, and each class only has one labeled example, as shown in Figs. 2.3 (a) and (d).

We adopted Eq. (2.5) and the linear model $f(\mathbf{x}) = \omega^\top \mathbf{x}$, respectively, to train a transductive LPDGL to classify all the examples, given very few labeled examples.

[1] AGR builds a hyper-graph that is different from other algorithms. The k-NN graph in LNP is not symmetrical, which is different from that in HF, LGC, LLReg, PBSSL, LapRLS, LapSVM and LPDGL. S3VM and S4VM are not graph-based methods, so graph is not needed to train the classifier.

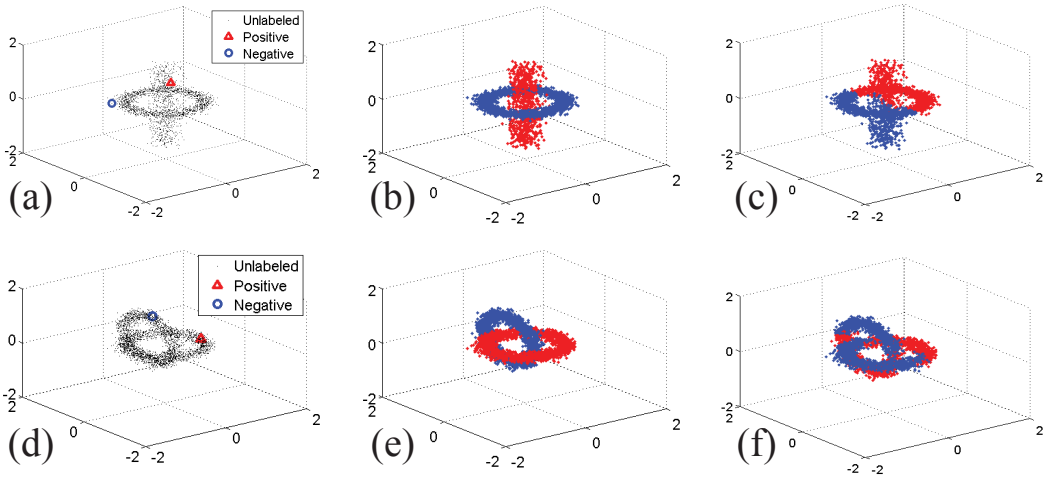

Figure 2.3: Transduction on two 3D datasets: (a) and (d) show the initial states of *Cylinder&Ring* and *Knot*, respectively, in which the red triangle denotes a positive example and the blue circle represents a negative example. (b) and (e) are the transduction results of developed LPDGL on these two datasets. (c) and (f) present the results of LPDGL (Linear).

The parameters in LPDGL were $\sigma = 2$, $k = 5$, $\beta = 1$, $\gamma = 0.001$ for *Cylinder&Ring*, and $\sigma = 0.5$, $k = 5$, $\beta = 1$, $\gamma = 1$ for *Knot*. From Figs 2.3 (b) and (e), we observe that LPDGL can effectively detect the geometric structure of the data distribution, which leads to encouraging performances on both synthetic datasets. Therefore, the proposed algorithm has a satisfactory transductive ability. Comparatively, the LPDGL (Linear) generates disastrous results (see Figs. 2.3 (c) and (f)) because both datasets are highly non-linear.

2.4.1.2 Visualization of Generalizability

LPDGL can not only handle the transductive problems, but also shows great potential for dealing with inductive tasks. The *DoubleMoon* dataset contains 400 examples which are equally divided into two moons centered at $(0, 0)$ and $(10, 0)$, respectively. Each moon represents a class. The data distribution is displayed in Fig 2.4 (a), in which the labeled examples are marked in color. In *Square&Ring*, a square centered at $(0.5, 0.5)$ is surrounded by a ring with the same center. The radius of the outer ring is 1.3, and the length of each side of the inner square is 1 (see Fig. 2.4 (d)).

In these two datasets, only one labeled example was selected for each class. The training set Ψ was made up of these few labeled examples and the abundant unlabeled examples, based on which Eq. (2.14) was utilized to train an inductive LPDGL. In LPDGL, we set $\sigma = 5$ and $\alpha = \beta = \gamma = 1$ for both datasets, and established the 9-NN

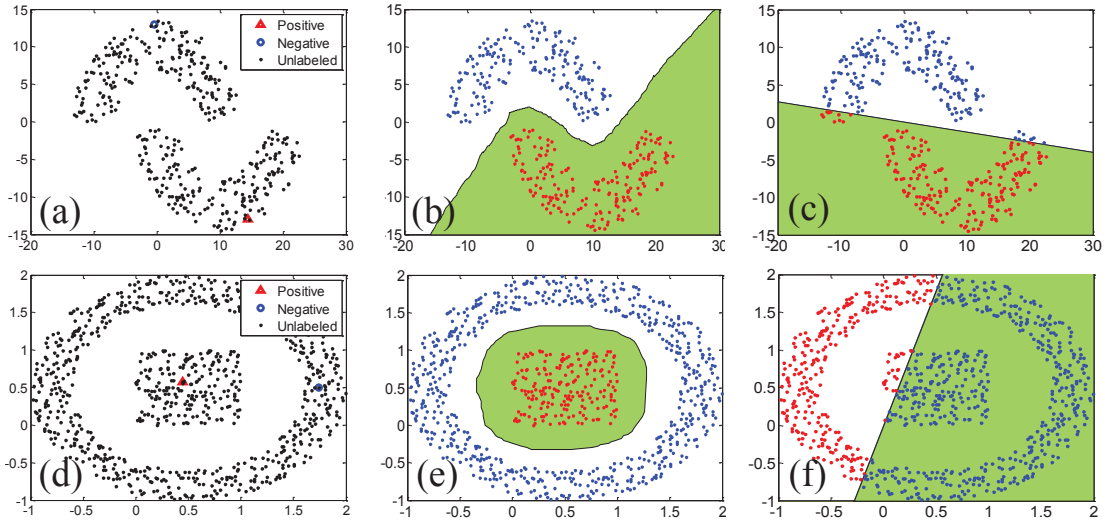

Figure 2.4: Induction on *DoubleMoon* and *Square&Ring* datasets. (a) and (d) show the initial states with the marked labeled examples. (b) and (e) are induction results, in which the decision boundaries are plotted. (c) and (f) are induction performances produced by LPDGL (Linear).

and 7-NN graphs for *DoubleMoon* and *Square&Ring*, respectively. Figs. 2.4 (b) and (e) reveal that the white and green regions partitioned by the learned decision boundary are consistent with the geometry of the training examples. Consequently, the proposed LPDGL correctly classifies all the training examples, and good generalizability is also guaranteed.

Besides, we provide the empirical illustrations on both synthetic datasets to show that the inductive LPDGL derived in RKHS performs better than the linear LPDGL. Figs. 2.4 (c) and (f) present the transductive results and the decision boundaries on each dataset, which clearly reveal that the linear function $f(\mathbf{x}) = \omega^\top \mathbf{x}$ cannot obtain as good performance as the LPDGL in RKHS for nonlinear datasets. Therefore, we suggest using Eq. (2.7) to implement transduction, and adopting Eq. (2.13) for induction.

2.4.2 Real Benchmark Data

This section compares the transductive accuracy of LPDGL with the results reported in Chapelle et al. [2006] on six real benchmark datasets, including *USPS_Imbalanced*, *BCI*, *g241c*, *g241d*, *Digit1*, and *COIL*. The detailed information about these datasets and the performances of different algorithms are provided in Chapelle et al. [2006].

All the algorithms are implemented under $l = 10$ and $l = 100$ for each data set, and the reported accuracies are the mean values of the outputs of 12 independent runs.

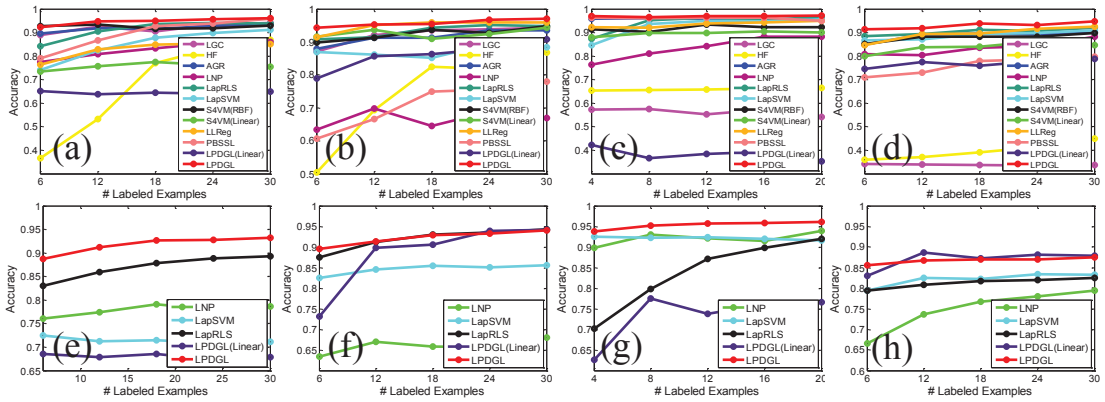

Figure 2.5: Experimental results on four UCI datasets. (a) and (e) are *Iris*, (b) and (f) are *Wine*, (c) and (g) are *BreastCancer*, and (d) and (h) are *Seeds*. The sub-plots in the first row compare the transductive performance of the algorithms, and the sub-plots in the second row compare their inductive performance.

In each run, the labeled and unlabeled examples are randomly generated. However, the 12 different partitions of labeled and unlabeled sets in each dataset are identical for all the compared algorithms. The parameters of LPDGL are optimally tuned to obtain the best performance. We set the number of neighbors of every data point $k = 10, 8, 40, 30, 20, 7$ for *USPS_Imbalanced*, *BCI*, *g241c*, *g241d*, *Digit1*, and *COIL*, respectively, and the widths of RBF kernel are $\sigma = 2, 1, 1, 1, 5, 2$ correspondingly. Table 2.1 shows the error rates of different algorithms, which reveals that LPDGL achieves comparable performances with the typical state-of-the-art algorithms. Specifically, we observe that the linear LPDGL is outperformed by non-linear model in all the datasets except *BCI*. We also want to mention that the relative size of positive and negative classes in *USPS_Imbalanced* is 1:4, so the experimental results on *USPS_Imbalanced* demonstrate that LPDGL can perfectly handle the situations when the examples of different classes are imbalanced.

2.4.3 UCI Data

We chose four UCI Machine Learning Repository datasets Frank and Asuncion [2010], *Iris*, *Wine*, *BreastCancer* and *Seeds*, to compare the performance of LPDGL with other baselines. The detailed information of the four datasets is summarized in Table 2.2. Throughout this chapter, we adopt the "one-versus-rest" strategy to deal with multi-class classifications.

We first evaluated the transductive abilities of HF, LGC, AGR, LNP, LLReg, PBSSL, LapRLS, LapSVM, S4VM and LPDGL by observing the classification

Table 2.1: Experimental results on the benchmark datasets for the variety of transduction algorithms. (The values in the table represent the error rate (%). The best three results for each dataset are marked in red, and blue, and green, respectively.)

Datasets	*USPS_Imbalanced*		*BCI*		*g241c*		*g241d*		*Digit1*		*COIL*	
$l(\#Labeled\ Examples)$	10	100	10	100	10	100	10	100	10	100	10	100
1NN	16.66	5.81	49.00	48.67	47.88	43.93	46.72	42.45	13.65	3.89	63.36	17.35
SVM	20.03	9.75	49.85	34.31	47.32	23.11	46.66	24.64	30.60	5.53	68.36	22.93
MVU+1NN	23.34	6.50	47.95	47.89	47.15	43.01	45.56	38.20	14.42	2.83	62.62	28.71
LEM+1NN	19.82	7.64	48.74	44.83	44.05	40.28	43.22	37.49	23.47	6.12	65.91	23.27
QC+CMN	13.61	6.36	50.36	46.22	39.96	22.05	46.55	28.20	9.80	3.15	59.63	10.03
Discrete Reg.	16.07	4.68	49.51	47.67	49.59	43.65	49.05	41.65	12.64	2.77	63.38	9.61
TSVM	25.20	9.77	49.15	33.25	24.71	18.46	50.08	22.42	17.77	6.15	67.50	25.80
SGT	25.36	6.80	49.59	45.03	22.76	17.41	18.64	9.11	8.92	2.61	-	-
Cluster-Kernel	19.41	9.68	48.31	35.17	48.28	13.49	42.05	4.95	18.73	3.79	67.32	21.99
Data-Dep. Reg.	17.96	5.10	50.21	47.47	41.25	20.31	45.89	32.82	12.49	2.44	63.65	11.46
LDS	17.57	4.96	49.27	43.97	28.85	18.04	50.63	23.74	15.63	3.46	61.90	13.72
Laplacian RLS	18.99	4.68	48.97	31.36	43.95	24.36	45.68	26.46	5.44	2.92	54.54	11.92
CHM (normed)	20.53	7.65	46.90	36.03	39.03	24.82	43.01	25.67	14.86	3.79	-	-
LPDGL(Linear)	19.77	13.44	43.50	24.90	44.15	34.04	45.11	33.62	38.11	10.19	73.21	70.64
LPDGL	17.88	5.09	48.17	34.52	42.73	21.54	42.01	23.90	5.37	2.23	61.69	7.27

Table 2.2: Summary of four UCI datasets

	Iris	*Wine*	*BreastCancer*	*Seeds*
#Instances	150	178	683	210
#Attributes	4	13	10	7
#Classes	3	3	2	3

accuracies with respect to different l for each dataset. The reported results are averaged over the outputs of 10 independent runs under each l. In AGR, we chose the number of anchor points $s = 40, 40, 30, 50$ for *Iris*, *Wine*, *BreastCancer* and *Seeds* datasets, respectively, and 5-NN graphs were constructed on these anchor points. The regression matrix in AGR was established by Local Anchor Embedding (LAE), which was recommended by the authors Liu et al. [2010]. The parameter λ in LLReg was set to 1 for all the UCI datasets. In *Iris*, *Wine*, *BreastCancer* and *Seeds*, we constructed identical 8-NN, 6-NN, 7-NN and 9-NN graphs correspondingly for HF, LGC, LLReg, PBSSL, LapRLS, LapSVM and LPDGL. Parameters β and γ in LPDGL were set to 1, and two simulations of S4VM with RBF kernel and linear kernel were conducted on the four UCI datasets. The transductive results are presented in Figs. 2.5 (a)-(d). We observe that some of the baselines achieve very encouraging performances on these datasets, *e.g.* LGC on *Iris*, S4VM on *Wine*, and AGR on *BreastCancer*, etc. However, the accuracies obtained by these baselines can still be improved by the proposed LPDGL, which demonstrates the strength of our algorithm.

To test inductive ability, we adopted LNP, LapRLS and LapSVM as baselines because they are state-of-the-art inductive algorithms. We not only compared the classification accuracies of LPDGL and other baselines, but also used the 5×2 cross-validation F-test (5×2 cv F-test) proposed by Alpaydin [1999] to make statistical comparisons. The F-statistics value produced by the 5×2 cv F-test is to identify whether two algorithms achieve the same performance on the test set. The null hypothesis is that they do obtain the same test accuracy, and we reject this hypothesis with 95% confidence if the F-statistics value is greater than 4.74. For conducting the 5×2 cv F-test, five replications of twofold cross-validation were performed, and the four datasets were equally split randomly into training and test sets in each replication; however, the splits in the five replications were identical for all the compared algorithms. In the training and test sets, the number of examples belonging to a certain class is proportional to the number of examples of this class in the entire dataset. Given $m_i^{(j)}$ as the difference of error rates generated by two algorithms on fold j ($j=1,2$) of replication i ($i = 1, 2, \cdots, 5$), then the mean error rate and the variance of replication i are $\bar{m}_i = (m_i^{(1)} + m_i^{(2)})/2$ and $s_i^2 = (m_i^{(1)} - \bar{m}_i)^2 + (m_i^{(2)} - \bar{m}_i)^2$, respectively.

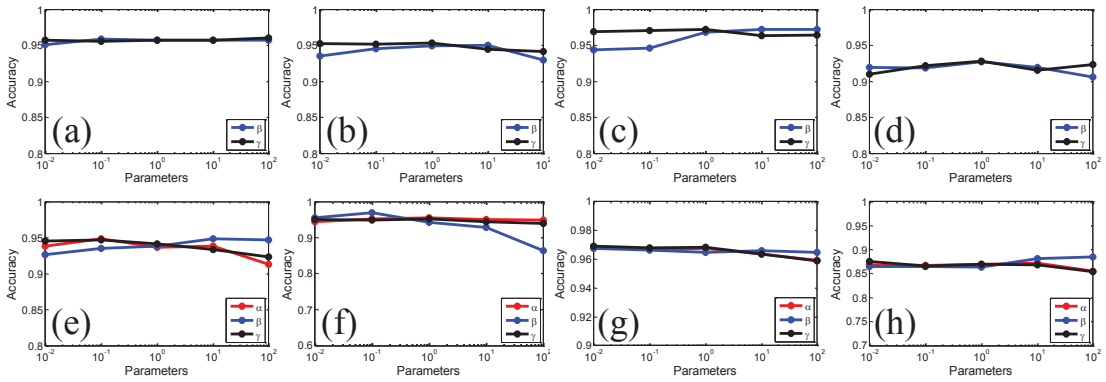

Figure 2.6: Empirical studies on the parametric sensitivity of LPDGL. (a) and (e) are *Iris*, (b) and (f) are *Wine*, (c) and (g) are *BreastCancer*, and (d) and (h) are *Seeds*. The sub-plots in the first row show the transductive results, and the sub-plots in the second row display the inductive results.

Therefore, according to Alpaydin [1999], the F-statistics value $F = \frac{\sum_{i=1}^{5}\sum_{j=1}^{2}\left(m_i^{(j)}\right)^2}{2\sum_{i=1}^{5} s_i^2}$ obeys the F-distribution with 10 and 5 degrees of freedom.

In the four datasets, the established graphs for induction were the same as those for transduction. The weight α of the inductive term in Eq. (2.14) was set to 1 on all UCI datasets, and the parameters in LapRLS and LapSVM were also tuned properly to achieve the best performance. We reported the test accuracies as the mean outputs of 5 replications of twofold cross-validation in every dataset, and they are plotted in Figs. 2.5 (e)-(h). We observe that LPDGL outperforms LNP, LapRLS and LapSVM significantly on the UCI datasets with the exception of *Wine*. On the *Wine* dataset, LPDGL achieves comparable performance with LapRLS. The F-statistics values of baselines versus LPDGL are listed in Table 2.3. The acceptable cases are marked in color, which means that the performance of the two algorithms is comparable. Note that the null hypothesis is rejected in most cases, so the superiority of LPDGL to the compared algorithms is statistically demonstrated. However, the null hypothesis is accepted on *Wine* for LapRLS, because there is no significant difference between the error rates of LPDGL and LapRLS, as revealed by Fig. 2.5 (f), so the performances of the two algorithms on the *Wine* dataset are considered to be essentially identical.

We studied the parametric sensitivity for both transductive and inductive tasks in particular. We observed accuracies under $l = 30$ on *Iris*, *Wine*, *Seeds*, and $l = 20$ on *BreastCancer*, with a wide choice of parameters α, β and γ (see Fig. 2.6). Recall that we have theoretically proved that the transductive performance of LPDGL is very robust to the variations of β and γ. Here, we empirically verify this point by examining the accuracies with one parameter changed and the other fixed to 1. Figs. 2.6 (a)-(d)

Table 2.3: F-statistics values of inductive algorithms versus LPDGL on UCI datasets. (The records smaller than 4.74 are marked in red, which mean that the null hypothesis is accepted.)

	l	LNP	LapSVM	LapRLS
Iris	6	16.69	26.31	6.53
	12	17.23	160.72	19.69
	18	8.0	132.11	28.17
	24	9.7	120.23	30.68
	30	5.5	343.85	19.01
Wine	6	13.87	23.12	2.94
	12	12.70	14.81	1.94
	18	19.18	49.67	2.67
	24	12.89	22.38	1.13
	30	11.25	43.12	1.74
BreastCancer	4	4.91	2.52	999.31
	8	1.81	11.99	384.48
	12	6.08	24.06	121.20
	16	2.40	24.51	23.79
	20	4.03	16.57	24.21
Seeds	6	20.85	36.36	8.80
	12	19.62	7.03	24.95
	18	32.18	31.78	52.62
	24	7.61	7.56	73.38
	30	8.76	12.01	33.81

suggest that these two parameters have little impact on the transductive performance, which is consistent with our theoretical understanding in Sections 2.1.1 and 2.1.2. The parametric sensitivity under the inductive case was also investigated by varying one of α, β and γ, and fixing the remaining two parameters to 1. Figs. 2.6 (e)-(h) reveal that although the parameters cover a wide range, *i.e.* 10^{-2}-10^{2}, the accuracy remains substantially unchanged on the four datasets. Therefore, we conclude that LPDGL shows profound parametric sensitivity for both transductive and inductive settings.

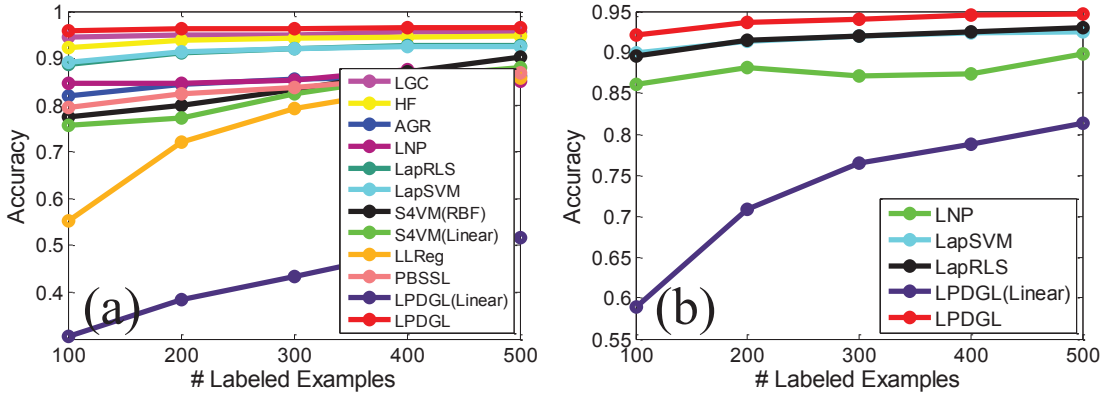

Figure 2.7: Experimental results on *USPS* dataset. (a) shows the transductive results, and (b) shows the inductive results.

2.4.4 Handwritten Digit Recognition

The *USPS* dataset[1] was adopted to assess the ability of algorithms to recognize handwritten digits. This dataset contains 9298 digit images belonging to 10 classes, *i.e.* digits 0-9. The resolution of all images is 16×16, so the pixel-wise feature we adopted was 256 dimensions, in which every dimension represents the gray value of corresponding pixels.

We used the whole dataset to test the transductive performances of various algorithms. A number of the examples were selected as a labeled set, and the rest were taken as the unlabeled examples. The classification accuracies were particularly observed when l changed from 100 to 500. The 10-NN graph was established, and the parameter σ for computing the edge weights was set to 5. For AGR, 300 anchor points were automatically generated by K-means clustering, and a 7-NN graph was constructed on these anchor points. In addition, β and γ in LPDGL were tuned to 10 and 1, and we used the default parameter settings in S4VM.

Fig. 2.7 (a) shows the transductive results. It is observed that LPDGL achieves higher classification accuracy than other baselines. By comparison, LGC and HF achieve slight lower accuracies than LPDGL, *i.e.* 95% approximately. LapSVM and LapRLS achieve similar results. The performances of LNP, LLReg, PBSSL, S4VM and AGR are not as satisfactory as the other five methods. Therefore, the proposed LPDGL is very effective on handwritten digit recognition, though only a small number of labeled examples are given.

To test the inductive abilities of algorithms, we split the original dataset into a training set and a test set. 600 examples per class were extracted to form the training

[1] http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html

set, and the remaining 3298 examples served as the test set. The weight α of the inductive term in (2.14) was tuned to 1 to extend LPDGL to out-of-sample data. Fig. 2.7 (b) reveals that LPDGL is superior to LNP, LapRLS and LapSVM in terms of inductive accuracy. Moreover, it can be observed that the inductive performance of LPDGL does not decrease the transductive settings too much. The reason is that LPDGL has successfully discovered the manifold from the training set in advance, so even though the test data are previously unseen, LPDGL can precisely predict their labels according to their locations on the manifold. Therefore, LPDGL achieves similar performances on transductive and inductive settings, which again demonstrates generalizability. Comparatively, the LPDGL (Linear) is significantly outperformed by the kernerlized LPDGL for both transductive and inductive settings, which also shows the strength of the developed non-linear model.

2.4.5 Face Recognition

Face recognition has been widely studied as a traditional research area of computer vision because of the extensive practical demands. LPDGL was performed on two face datasets: *Yale*[1] and *Labeled Face in the Wild (LFW)*[2] Gary et al. [2007]. The face images in *Yale* are collected in a laboratory environment. In contrast, the images in *LFW* are directly downloaded from the web, and faces are presented in natural scenes.

2.4.5.1 *Yale*

The *Yale* face dataset contains 165 grayscale images of 15 individuals. Each individual has 11 face images covering a variety of facial expressions and configurations including: center-light, wearing glasses, happy, left-light, wearing no glasses, normal, right-light, sad, sleepy, surprised, and wink. The resolution of every image is 64×64, so we directly rearranged each image to a 4096-dimensional long vector as input for all the algorithms.

The transductive abilities of LPDGL and other baselines were first evaluated. In this experiment, we chose $\sigma = 10$ and $k = 5$ for graph construction, and other parameters for LPDGL were $\beta = \gamma = 1$. In AGR, a 7-NN graph was built on the 35 anchor points. In LNP, we established a 9-NN graph to achieve the best performance. In LLReg, λ was optimally tuned to 1. The accuracies of algorithms with different l are listed in Table 2.4, in which the best record under each l is marked in red. The proposed LPDGL is able to achieve the highest accuracy, and the small standard deviations suggest that LPDGL is very robust to the choice of labeled examples.

Inductive performance was also studied on the *Yale* dataset. We chose the first 6 examples of every individual to establish the training set, and the other 5 examples

[1] http://cvc.yale.edu/projects/yalefaces/yalefaces.html
[2] http://vis-www.cs.umass.edu/lfw/

Table 2.4: Transductive comparison on *Yale* dataset

	$l = 30$	$l = 60$
LGC	0.66 ± 0.06	0.76 ± 0.02
HF	0.65 ± 0.04	0.79 ± 0.01
AGR	0.50 ± 0.03	0.64 ± 0.02
LNP	0.32 ± 0.05	0.34 ± 0.04
LapRLS	0.63 ± 0.05	0.71 ± 0.03
LapSVM	0.63 ± 0.05	0.72 ± 0.03
S4VM(Linear)	0.27 ± 0.07	0.52 ± 0.06
S4VM(RBF)	0.11 ± 0.02	0.23 ± 0.04
LLReg	0.65 ± 0.08	0.79 ± 0.09
PBSSL	0.51 ± 0.05	0.67 ± 0.02
LPDGL(Linear)	0.65 ± 0.04	0.79 ± 0.01
LPDGL	0.67 ± 0.03	0.81 ± 0.01

Table 2.5: Inductive comparison on *Yale* dataset

	$l = 30$	$l = 60$
LNP	0.10 ± 0.04	0.15 ± 0.05
LapSVM	0.69 ± 0.01	0.77 ± 0.01
LapRLS	0.68 ± 0.01	0.79 ± 0.01
LPDGL(Linear)	0.58 ± 0.06	0.80 ± 0.01
LPDGL	0.69 ± 0.04	0.83 ± 0.03

made up the test set; the sizes of the training and test sets were $6 \times 15 = 90$ and $5 \times 15 = 75$, respectively. Labeled sets of size $l = 30$ and 60 were then randomly generated in the training set. The main difficulty for induction on *Yale* is that the expressions or appearances of test faces are never observed in the training set, which requires the classifiers to be immune to illumination or changes in facial expression. The inductive accuracies are compared in Table 2.5 and suggest that LPDGL outperforms other baselines when l varies from small to large. In particular, LPDGL achieves 83% accuracy when $l = 60$, which is a very encouraging result. It is widely acknowledged that although faces have different expressions or observation angles, they are actually embedded in a potential manifold Roweis and Saul [2000]. Fortunately, LPDGL is exactly developed on the manifold assumption, so it is able to recognize faces accurately even though their appearance differs dramatically. It is

Table 2.6: Transductive comparison on *LFW* dataset

	$l = 50$	$l = 100$	$l = 150$	$l = 200$
LGC	0.50 ± 0.07	0.60 ± 0.05	0.65 ± 0.08	0.69 ± 0.06
HF	0.66 ± 0.03	0.78 ± 0.02	0.83 ± 0.01	0.87 ± 0.01
AGR	0.60 ± 0.03	0.71 ± 0.01	0.76 ± 0.02	0.80 ± 0.01
LNP	0.32 ± 0.07	0.38 ± 0.16	0.57 ± 0.12	0.59 ± 0.11
LapRLS	0.48 ± 0.03	0.62 ± 0.04	0.71 ± 0.03	0.75 ± 0.03
LapSVM	0.57 ± 0.02	0.70 ± 0.03	0.74 ± 0.03	0.76 ± 0.03
S4VM(Linear)	0.56 ± 0.05	0.68 ± 0.03	0.73 ± 0.03	0.77 ± 0.02
S4VM(RBF)	0.45 ± 0.06	0.61 ± 0.02	0.70 ± 0.02	0.73 ± 0.02
LLReg	0.52 ± 0.04	0.69 ± 0.02	0.86 ± 0.02	0.88 ± 0.01
PBSSL	0.33 ± 0.03	0.46 ± 0.02	0.58 ± 0.02	0.68 ± 0.02
LPDGL(Linear)	0.43 ± 0.02	0.59 ± 0.04	0.64 ± 0.02	0.71 ± 0.02
LPDGL	0.71 ± 0.02	0.81 ± 0.02	0.86 ± 0.01	0.90 ± 0.01

also worth pointing out that we have only adopted the simple pixel-wise gray values of images as features. If more high level features are utilized, the performance of LPDGL is expected to be further improved.

2.4.5.2 *LFW*

LFW is a gigantic collection of face images gathered directly from the web. The facial expressions, observation angle, illumination conditions and background setting are not intentionally controlled for recognition, therefore identifying faces in such unconstrained situations is a big challenge. This dataset contains more than 13000 face images, and each face is labeled with the name of the person. The faces in all the images are detected by the Viola-Jones detector Viola and Jones [2001].

Most people in the original *LFW* have fewer than 5 images, which is insufficient for splitting into training and test sets, so we used a subset of *LFW* by choosing persons who have more than 30 face images. We chose the images of politicians Toledo, Sharon, Schwarzenegger, Powell, Rumsfeld, Bush, and Arroyo, and the images of sports stars Agassi, Beckham and Hewitt. There were thus 392 examples belonging to 10 people in total in the subset. We adopted the 73-dimensional feature developed by Kumar et al. [2009], which describes the biometrics traits of visual appearance, such as gender, race, age, and hair color.

To test the transductive performance, a 6-NN graph with $\sigma = 5$ was built on the entire dataset. Other parameters in LPDGL were $\beta = \gamma = 1$. Algorithms are compared in Table 2.6, in which the best performance under each l is marked in color. It is

Table 2.7: Inductive comparison on *LFW* dataset

	$l = 50$	$l = 100$	$l = 150$	$l = 200$
LNP	0.30 ± 0.07	0.38 ± 0.09	0.45 ± 0.13	0.45 ± 0.09
LapSVM	0.65 ± 0.01	0.69 ± 0.03	0.75 ± 0.02	0.76 ± 0.01
LapRLS	0.67 ± 0.04	0.73 ± 0.02	0.78 ± 0.01	0.79 ± 0.01
LPDGL(Linear)	0.68 ± 0.04	0.78 ± 0.03	0.81 ± 0.03	0.83 ± 0.01
LPDGL	0.70 ± 0.03	0.78 ± 0.03	0.80 ± 0.02	0.83 ± 0.02

observed that LPDGL achieves very satisfactory results and significantly outperforms other methods. In particular, the proposed LPDGL obtains very high accuracy under relatively small l, *e.g.* 71% under $l = 50$ and 81% under $l = 100$, which further demonstrates the effectiveness of LPDGL.

Inductive experiments were conducted by separating the dataset into a training set of 250 examples and a test set of 142 examples. The inductive results of algorithms under different l are listed in Table 2.7, from which we find that the proposed LPDGL achieves the best performance compared to other baselines. Table 2.7 also reveals that the accuracy of LPDGL exceeds 80% when l is larger than 150, so LPDGL has strong potential for successful unconstrained face recognition.

2.4.6 Violent Behavior Detection

In recent years, various intelligent surveillance techniques have been applied to ensure public safety. One desirable application is to permit computers automatically detect violent behavior, such as fighting and robbery, in surveillance videos. In this section, we utilize the proposed LPDGL to detect fight behavior. The *HockeyFight*[1] dataset is made up of 1000 video clips collected in ice hockey competitions, of which 500 contain fight behavior and 500 are non-fight sequences. The task is to identify the clips with fighting. As with Nievas et al. [2011], we adopted the space-time interest points (STIP) and motion SIFT (MoSIFT) as action descriptors, and used the Bag-of-Words (BoW) approach to represent each video clip as a histogram over 100 visual words. Every clip in the dataset was therefore characterized by a 100-dimensional feature vector.

A 5-NN graph was exploited to evaluate the transductive performance of HF, LGC, LLReg, PBSSL, LapRLS, LapSVM and LPDGL. In LNP and AGR, we chose 20 and 5 neighbors respectively for graph construction. Transductive accuracies with different l are listed in Table 2.8. We observe that HF, S4VM and LPDGL already achieve

[1] http://visilab.etsii.uclm.es/personas/oscar/FightDetection/index.html

Table 2.8: Transductive results on *HockeyFight* dataset

	$l = 40$	$l = 80$	$l = 120$	$l = 160$
LGC	0.80 ± 0.03	0.82 ± 0.02	0.83 ± 0.02	0.84 ± 0.01
HF	0.80 ± 0.02	0.84 ± 0.01	0.86 ± 0.01	0.87 ± 0.01
AGR	0.79 ± 0.02	0.82 ± 0.01	0.83 ± 0.01	0.83 ± 0.01
LNP	0.61 ± 0.08	0.65 ± 0.10	0.65 ± 0.09	0.67 ± 0.11
LapRLS	0.72 ± 0.02	0.76 ± 0.01	0.79 ± 0.01	0.79 ± 0.01
LapSVM	0.67 ± 0.03	0.66 ± 0.02	0.70 ± 0.02	0.71 ± 0.01
S4VM(Linear)	0.80 ± 0.05	0.84 ± 0.02	0.84 ± 0.03	0.86 ± 0.01
S4VM(RBF)	0.81 ± 0.03	0.84 ± 0.01	0.86 ± 0.01	0.87 ± 0.01
LLReg	0.78 ± 0.04	0.79 ± 0.01	0.82 ± 0.01	0.82 ± 0.01
PBSSL	0.74 ± 0.02	0.76 ± 0.01	0.78 ± 0.01	0.78 ± 0.01
LPDGL(Linear)	0.81 ± 0.02	0.84 ± 0.02	0.87 ± 0.00	0.87 ± 0.02
LPDGL	0.81 ± 0.03	0.85 ± 0.01	0.87 ± 0.01	0.88 ± 0.01

Table 2.9: Inductive results on *HockeyFight* dataset

	$l = 40$	$l = 80$	$l = 120$	$l = 160$
LNP	0.58 ± 0.12	0.58 ± 0.08	0.58 ± 0.10	0.59 ± 0.11
LapSVM	0.59 ± 0.02	0.61 ± 0.01	0.61 ± 0.01	0.65 ± 0.01
LapRLS	0.70 ± 0.01	0.73 ± 0.01	0.73 ± 0.01	0.74 ± 0.01
LPDGL(Linear)	0.75 ± 0.04	0.76 ± 0.01	0.76 ± 0.01	0.76 ± 0.01
LPDGL	0.71 ± 0.02	0.73 ± 0.03	0.74 ± 0.02	0.75 ± 0.01

more than 80% accuracy, which is a very encouraging result. Of particular note is that LPDGL can still improve the performances of S4VM and HF, so its superiority is demonstrated.

Inductive experiments were performed by splitting the original dataset into a training set of 600 examples and a test set of 400 test examples. Fight clips and non-fight clips constituted 50% for each of both the training set and the test set. The results of the algorithms are displayed in Table 2.9, in which the best performance under each l is marked in red. We find that LPDGL obtains very impressive inductive results.

2.5 Summary of This Chapter

This chapter has proposed a manifold-based graph transduction algorithm called Label Prediction via Deformed Graph Laplacian (LPDGL). By adopting the deformed graph Laplacian, a local smoothness term was naturally incorporated. This term can effectively prevent erroneous label propagation between classes by suppressing the labels of ambiguous examples, such as the "bridge point" mentioned above. Fig. 2.2 implies that the only difference between LapRLS and inductive LPDGL is that LPDGL has the local smoothness term but LapRLS does not, so the better performances of LPDGL over LapRLS in experiments also validates the importance of this term.

The proposed method has several profound properties, which lead to the superiority of LPDGL over other representative algorithms. Firstly, LPDGL is formulated as a convex optimization framework, so the obtained decision function is globally optimal. Secondly, the classification performance is insensitive to the change of parameters, which indicates that the parameters in LPDGL are very easy to tune. Thirdly, there exists a theoretical bound for the generalization error, so the test examples can be classified reliably and accurately. Fourthly, LPDGL can be regarded as a unified framework of various SSL algorithms, so it combines the advantages of different methodologies. Finally, the standard deviations of LPDGL listed in Tables 2.4-2.9 are very small, which reflects that the selection of initially labeled examples will not influence the final results significantly.

Chapter 3

Fick's Law Assisted Propagation

This chapter develops an iterative label propagation approach for robust graph transduction. In contrast to existing graph transduction algorithms that are derived from the perspective of statistical learning, this chapter explains label propagation by Fick's First Law of Diffusion in fluid mechanics and presents a new scheme named "Fick's Law Assisted Propagation" (FLAP). In particular, FLAP simulates the diffusion of fluid for label propagation, thus the labeled examples can be regarded as the diffusive source with a high concentration of label information. When the diffusion process starts, the flux of label information will be transferred from the labeled examples to the remaining unlabeled examples. When the diffusion process is completed, all the examples on the graph will receive a certain concentration of label information, providing the foundation for final classification.

Note that FLAP is more lifelike because it is straightforwardly derived from statistical physics. As a result, when and how much label information is received or transferred by an example, or where these labels should be propagated to, are directly governed by the well-known Fick's law, which is better than decided via some heuristic and ad hoc requirements or criteria exploited in conventional machine learning algorithms. As a result, FLAP yields more robust propagation result than the existing methods.

It is proven that FLAP can converge more quickly than other iterative methods by analyzing the relationship between the convergence rate and the eigenvalues of the iteration matrix. We show that eigenvalues of the iteration matrix in FLAP are close to 1, while those in other methods may scatter in a wide range. This difference makes FLAP is superior to other iterative methods in terms of convergence speed. We conduct the experiments on several computer vision and pattern recognition repositories, including handwritten digit recognition, face recognition and teapot image classification. Thorough empirical studies show FLAP obtains promising performance by comparing with HF Zhu et al. [2003a], LGC Zhou and Bousquet [2003], LNP Wang et al. [2009b], MinCut Joachims [2003], and NTK Zhu et al.

[2005].

3.1 Model Description

Fick's First Law of Diffusion governs mass transfer through diffusive means and has been widely used to understand the diffusion in solids, liquids, and gases. It postulates that the flux diffuses from regions of high concentration to regions of low concentration, with a magnitude that is proportional to the concentration gradient. Along one diffusion direction, the law is

$$J = -\gamma \frac{\partial f}{\partial \widetilde{d}}, \tag{3.1}$$

where γ is the diffusion coefficient, $\widetilde{d}$ is the diffusion distance, f is the concentration that evaluates the density of molecules of fluid, and J is the diffusion flux that measures the quantity of molecules flowing through the unit area per unit time.

It is natural to draw a parallel between this label propagation in the dataset $\Psi = \mathcal{L} \bigcup \mathcal{U}$ and the molecule diffusion in the fluid. In particular, the label information propagating from $\mathcal{L}$ to $\mathcal{U}$ can be compared to the molecule diffusing from high concentration regions to low concentration regions; each labeled example can be compared to a high concentration region, and each unlabeled example can be compared to a low concentration region. By treating γ as the *propagation coefficient*, d as the *propagation distance*, and f as the *label information*, Eq. (3.1) explains the process of label propagation:

$$J = -\gamma \frac{f_j^{(t)} - f_i^{(t)}}{\widetilde{d}_{ij}}. \tag{3.2}$$

Eq. (3.2) informs us that $\mathbf{x}_i$ propagates its label information to $\mathbf{x}_j$ at the diffusion distance $\widetilde{d}_{ij} = \frac{1}{\exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2)\right)}$, and their soft labels at time t are $f_i^{(t)}$ and $f_j^{(t)}$, respectively. Here soft label means that f takes a value from a real range $[-1, 1]$.

Fig. 3.1 shows the propagation process from a labeled example $\mathbf{x}_i$ to an unlabeled example $\mathbf{x}_j$, where each example is modeled by a cube. The volume of both cubes is equal to V, and the area of their interface is A. Therefore, after a short time Δt from t, the amount of label information (*i.e.* the number of molecules) received by $\mathbf{x}_j$ satisfies the following equation:

$$\frac{f_j^{(t+\Delta t)} - f_j^{(t)}}{\Delta t} V = JA, \tag{3.3}$$

where the variable J can be exactly expressed by Fick's First Law of Diffusion (3.2).

Because the label f_j varies in a discrete manner with respect to the iteration time t, Δt in Eq. (3.3) can be simply set to 1. Note that $V = \widetilde{d}_{ij} A$ and substituting Eq. (3.2)

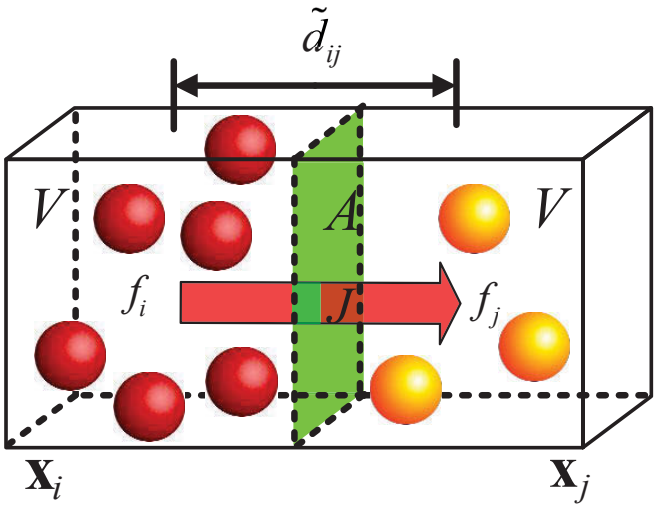

Figure 3.1: The parallel between fluid diffusion and label propagation. The left cube with more balls is compared to the example with more label information. The right cube with fewer balls is compared to the example with less label information. The red arrow indicates the diffusion direction.

into Eq. (3.3), we have

$$f_j^{(t+1)} = f_j^{(t)} - \gamma \frac{f_j^{(t)} - f_i^{(t)}}{\widetilde{d}_{ij}^2}. \tag{3.4}$$

By taking the initial state of $\mathbf{x}_j$ into account, Eq. (3.4) is modified to

$$f_j^{(t+1)} = \alpha(f_j^{(t)} - \gamma \frac{f_j^{(t)} - f_i^{(t)}}{\widetilde{d}_{ij}^2}) + (1-\alpha)y_j, \tag{3.5}$$

where $y_j = f_j^{(0)}$ and it takes a value of 1,-1 or 0, if $\mathbf{x}_j$ is a positive, negative or unlabeled example, respectively. $\alpha \in (0,1)$ is the trade off between the received information from $\mathbf{x}_i$ and the initial state of $\mathbf{x}_j$. Eq. (3.5) models the label propagation process between two examples. However in practice, one example receives the label information from all the other examples in the dataset. Therefore, if $J_{k\to j}$ is used to represent the *propagation flux* from $\mathbf{x}_k$ to $\mathbf{x}_j$, then the following equation holds:

$$(f_j^{(t+1)} - f_j^{(t)})V = \sum_{k=1}^{n} J_{k\to j} A_k. \tag{3.6}$$

We assume that $\forall k \in \{1, 2\cdots, n\}$, $J_{k\to j} = J$ and $A_k = A$. Then similar to the derivation of Eq. (3.5), after substituting Fick's First Law of Diffusion (3.2) into

Eq. (3.6) and including y_j, the propagation process in the whole dataset is given by

$$f_j^{(t+1)} = \alpha(f_j^{(t)} - \sum\nolimits_{k=1}^{n} \gamma \frac{f_j^{(t)} - f_k^{(t)}}{\widetilde{d}_{kj}^2}) + (1-\alpha)y_j. \tag{3.7}$$

Eq. (3.7) explains the propagation process between one example and the other examples in the dataset, and indicates that the received label information of an unlabeled example can be understood as an integration of the information emitted by the other examples.

According to Fick's First Law of Diffusion, we update the labels of all examples simultaneously by denoting the label vector $\mathbf{f}^{(t)} = \begin{pmatrix} f_1^{(t)} & f_2^{(t)} & \cdots & f_n^{(t)} \end{pmatrix}^\top$, and the initial state vector $\mathbf{y} = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \end{pmatrix}^\top$, and then at time t, we have

$$\mathbf{f}^{(t+1)} = \alpha \mathbf{P}\mathbf{f}^{(t)} + (1-\alpha)\mathbf{y}, \tag{3.8}$$

where the iteration matrix $\mathbf{P}$ is defined by

$$\mathbf{P} = \begin{pmatrix} 1-\gamma \sum\limits_{k=1,k\neq 1}^{n} d_{1k}^{-2} & \gamma d_{12}^{-2} & \cdots & \gamma d_{1n}^{-2} \\ \gamma d_{21}^{-2} & 1-\gamma \sum\limits_{k=1,k\neq 2}^{n} d_{2k}^{-2} \cdots & & \gamma d_{2n}^{-2} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma d_{n1}^{-2} & \gamma d_{n2}^{-2} & \cdots 1-\gamma \sum\limits_{k=1,k\neq n}^{n} d_{nk}^{-2} & \end{pmatrix}. \tag{3.9}$$

To guarantee that $\mathbf{P}$ is a non-negative stochastic matrix and the summation of elements in every row is 1, we set $0 < \gamma < \left(\max\limits_j \sum_{k=1,k\neq j}^{n} d_{jk}^{-2} \right)^{-1}$. Eq. (3.8) reveals that the soft label of an example is the linear combination of its initial state and the labels of all the examples including itself. We conduct Eq. (3.8) iteratively until convergence, *i.e.*

$$\left\| \mathbf{f}^{(t+1)} - \mathbf{f}^{(t)} \right\|_2 < \varepsilon, \tag{3.10}$$

where ε is a pre-defined small positive number. The converged vector $\mathbf{f}^*$ can then be obtained after the iteration process. Given a hard threshold 0, $\mathbf{x}_j$ is determined as positive if $f_j^* > 0$ (f_j^* is the j-th element in the vector $\mathbf{f}^*$) and negative otherwise.

Though Eq. (3.8) is derived for binary classification, it can be straightforwardly extended to multi-class classification by replacing the label vector $\mathbf{f}$ and the initial state vector $\mathbf{y}$ with the label matrix $\mathbf{F}$ and the initial state matrix $\mathbf{Y}$, respectively,

$$\mathbf{F}^{(t+1)} = \alpha \mathbf{P}\mathbf{F}^{(t)} + (1-\alpha)\mathbf{Y}, \tag{3.11}$$

where matrices $\mathbf{F}$ and $\mathbf{Y}$ are of size $n \times C$, and C denotes the total number of categories. We follow the conventional notation that $\mathbf{Y}_{jc'} = 1$ if $\mathbf{x}_j$ is labeled as $y_j = c'$, and $\mathbf{Y}_{jc'} = 0$ otherwise. If $\mathbf{x}_j$ is unlabeled, then $\mathbf{Y}_{jc'} = 0$ for $1 \leq c' \leq C$. Finally, $\mathbf{x}_j$ belongs to the class $c_j = \arg \max_{1 \leq c' \leq C} \mathbf{F}_{jc'}$. The stopping criterion for propagation is modified accordingly

$$\left\| \mathbf{F}^{(t+1)} - \mathbf{F}^{(t)} \right\|_{\mathrm{F}} < \varepsilon, \tag{3.12}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. Because Eq. (3.11) is a propagation model which simply transmits the label information according to the similarity between pairs of examples, so it can handle any kinds of data distributions. Moreover, FLAP is a graph-based algorithm, thus it can effectively exploit the manifold structure hidden in the dataset Chapelle et al. [2006]; Zhu and Goldberg [2009], which explains why it always obtains encouraging classification results.

3.2 Convergence Analysis

This section will show that FLAP (3.11) converges at linear rate. This result is directly applicable to Eq. (3.8).

Theorem 3.1. *The sequence* $\left\{ \mathbf{F}^{(t)} \right\}$, $t = 1, 2, \cdots$ *generated by Eq.* (3.11) *eventually converges to*

$$\mathbf{F}^* = \lim_{t \to \infty} \mathbf{F}^{(t)} = (1 - \alpha)(\mathbf{I} - \alpha \mathbf{P})^{-1} \mathbf{Y} \tag{3.13}$$

linearly, i.e.

$$\lim_{t \to \infty} \frac{\left\| \mathbf{F}^{(t+1)} - \mathbf{F}^* \right\|_{\mathrm{F}}}{\left\| \mathbf{F}^{(t)} - \mathbf{F}^* \right\|_{\mathrm{F}}} = \theta < 1, \tag{3.14}$$

where θ *denotes the convergence rate.*

Proof. Without loss of generality, the convergence of FLAP is studied for multi-class classification. By iteratively using Eq. (3.11), $\mathbf{F}^{(t)}$ is given by

$$\mathbf{F}^{(t)} = (\alpha \mathbf{P})^t \mathbf{Y} + (1 - \alpha) \sum\nolimits_{i=0}^{t-1} (\alpha \mathbf{P})^i \mathbf{Y}. \tag{3.15}$$

Note that $\mathbf{P}$ is a stochastic matrix ($\mathbf{P}_{ij} > 0$ and $\sum_j \mathbf{P}_{ij} = 1$), then according to the *Perron-Frobenius Theorem* Golub and Loan [1996], the spectral radius of $\mathbf{P}$ satisfies $\rho(\mathbf{P}) \leq 1$, and thus we have

$$\lim_{t \to \infty} (\alpha \mathbf{P})^t = \mathbf{0}, \quad \lim_{t \to \infty} \sum\nolimits_{i=0}^{t-1} (\alpha \mathbf{P})^i = (\mathbf{I} - \alpha \mathbf{P})^{-1}.$$

Therefore, the sequence $\left\{ \mathbf{F}^{(t)} \right\}$ will finally converge to (3.13).

To prove that FLAP converges at linear rate, we need to demonstrate Eq. (3.14) holds. Considering that

$$\mathbf{F}^{(t+1)} = (\alpha\mathbf{P})^{t+1}\mathbf{Y} + (1-\alpha)\sum\nolimits_{i=0}^{t}(\alpha\mathbf{P})^{i}\mathbf{Y}, \tag{3.16}$$

we have Eq. (3.17) by plugging Eqs. (3.13), (3.15) and (3.16) into Eq. (3.14),

$$\begin{aligned}&\frac{\left\|\mathbf{F}^{(t+1)}-\mathbf{F}^{*}\right\|_{\mathrm{F}}}{\left\|\mathbf{F}^{(t)}-\mathbf{F}^{*}\right\|_{\mathrm{F}}}\\ =&\frac{\left\|\left[(\alpha\mathbf{P})^{t+1}+(1-\alpha)\sum_{i=0}^{t}(\alpha\mathbf{P})^{i}-(1-\alpha)(\mathbf{I}-\alpha\mathbf{P})^{-1}\right]\mathbf{Y}\right\|_{\mathrm{F}}}{\left\|\left[(\alpha\mathbf{P})^{t}+(1-\alpha)\sum_{i=0}^{t-1}(\alpha\mathbf{P})^{i}-(1-\alpha)(\mathbf{I}-\alpha\mathbf{P})^{-1}\right]\mathbf{Y}\right\|_{\mathrm{F}}}.\end{aligned} \tag{3.17}$$

Since $\mathbf{P}$ is symmetric, it can be decomposed into

$$\mathbf{P} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top}, \tag{3.18}$$

where $\mathbf{U}$ is an unitary matrix and $\mathbf{\Lambda} = diag(\lambda_1, \lambda_2, \cdots, \lambda_n)$ is a diagonal matrix containing eigenvalues of $\mathbf{P}$. According to (3.18) and the *Woodbury matrix identity* Higham [1996], we have

$$(\mathbf{I}-\alpha\mathbf{P})^{-1} = \mathbf{I} + \alpha\mathbf{U}\big(\mathbf{\Lambda}^{-1}-\alpha\mathbf{I}\big)^{-1}\mathbf{U}^{\top}. \tag{3.19}$$

By substituting Eq. (3.18) and Eq. (3.19) into Eq. (3.17), we obtain

$$\begin{aligned}&\frac{\left\|\mathbf{F}^{(t+1)}-\mathbf{F}^{*}\right\|_{\mathrm{F}}}{\left\|\mathbf{F}^{(t)}-\mathbf{F}^{*}\right\|_{\mathrm{F}}}\\ =&\frac{\left\|\left\{\mathbf{U}(\alpha\mathbf{\Lambda})^{t+1}\mathbf{U}^{\top}+(1-\alpha)\sum_{i=0}^{t}\mathbf{U}(\alpha\mathbf{\Lambda})^{i}\mathbf{U}^{\top}-\mathbf{F}^{*}\right\}\mathbf{Y}\right\|_{\mathrm{F}}}{\left\|\left\{\mathbf{U}(\alpha\mathbf{\Lambda})^{t}\mathbf{U}^{\top}+(1-\alpha)\sum_{i=0}^{t-1}\mathbf{U}(\alpha\mathbf{\Lambda})^{i}\mathbf{U}^{\top}-\mathbf{F}^{*}\right\}\mathbf{Y}\right\|_{\mathrm{F}}}\\ =&\frac{\left\|\left\{\mathbf{U}\left(\alpha^{t+1}\mathbf{\Lambda}^{t+1}+\mathbf{M}\right)\mathbf{U}^{\top}\right\}\mathbf{Y}\right\|_{\mathrm{F}}}{\left\|\left\{\mathbf{U}\left(\alpha^{t+1}\mathbf{\Lambda}^{t}+\mathbf{M}\right)\mathbf{U}^{\top}\right\}\mathbf{Y}\right\|_{\mathrm{F}}},\end{aligned} \tag{3.20}$$

where $\mathbf{M} = (1-\alpha)\Big[\sum_{i=1}^{t}(\alpha\mathbf{\Lambda})^{i}-\alpha(\mathbf{\Lambda}^{-1}-\alpha\mathbf{I})^{-1}\Big]$. From Eq. (3.20) we can easily observe that the only difference between the denominator and numerator is that one more diagonal matrix $\mathbf{\Lambda}$ is multiplied to the first term in the bracket of numerator. Thus Eq. (3.20) can be rewritten as

$$\frac{\left\|\mathbf{F}^{(t+1)}-\mathbf{F}^{*}\right\|_{\mathrm{F}}}{\left\|\mathbf{F}^{(t)}-\mathbf{F}^{*}\right\|_{\mathrm{F}}} = \frac{\left\|\left\{\mathbf{U}\alpha^{t+2}diag\left(\kappa_1 \quad \kappa_2 \quad \cdots \quad \kappa_n\right)\mathbf{U}^{\top}\right\}\mathbf{Y}\right\|_{\mathrm{F}}}{\left\|\left\{\mathbf{U}\alpha^{t+1}diag\left(\pi_1 \quad \pi_2 \quad \cdots \quad \pi_n\right)\mathbf{U}^{\top}\right\}\mathbf{Y}\right\|_{\mathrm{F}}}, \tag{3.21}$$

in which $\kappa_i = \frac{\lambda_i^{t+1}(1-\lambda_i)}{1-\alpha\lambda_i}$ and $\pi_i = \frac{\lambda_i^{t}(1-\lambda_i)}{1-\alpha\lambda_i}$. If we denote

$$\mathbf{U}\mathbf{J}_A\mathbf{U}^\top \triangleq \mathbf{A}, \quad \mathbf{U}\mathbf{J}_B\mathbf{U}^\top \triangleq \mathbf{B},$$

where $\mathbf{J}_A = \alpha^{t+2} diag\begin{pmatrix}\kappa_1 & \kappa_2 & \cdots & \kappa_n\end{pmatrix}$ and $\mathbf{J}_B = \alpha^{t+1} diag\begin{pmatrix}\pi_1 & \pi_2 & \cdots & \pi_n\end{pmatrix}$ that contain the eigenvalues of $\mathbf{A}$ and $\mathbf{B}$, respectively, then Eq. (3.21) is simplified as

$$\frac{\left\|\mathbf{F}^{(t+1)} - \mathbf{F}^*\right\|_{\mathrm{F}}}{\left\|\mathbf{F}^{(t)} - \mathbf{F}^*\right\|_{\mathrm{F}}} = \frac{\left\|\mathbf{A}\mathbf{Y}\right\|_{\mathrm{F}}}{\left\|\mathbf{B}\mathbf{Y}\right\|_{\mathrm{F}}}. \tag{3.22}$$

Moreover, since $\mathbf{P}$ is a stochastic matrix with all its eigenvalues $\lambda_i \in [-1, 1]$ (In fact, 1 is the single eigenvalue of $\mathbf{P}$ according to the *Perron-Frobenius Theorem* Golub and Loan [1996]) and $\alpha \in (0, 1)$, so all the eigenvalues of $\mathbf{A}$ and $\mathbf{B}$ are non-negative, and the eigenvalues of $\mathbf{A}$ are smaller than those of $\mathbf{B}$ in the corresponding position expressed in $\mathbf{J}_A$ and $\mathbf{J}_B$. Therefore,

$$\frac{\left\|\mathbf{A}\mathbf{Y}\right\|_{\mathrm{F}}}{\left\|\mathbf{B}\mathbf{Y}\right\|_{\mathrm{F}}} = \frac{\left\|\mathbf{U}\mathbf{J}_A\mathbf{U}^\top\mathbf{Y}\right\|_{\mathrm{F}}}{\left\|\mathbf{U}\mathbf{J}_B\mathbf{U}^\top\mathbf{Y}\right\|_{\mathrm{F}}} = \sqrt{\frac{tr\left(\mathbf{Y}^\top\mathbf{U}{\mathbf{J}_A}^2\mathbf{U}^\top\mathbf{Y}\right)}{tr\left(\mathbf{Y}^\top\mathbf{U}{\mathbf{J}_B}^2\mathbf{U}^\top\mathbf{Y}\right)}}, \tag{3.23}$$

where $tr(\cdot)$ is the symbol of trace. Let $\mathbf{Q} = \mathbf{U}\mathbf{J}\mathbf{U}^\top$ where $\mathbf{J} = {\mathbf{J}_B}^2 - {\mathbf{J}_A}^2 \geq 0$, then we obtain the margin between the denominator and numerator in Eq. (3.23) as

$$tr\left(\mathbf{Y}^\top\mathbf{U}{\mathbf{J}_B}^2\mathbf{U}^\top\mathbf{Y}\right) - tr\left(\mathbf{Y}^\top\mathbf{U}{\mathbf{J}_A}^2\mathbf{U}^\top\mathbf{Y}\right) = tr\left[\mathbf{Y}^\top\mathbf{U}\left({\mathbf{J}_B}^2 - {\mathbf{J}_A}^2\right)\mathbf{U}^\top\mathbf{Y}\right] = tr\left(\mathbf{Y}^\top\mathbf{Q}\mathbf{Y}\right). \tag{3.24}$$

Note that $\mathbf{Q} = \mathbf{U}\sqrt{\mathbf{J}}\left(\mathbf{U}\sqrt{\mathbf{J}}\right)^\top$ and $\mathbf{U}\sqrt{\mathbf{J}}$ are invertible ($\det(\mathbf{U}\sqrt{\mathbf{J}}) = \det(\mathbf{U}) \cdot \det(\sqrt{\mathbf{J}}) \neq 0$), so $\mathbf{Q}$ is a positive definite matrix. Furthermore, if $\mathbf{Y}$ is partitioned by columns as $\mathbf{Y} = \begin{pmatrix}Y_1 & Y_2 & \cdots & Y_n\end{pmatrix}$, then

$$tr\left(\mathbf{Y}^\top\mathbf{Q}\mathbf{Y}\right) = tr\left(\sum\nolimits_{i=1}^{n} Y_i^\top\mathbf{Q}Y_i\right). \tag{3.25}$$

Because $\mathbf{Q}$ is positive definite and $Y_1,\ Y_2 \cdots Y_n$ are non-zero vectors (this is guaranteed, because each class should has at least one labeled example), the result of Eq. (3.25) is definitely above zero. Therefore, the denominator of Eq. (3.23) is larger than the numerator, which indicates that

$$0 < \frac{\left\|\mathbf{F}^{(t+1)} - \mathbf{F}^*\right\|_{\mathrm{F}}}{\left\|\mathbf{F}^{(t)} - \mathbf{F}^*\right\|_{\mathrm{F}}} < 1. \tag{3.26}$$

According to Eq. (3.26), we can conclude

$$\lim_{t\to\infty} \frac{\left\|\mathbf{F}^{(t+1)} - \mathbf{F}^*\right\|_{\mathrm{F}}}{\left\|\mathbf{F}^{(t)} - \mathbf{F}^*\right\|_{\mathrm{F}}} = \theta < 1. \tag{3.27}$$

Thus the linear convergence rate of FLAP for multi-class classification is proved. □

3.3 Interpretation and Connections

In this section, we will first build the relationship between our FLAP and other existing methods by reformulating FLAP into the traditional regularization theory, and then elaborate that FLAP can be understood from the perspective of Markov random fields and graph kernels.

3.3.1 Regularization Networks

To straightforwardly compare FLAP with other graph transduction algorithms and show its superiority, we reformulate FLAP in the context of the classical regularization theory:

$$\min_{\mathbf{f}\in\mathbb{R}^n} Q(\mathbf{f}) = \frac{1}{2}\left[\sum_{k=1}^{n}\sum_{j=1}^{n} p_{kj}(f_k - f_j)^2 + \tau\sum_{k=1}^{n}(f_k - y_k)^2\right], \tag{3.28}$$

where p_{kj} is the (k,j)-th element in the matrix $\mathbf{P}$. The first term in the right-hand side of Eq. (3.28) forms a specific prior knowledge and enforces the labels in $\mathbf{f}$ varying smoothly. It indicates that the soft labels of similar examples should not differ too much from one another. The second fitting term means that the decided labels should be consistent with the examples' original states well[1]. The regularization parameter $\tau > 0$ controls the trade-off between smoothness term and fitting term.

It is straightforward to show that the optimal solution of Eq. (3.28) equals the iterative result of Eq. (3.8). The derivative of $Q(\mathbf{f})$ with respect to $\mathbf{f}$ is

$$\partial Q/\partial \mathbf{f} = 2\left(\mathbf{I} - \mathbf{P}\right)\mathbf{f} + \tau\left(\mathbf{f} - \mathbf{y}\right). \tag{3.29}$$

Set the right-hand side of Eq. (3.29) to 0 and let $\alpha = \frac{2}{2+\tau}$, $\beta = \frac{\tau}{2+\tau}$ so that $\alpha + \beta = 1$, then Eq. (3.29) is reformulated as

$$\mathbf{f} - \alpha\mathbf{P}\mathbf{f} - \beta\mathbf{y} = \mathbf{0}, \tag{3.30}$$

which leads to the same closed-form solution as Eq. (3.13):

$$\mathbf{f} = \beta(\mathbf{I} - \alpha\mathbf{P})^{-1}\mathbf{y}. \tag{3.31}$$

Theorem 3.2. *The regularization form* (3.28) *is a convex optimization problem.*

[1] Similar to LGC Zhou and Bousquet [2003], LNP Wang et al. [2009b] and GTAM Wang et al. [2008b], we penalize the deviations of the labels of all the examples from their initial states. This formulation prefers the consistency for labeled data, and the compromise in assigning the labels to the unlabeled data Wang et al. [2009b].

Proof. By denoting $d_k = \sum_{j=1,j\neq k}^{n} p_{kj}$ and calculating the Hessian matrix $\mathbf{H}_0$ of Eq. (3.28), we have

$$\mathbf{H}_0 = \begin{pmatrix} 2d_1+\tau & -2p_{12} & \cdots & -2p_{1n} \\ -2p_{21} & 2d_2+\tau & \cdots & -2p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -2p_{n1} & -2p_{n2} & \cdots & 2d_n+\tau \end{pmatrix}. \tag{3.32}$$

From *Gerschgorin circle theorem* Golub and Loan [1996] we know that all the eigenvalues of $\mathbf{H}_0$ belong to $[\tau, 4\max_{1\leq k\leq n}(d_k)+\tau]$, which reveals that $\mathbf{H}_0$ is positive definite. Therefore, the optimization problem (3.28) is convex and the convergent result is globally optimal. □

Theorem 3.2 also implies that the convergent point of FLAP (3.31) corresponds to the global optimal solution. Given Eq. (3.28), Table 3.1 compares FLAP with representative algorithms in a straightforward way, in which $N(\mathbf{x}_k)$ denotes the neighbors of the example $\mathbf{x}_k$ and $\omega_{kj}=\exp\left(-\|\mathbf{x}_k-\mathbf{x}_j\|^2/(2\sigma^2)\right)$. It can be observed that the main difference between these methods is the definition of smoothness. FLAP uses the matrix $\mathbf{P}$ to describe the smoothness, while Mincut, HF, LNP and LGC adopt the (normalized) graph Laplacian to define the smoothness. This difference leads to the faster convergence rate achieved by FLAP.

The regularization framework of FLAP is also much related to the semi-supervised formulation of binary kernel spectral clustering (BKSC) Alzate and Suykens [2012]. Suppose the adjacency matrix of a graph is $\mathbf{W}$, and the diagonal matrix $\mathbf{D}$ is defined by $\mathbf{D}_{kk}=\sum_j \mathbf{W}_{kj}$, $\forall k=1,2,\cdots,n$, then BKSC is

$$\min_{\omega,\mathbf{b}} \frac{1}{2}\left[\omega^\top\omega-\gamma_1\mathbf{f}^\top\mathbf{D}^{-1}\mathbf{f}+\gamma_2\sum\nolimits_{i=1}^{l}(f_i-y_i)^2\right]. \tag{3.33}$$

In Eq. (3.33), $\mathbf{f}=\mathbf{\Theta}\omega+\mathbf{b}\mathbf{e}_l$, where $\mathbf{e}_l$ is an all-one l-dimensional column vector, and $\mathbf{\Theta}$ is the kernelized data matrix with each row representing an example. ω and $\mathbf{b}$ are unknown vectors to be optimized, and γ_1, γ_2 are non-negative parameters balancing the three terms. BKSC (3.33) differs from FLAP (3.28) regarding the prior knowledge. FLAP prefers the solution to be smooth, while BKSC emphasizes the examples corresponding to large $\mathbf{D}_{kk}$.

3.3.2 Markov Random Fields

Markov Random Fields (MRF) are a set of random variables having the Markov property described by an undirected graph. First Order Intrinsic Gaussian Markov Random Fields (FO-IGMRF) is an MRF in which the joint distribution is Gaussian and the precision matrix $\mathbf{Q}$ is rank reduced and meanwhile satisfies $\mathbf{Q}\mathbf{e}=\mathbf{0}$ ($\mathbf{e}$ is

Table 3.1: FLAP vs. popular graph transduction algorithms.

Algorithm	Regularization Framework
Mincut	$\min_{\mathbf{f}: f_{\{j,k\}} \in \{-1,1\}} \quad Q(\mathbf{f}) = \sum_{k=1}^{n} \sum_{j=1}^{n} \omega_{kj} (f_k - f_j)^2 + \theta \sum_{k=1}^{n} (f_k - y_k)^2$
HF	$\min_{\mathbf{f}: f_{\{j,k\}} \in \mathbb{R}} \quad Q(\mathbf{f}) = \sum_{k=1}^{n} \sum_{j=k}^{n} \omega_{kj} (f_k - f_j)^2 + \theta \sum_{k=1}^{n} (f_k - y_k)^2$
LNP	$\min_{\mathbf{f}: f_{\{j,k\}} \in \mathbb{R}} \quad Q(\mathbf{f}) = \sum_{k=1}^{n} \sum_{j: \mathbf{x}_j \in N(\mathbf{x}_k)} \omega_{kj} (f_k - f_j)^2 + \theta \sum_{k=1}^{n} (f_k - y_k)^2$
LGC	$\min_{\mathbf{f}: f_{\{j,k\}} \in \mathbb{R}} \quad Q(\mathbf{f}) = \frac{1}{2} \left[\sum_{k=1}^{n} \sum_{j=1}^{n} \omega_{kj} \left(\frac{1}{\sqrt{D_{kk}}} f_k - \frac{1}{\sqrt{D_{jj}}} f_j \right)^2 + \theta \sum_{k=1}^{n} (f_k - y_k)^2 \right]$
FLAP	$\min_{\mathbf{f}: f_{\{j,k\}} \in \mathbb{R}} \quad Q(\mathbf{f}) = \frac{1}{2} \left[\sum_{k=1}^{n} \sum_{j=1}^{n} p_{kj} (f_k - f_j)^2 + \theta \sum_{k=1}^{n} (f_k - y_k)^2 \right]$

a vector of all ones) Rue and Held [2005]. Similar to other propagation algorithms, *e.g.* Wang et al. [2009b], FLAP can also be cast into the framework of FO-IGMRF. Defining the increment along the edge in a graph as

$$\Delta d_{kj} = f_k - f_j. \tag{3.34}$$

Suppose Δd_{kj} fits the Gaussian distribution with $\Delta d_{kj} \sim N(0, d_{kj}^2/\gamma)$, then we have

$$p(\Delta d_{kj}) \propto \exp\left[-\frac{\gamma}{2 d_{kj}^2} (f_k - f_j)^2 \right]. \tag{3.35}$$

Assuming that all the increments along the edges are conditionally independent, the joint probability over $\mathbf{f}$ is calculated as the product of the probabilities over all the increments:

$$p(\mathbf{f}) \propto \prod p(\Delta d_{kj}) = \exp\left[-\frac{1}{2} \gamma \sum \frac{1}{d_{kj}^2} (f_k - f_j)^2 \right]. \tag{3.36}$$

Suppose the adjacency matrix $\mathbf{W}$ of a graph is

$$\mathbf{W}_{kj} = \begin{cases} \gamma d_{kj}^{-2} & k \neq j \\ 0 & k = j \end{cases}, \tag{3.37}$$

then Eq. (3.36) is reformulated as

$$p(\mathbf{f}) \propto \exp\left[-\frac{1}{2} \gamma \mathbf{f}^\top (\mathbf{D} - \mathbf{W}) \mathbf{f} \right] = \exp\left[-\frac{1}{2} \gamma \mathbf{f}^\top \mathbf{L} \mathbf{f} \right], \tag{3.38}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the graph Laplacian. Since $\mathbf{L}\mathbf{e} = \mathbf{0}$, $\mathbf{L}$ is exactly the precision matrix $\mathbf{Q}$ in FO-IGMRF.

3.3.3 Graph Kernels

Because of the popularity of graph-based methods for data analyses, many diffusion kernels on graph vertices have been developed, such as exponential diffusion kernel Kondor and Lafferty [2002] and Von Neumann diffusion kernel Shawe-Taylor and Cristianini [2004]. This section examines the relationship between the proposed FLAP and these popular diffusion kernels.

If we ignore the manually incorporated initial state term y_i in Eq. (3.7), and only focus on the propagation process itself, Eq. (3.7) reduces to

$$f_j^{(t+1)} = f_j^{(t)} + \gamma \sum\nolimits_{k=1}^{n} d_{kj}^{-2} f_k^{(t)} - \gamma \sum\nolimits_{k=1}^{n} d_{kj}^{-2} f_j^{(t)}. \tag{3.39}$$

By regarding f_j as a variable w.r.t. the time t, we have

$$\begin{aligned} \frac{df_j}{dt} &= \gamma \sum\nolimits_{k=1}^{n} \left[(1-\delta_{kj}) d_{kj}^{-2} f_k - \delta_{kj} \sum\nolimits_{i=1}^{n} d_{ij}^{-2} f_j \right], \\ &= \gamma \sum\nolimits_{k=1}^{n} \left[(1-\delta_{kj}) d_{kj}^{-2} - \delta_{kj} \sum\nolimits_{i=1}^{n} d_{ij}^{-2} \right] f_k \end{aligned} \tag{3.40}$$

where δ_{kj} is the Kronecker delta with $\delta_{kj} = 1$ if $k = j$, and 0 otherwise. Therefore, we can write the variables f_j $(j = 1, \cdots, n)$ as Eq. (3.40) into a compact matrix form:

$$d\mathbf{f}/dt = (\mathbf{P} - \mathbf{I})\mathbf{f}, \tag{3.41}$$

where $\mathbf{P}$ is the iteration matrix defined by Eq. (3.9), and $\mathbf{f} = (f_1, \cdots f_n)^\top$ is an n-dimensional variable vector of time t. By denoting $\mathbf{H} = \mathbf{P} - \mathbf{I}$, the above differential equation leads to the solution $\mathbf{f} = \exp(\mathbf{H}t)\mathbf{f}^{(0)}$, which results in the graph kernel related to FLAP:

$$\mathbf{K}_{FLAP} = \exp(\tilde{\sigma}\mathbf{H}) = \sum\nolimits_{i=0}^{\infty} \frac{\tilde{\sigma}^i \mathbf{H}^i}{i!}, \tag{3.42}$$

where $\tilde{\sigma}$ is a real parameter. Since $\mathbf{H}$ is a positive semi-definite matrix, $\mathbf{K}_{FLAP}$ can be regarded as an exponential diffusion kernel according to Kondor and Lafferty [2002].

FLAP is also related to the Von Neumann diffusion kernel, which has the formation of $\mathbf{K}_{VND} = (\mathbf{I} - \tilde{\alpha}\mathbf{A})^{-1} = \sum_{i=0}^{\infty} \tilde{\alpha}^i \mathbf{A}^i$ with $0 < \tilde{\alpha} < \rho(\mathbf{A})^{-1}$ Shawe-Taylor and Cristianini [2004]. For FLAP, since $\mathbf{P}$'s spectral radius $\rho(\mathbf{P}) = 1$, and $\alpha \in (0, 1)$ as explained in Section 3.1, the term $(\mathbf{I} - \alpha\mathbf{P})^{-1}$ in Eq. (3.13) is actually a Von Neumann diffusion kernel and encodes the similarities of pairs of examples.

3.4 Experimental Results

In this section, FLAP will be evaluated on typical synthetic and vision datasets. Several popular graph transduction algorithms serve as baselines for comparison, including

Table 3.2: Performances of all the methods on two synthetic datasets. Each record follows the format "iteration time/CPU seconds/accuracy".

Type	Algorithms	*Square&Ring*	*DoubleMoon*
Iterative	LGC Zhou and Bousquet [2003]	68/0.436/54.2%	880/4.488/100.0%
	LNP Wang et al. [2009b]	235/1.685/79.3%	156/1.044/91.8%
	FLAP	114/0.658/100.0%	20/0.338/100.0%
Non-iterative	MinCut Joachims [2003]	-/-/100.0%	-/-/89.6%
	HF Zhu et al. [2003a]	-/-/100.0%	-/-/100.0%
	NTK Zhu et al. [2005]	-/-/24.7%	-/-/99.6%

HF Zhu et al. [2003a], LGC Zhou and Bousquet [2003], LNP Wang et al. [2009b], NTK Zhu et al. [2005] and MinCut Joachims [2003]. We focus on two issues of these algorithms: one is the classification accuracy on unlabeled examples given very few labeled examples for every dataset, and the other is the efficiency such as CPU seconds and iteration times. In all the experiments below, the parameters of FLAP are set to $\alpha = 0.99$, and $\gamma = \eta\left(\max_j \sum_{k=1,k\neq j}^{n} d_{jk}^{-2}\right)^{-1}$ where η is chosen within $[0, 1]$. In HF, LGC and LNP, the parameter α is also set to 0.99. For fair comparison, we construct the identical K-NN graph for all the algorithms, and the σ in the expression of the diffusion distance has been also respectively adjusted to achieve the best performance for different datasets.

3.4.1 Synthetic Data

Square&Ring and *DoubleMoon* datasets are used to visualize the propagation processes of three iterative methods, including LGC, LNP and the proposed FLAP. The results of the three non-iterative methods, *i.e.* MinCut, HF, and NTK, are also reported.

The *Square&Ring* dataset contains a square and a ring, both of which are centered at (0.5,0.5). The radius of the outer ring is 1.3, and the length of each side of the inner square is 1 (see Fig. 3.2). In the *DoubleMoon* dataset, 500 examples are equally divided into two moons that are centered at $(0, 0)$ and $(6, 0)$, respectively. The width of each moon is set to 6 to make the moons "fatter". Compared with other literatures Wang and Zhang [2006]; Zhou and Bousquet [2003] that use the *DoubleMoon* dataset for illustration, we reduce the inter-class distance to increase the classification difficulty (see Fig. 3.3). Note that there is only one labeled example in each class for both datasets (see $t = 0$ in every subplot).

For implementing FLAP on the *Square&Ring* dataset, we set $K = 5$, $\sigma = 0.2$

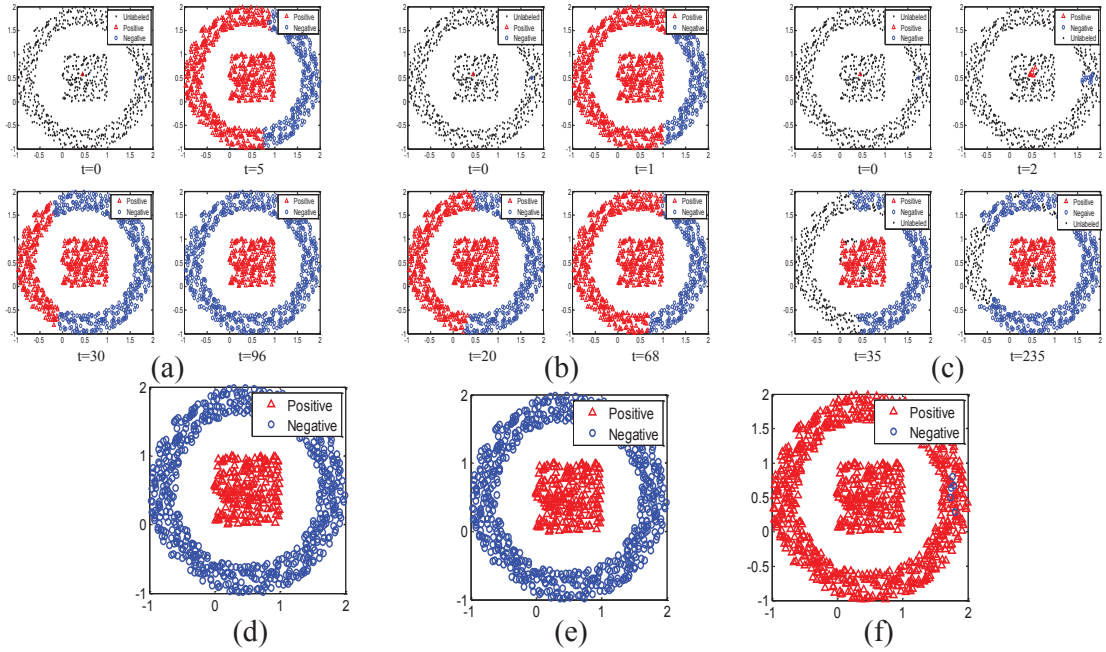

Figure 3.2: The propagation results on *Square&Ring* dataset. (a), (b), (c) are propagation processes of FLAP, LGC and LNP, respectively. (d), (e), (f) present the classification results brought by MinCut, HF and NTK.

and $\eta = 0.95$. Fig. 3.2 reveals that LGC wrongly propagates positive labels to the outer ring. Part of the unlabeled examples cannot receive label information through LNP until convergence. Most negative examples are classified as positive by NTK. By contrast, FLAP, MinCut and HF are able to correctly classify all the examples. The parameter settings of FLAP on *DoubleMoon* are $K = 5$, $\sigma = 0.2$ and $\eta = 0.6$. The results are displayed in Fig. 3.3. It can be observed that LNP propagates the positive label to the upper moon by mistake in about $t = 27$. Mincut and NTK also fail to obtain the perfect results. Comparatively, LGC, HF and FLAP perform better than the above methods. For quantitative comparison, we also report the classification accuracies of these methods on *Square&Ring* and *DoubleMoon* in Table 3.2.

Moreover, to validate the efficiency of FLAP, we record both iteration times and CPU seconds of the three iterative methods. In this work, all the algorithms are conducted on a work station with 2.40GHz Intel Xeon CPU and 24G memory. Table 3.2 suggests FLAP achieves comparable time costs with LGC on the *Square&Ring* dataset. However, the convergent result of LGC is incorrect shown by Fig. 3.2, and this can be the reason why it converges quickly in this dataset. The convergence curves are also shown in Fig. 3.4 and demonstrate that FLAP converges very quickly on both datasets.

We further use the *DoubleMoon* dataset to test the ability of FLAP for handling

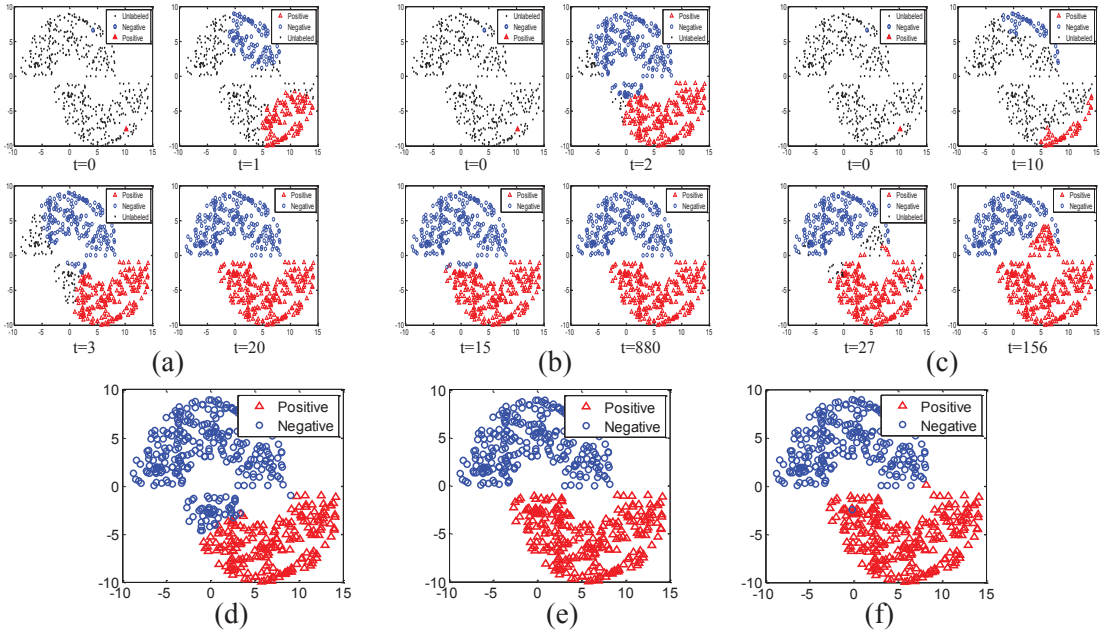

Figure 3.3: The propagation results on *DoubleMoon* dataset. (a), (b), (c) are propagation processes of FLAP, LGC and LNP, respectively. (d), (e), (f) present the classification results brought by MinCut, HF and NTK.

the problem of imbalanced data. The number of negative examples is fixed to 250, while we randomly sample 125, 50, 25, 10 positive examples from the bottom moon, so the relative ratio of positive examples to negative examples are 1:2, 1:5, 1:10 and 1:25, respectively. The initial labeled examples for this experiment are identical to Fig. 3.3, and the propagation results are presented in Fig. 3.5. It can be observed that FLAP can successfully classify all the examples except 1:25 situation. In Fig. 3.5 (d), two positive examples are mistakenly classified into the negative class. This is mainly because 1) the positive examples are so scarce that they fail to represent the underlying manifold of the real distribution of the positive examples (*i.e.* the moon shaped by positive points); 2) the two mistakenly classified examples are closer to the upper moon, so the negative labels can be more conveniently propagated to them than the positive labels. This experiment well demonstrates that FLAP is insensitive to the problem of imbalanced data.

3.4.2 Real Benchmarks Data

In this section, the performances of FLAP, LGC, LNP, HF, NTK and MinCut are evaluated by testing on the two public datasets *USPS* and *BCI* in Chapelle's book

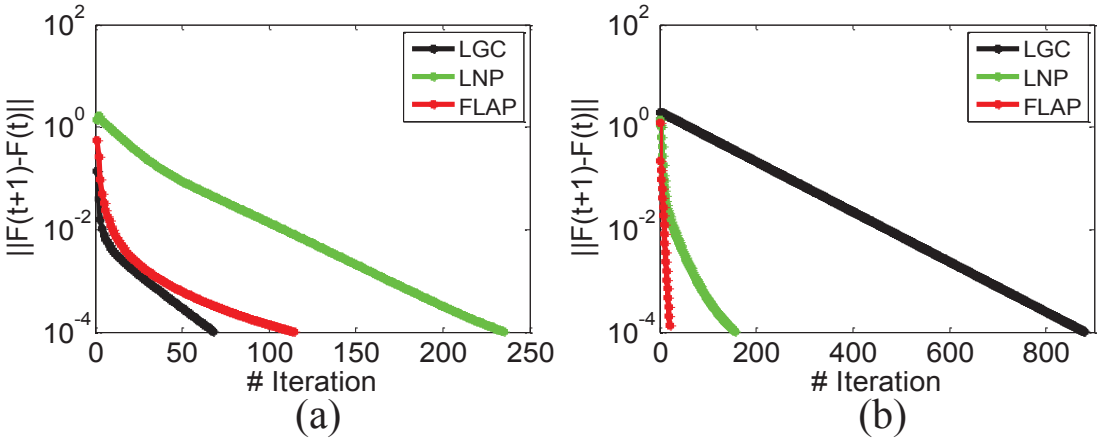

Figure 3.4: The comparison of convergence curves. (a) is the result on *Square&Ring* and (b) is that on *DoubleMoon*.

Chapelle et al. [2006].

The *USPS* dataset contains ten digits, each with 150 images. We combine digits 2 and 5 as the positive class, and form all the others as the negative class. Therefore, the two classes are imbalanced and the relative size of the two classes is 1:4. The *BCI* dataset is used to study the brain-computer interface. It contains 400 imagined movements (examples) with 200 using the left hand (negative class) and 200 using the right hand (positive class).

In each dataset, we implement all the algorithms under $l = 10$ and $l = 100$, and the final results are averaged over 12 independent runs with different partitions of labeled and unlabeled subsets. We built 9-NN and 8-NN graphs with $\sigma = 0.2$ for *USPS* and *BCI* respectively, and η in FLAP is set to 0.1 and 0.01 for these two datasets. The number of principal components in NTK is set to $m = 100$ to achieve the optimal performance. Table 3.3 shows the accuracies of different algorithms, in which the highest and the second highest records are marked with red and blue color, respectively. We observe that compared with all the baselines, the proposed FLAP obtains encouraging performance generally. An exceptional case is that FLAP performs slightly worse than LGC on *USPS* when $l = 100$. Specifically, the experimental results on *USPS* also demonstrate that FLAP is not sensitive to the problem of the imbalanced data.

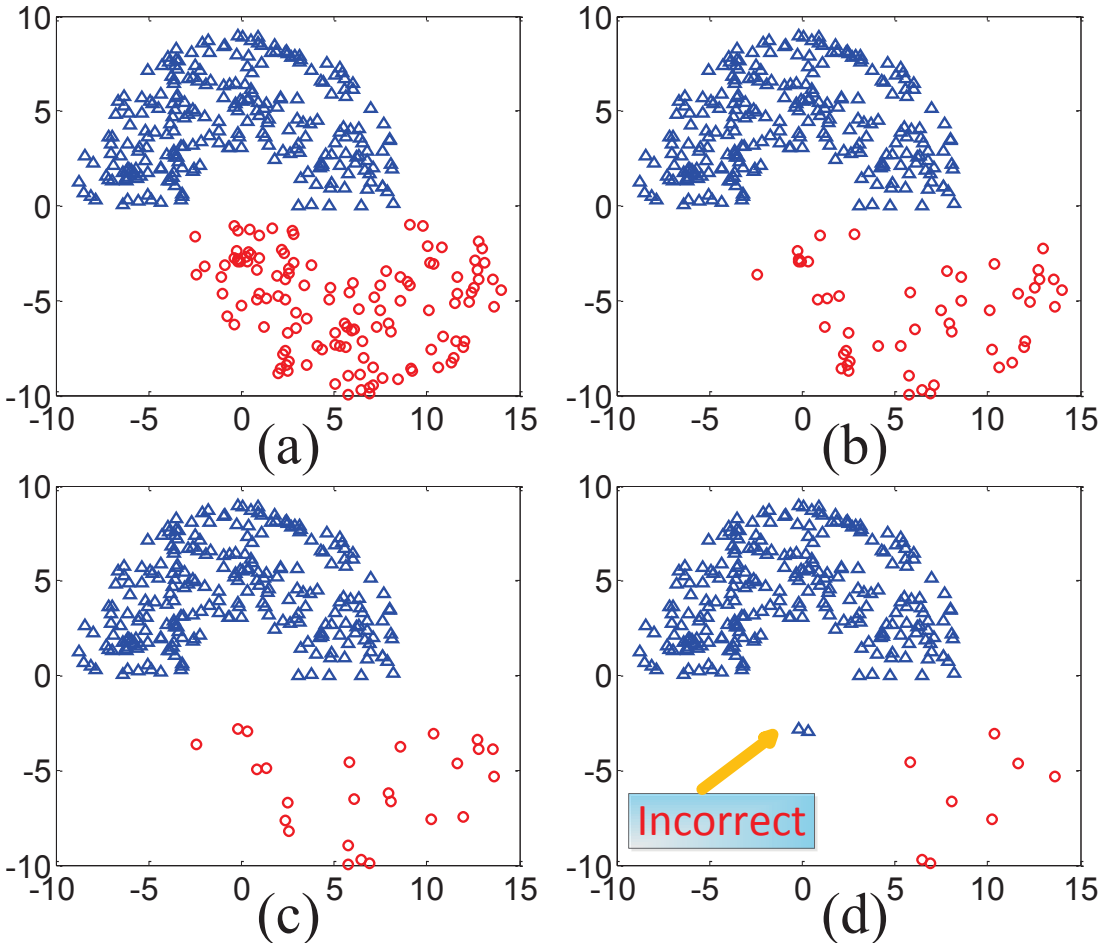

Figure 3.5: Classification outputs on imbalanced *DoubleMoon*. (a), (b), (c) and (d) are results of 1:2, 1:5, 1:10 and 1:25 situations, respectively.

3.4.3 UCI Data

We adopt four UCI Machine Learning Repository datasets[1], *Iris*, *Wine*, *BreastCancer*, and *CNAE-9* to compare FLAP with the other graph-based algorithms. The classification accuracy and iteration times under different sizes of the labeled set l, in particular, are evaluated. Algorithms are implemented 30 times independently under each l with randomly selected examples, and the reported accuracy and iteration times are calculated as the mean value of the outputs of these runs. Note that at least one labeled example is guaranteed in each class when the labeled sets are generated. Five state-of-the-art graph transduction algorithms are adopted for baselines, including LGC, LNP,

[1] http://archive.ics.uci.edu/ml/.

Table 3.3: Experimental results on the benchmark datasets for the variety of graph transduction algorithms. (The values in the table represent accuracy (%).)

Datasets	*USPS*		*BCI*	
$l(\#Labeled\ Examples)$	10	100	10	100
LGC Zhou and Bousquet [2003]	85.90	94.96	51.40	59.85
LNP Wang et al. [2009b]	77.75	79.98	50.85	56.73
HF Zhu et al. [2003a]	86.39	93.64	49.64	53.78
NTK Zhu and Lafferty [2005]	79.57	87.65	49.29	48.97
MinCut Joachims [2003]	80.21	94.21	51.88	66.35
FLAP	88.31	94.24	53.40	66.90

MinCut, HF and NTK.

We built 10-NN, 6-NN, 7-NN, and 10-NN graphs for *Iris*, *Wine*, *BreastCancer*, and *CNAE-9*, respectively. Other parameters of FLAP are $\sigma = 0.2$, $\eta = 0.1$ for *Iris*, $\sigma = 0.5$, $\eta = 0.01$ for *Wine*, $\sigma = 0.5$, $\eta = 0.001$ for *BreastCancer*, and $\sigma = 1$, $\eta = 0.1$ for *CNAE-9*. The m in NTK is set to 50 for all the four UCI datasets. Figs. 3.6 (a) (c) (e) (g) demonstrate that FLAP is generally able to reach the highest accuracy among the comparators when l changes from small to large.

Moreover, Figs. 3.6 (b) (d) (f) (h) present the iteration times of all the iterative methods (LGC, LNP and FLAP) on four UCI datasets, respectively. Other baselines including HF, MinCut and NTK are not compared here because they are not iteration-based. It can be observed that FLAP requires the least iteration times in most cases. An exceptional case is that LNP is more efficient than FLAP on the *Wine* dataset. However, the accuracy of LNP is not as high as that of FLAP shown by Fig. 3.6 (c), so the results obtained by FLAP are encouraging.

3.4.4 Handwritten Digit Recognition

Handwritten digit recognition is a branch of optical character recognition (OCR). We compare FLAP with baselines on the *Optical Recognition of Handwritten Digits* Dataset[1], (abbreviated as "*Digits*") in which numbers 0∼9 are considered as different classes. This dataset contains 5620 digital images and the resolution of each image is 8×8. The pixel-wise feature is described by a 64-dimensional vector with elements representing the gray values. Labeled examples are randomly selected from the whole dataset. We built a 10-NN graph for all the methods, and choose $\sigma = 7$, $\eta = 0.01$ for

[1] http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

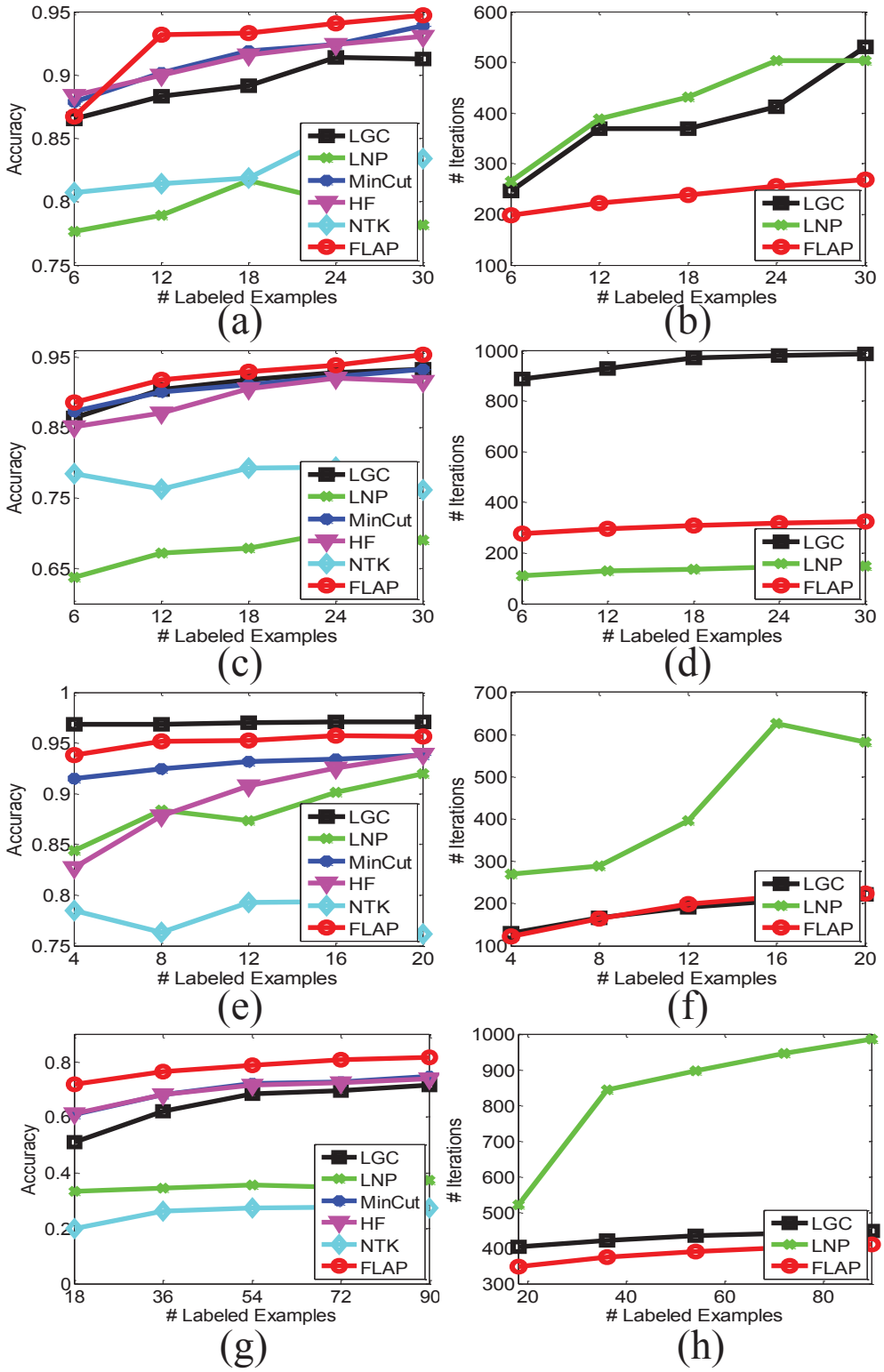

Figure 3.6: Comparison of accuracy and iteration times. (a) (b) denote *Iris*, (c) (d) denote *Wine*, (e) (f) denote *BreastCancer*, and (g) (h) denote *CNAE-9*.

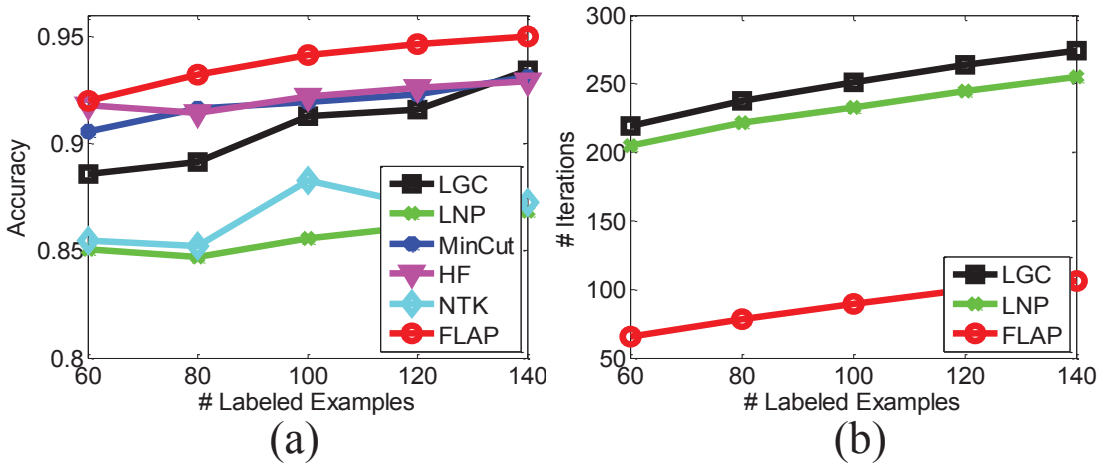

Figure 3.7: Comparison of accuracy and iteration times on digit recognition dataset. (a), (b) are the curves of accuracy and iteration times with the growing of the labeled examples, respectively.

FLAP.

When l varies from small to large, the accuracy and iteration times of the methods are plotted in Figs. 3.7 (a) and (b), respectively. This figure indicates that FLAP achieves the highest accuracy with the least iteration times.

3.4.5 Teapot Image Classification

The *Teapot* dataset Zhu and Lafferty [2005] contains 365 images of a teapot, and the angle of the spout in every image is different (see Fig. 3.8 (a)). The goal is to determine whether the orientation of the spout is right or left. The resolution of each image is 12×16, and hence the pixel-wise feature adopted is a 192-dimensional vector.

A 10-NN graph with $\sigma = 30$ was established for every algorithm. η for FLAP was set to 0.01 to get the optimal performance. Fig. 3.8 (b) shows the accuracy curve of several algorithms when the size of the labeled data varies from 4 to 20, which reveals that FLAP performs better than other algorithms.

3.4.6 Face Recognition

We tested the ability of FLAP on face recognition by using the challenging dataset *Labeled Face In the Wild* (*LFW*)[1]. The face images in this dataset are directly collected under natural scenes, so the facial expressions, observation angle, illumination

[1] http://vis-www.cs.umass.edu/lfw/

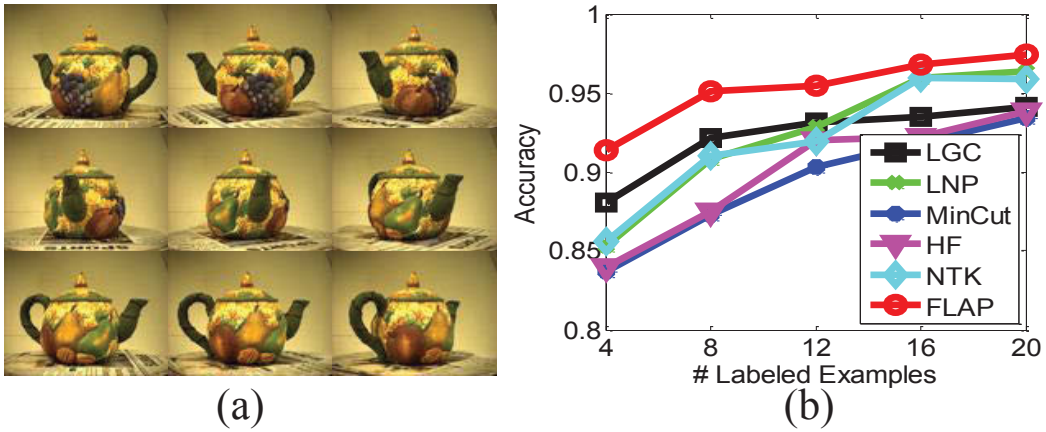

Figure 3.8: Experiment on *Teapot* dataset. (a) shows some typical images. (b) is the accuracy curve for comparison.

conditions and background setting are not intentionally controlled for recognition. The faces in all the images are detected by the well-known Viola-Jones detector Viola and Jones [2001].

In this experiment, we used a subset of *LFW* by choosing the persons who have more than 30 face images. We chose the images of Toledo, Sharon, Schwarzenegger, Powell, Rumsfeld, Bush, Arroyo, Agassi, Beckham and Hewitt for recognition, which leads to totally 392 examples belonging to 10 people in the subset (see Fig. 3.9 (a) for some examples). We adopted the 73-dimensional feature developed by Kumar *et al.* Kumar et al. [2009], which describes some biometrics traits such as gender, race, age, and hair color. A 10-NN graph with $\sigma = 1$ was built on the entire dataset, and η was set to 0.5. The recognition rates of algorithms are compared in Fig. 3.9 (b). We observe that HF and MinCut perform comparably to FLAP, while NTK and LNP are inferior to FLAP notably. Generally, FLAP achieves very satisfactory results and outperforms other methods with l changing from 20 to 100.

3.4.7 Statistical Significance

Above experiments have empirically shown the superiority of FLAP to other baselines in terms of classification accuracy. In this section we use the 5×2 cross-validation F-test (5×2 cv F-test) Alpaydin [1999] introduced in Section 2.4.3 to show the statistical significance of FLAP over the other methods. Table 3.4 lists the F-statistics values of baselines versus FLAP, which suggests that the hypothesis is rejected in most cases. Therefore, the superiority of FLAP to other baselines is statistically verified. Besides, we observe that the null hypothesis is often accepted by MinCut on *Wine*, and LNP on

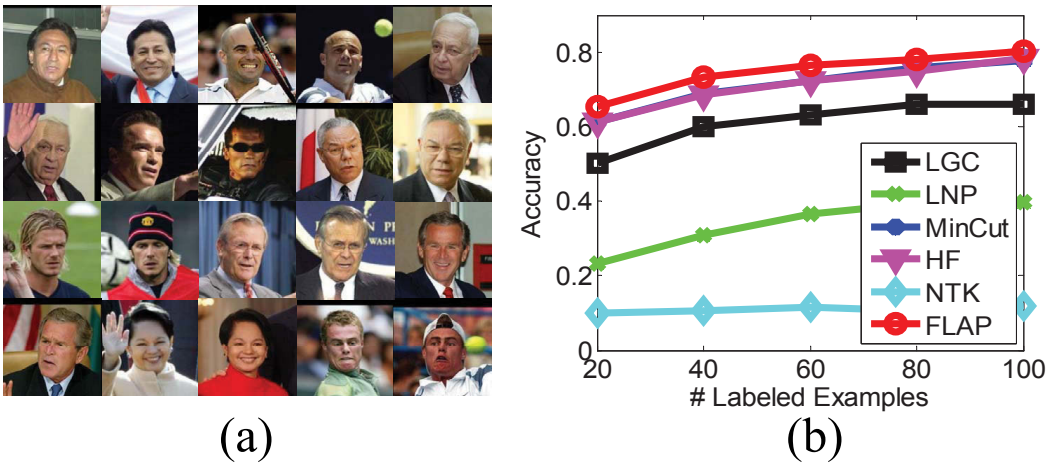

Figure 3.9: Experimental results on *LFW* face dataset. (a) shows some representative images, and (b) compares the recognition accuracy.

Teapot, which suggest that they achieve comparable performances with FLAP on the corresponding datasets. Fig. 3.6 (c) and Fig. 3.8 (b) also show this point.

3.4.8 Computational Cost

To further evaluate the computational efficiency of baselines and the proposed FLAP, this section compares the CPU seconds of all the algorithms on all the adopted vision datasets. The results are presented in Table 3.5. It can be observed that HF is the most efficient algorithm among the non-iterative methods, and the proposed FLAP generally needs the least CPU time compared with other iteration-based approaches. By properly setting the parameter η, FLAP can perfectly control the distribution of eigenvalues of its iteration matrix, so it achieves the fastest convergence speed among the compared iterative algorithms. The detailed reason will be explained in the next section. Besides, we note that the non-iterative methods are more efficient than the iterative algorithms when the dataset is small (*e.g. Iris* and *Wine* datasets). However, when the database contains a large number of examples, *e.g. Digits* with 5620 examples, the proposed iterative FLAP begins to show its strength in terms of efficiency. Compared with non-iterative methods (MinCut, HF, and NTK) that take more than 30 seconds to complete one implementation, FLAP requires only less than 20 CPU seconds. The reason is that non-iterative methods usually need to invert a large matrix when massive examples exist, while FLAP successfully avoids this inversion by conducting in an iterative way, so FLAP is more suitable for dealing with relatively large data collections.

3.4.9 Parametric Settings

Two parameters, K and η, are critical for FLAP to obtain satisfying results. K controls the sparsity of an established graph, and η plays an important role in balancing the accuracy and efficiency.

3.4.9.1 Choosing η

The impacts of η for FLAP's performance are twofold. Intuitively, from (3.15) we observe that a diagonally dominant iteration matrix (such as $\mathbf{P}$ in FLAP) leads to higher convergence rate. In other words, a small η helps to reduce the iteration times. The following analyses will elaborate this point strictly.

Lemma 3.3. *Suppose $\mathbf{G}_{n\times n}$ is the iteration matrix of an iterative algorithm, e.g. FLAP defined in Eq.* (3.11)*, of which the eigenvalues are $\xi_1, \xi_2 \cdots \xi_n$, and then the upper bound of the iteration times $t_{\max}$ satisfies*

$$\sum\nolimits_{i=1}^{n} (\alpha\xi_i)^{2t_{\max}} (\xi_i - 1)^2 = \varepsilon^2/(l\alpha^2), \tag{3.43}$$

where ε, l and α are the same as those defined above.

Proof. Suppose the general iterative expression of graph-based label propagation is

$$\mathbf{F}^{(t+1)} = \alpha\mathbf{G}\mathbf{F}^{(t)} + (1-\alpha)\mathbf{Y}, \tag{3.44}$$

where $\mathbf{G}$ is the iteration matrix, then according to the stopping criterion

$$\left\|\mathbf{F}^{(t+1)} - \mathbf{F}^{(t)}\right\|_{\mathrm{F}} < \varepsilon, \tag{3.45}$$

we have the following inequality:

$$\begin{aligned} &\left\|\mathbf{F}^{(t+1)} - \mathbf{F}^{(t)}\right\|_{\mathrm{F}} = \alpha^{t+1}\left\|(\mathbf{G}-\mathbf{I})\mathbf{G}^t\mathbf{Y}\right\|_{\mathrm{F}} \\ &\leq \alpha^{t+1}\left\|(\mathbf{G}-\mathbf{I})\mathbf{G}^t\right\|_{\mathrm{F}}\|\mathbf{Y}\|_{\mathrm{F}} \leq \varepsilon, \end{aligned} \tag{3.46}$$

where $\|\mathbf{Y}\|_{\mathrm{F}} = \sqrt{l}$. Similarly, $\mathbf{G}$ can also be decomposed as $\mathbf{G} = \mathbf{V}\widetilde{\mathbf{D}}\mathbf{V}^\top$, where $\mathbf{V}$ is an unitary matrix and $\widetilde{\mathbf{D}} = diag(\xi_1, \xi_2 \cdots \xi_n)$ is a diagonal matrix containing n eigenvalues of $\mathbf{G}$. If $\mathbf{V}$ is partitioned by columns as $\mathbf{V} = \begin{pmatrix}\mathbf{V}_1 & \mathbf{V}_2 & \cdots & \mathbf{V}_n\end{pmatrix}$, then $\mathbf{G} = \sum_{i=1}^{n} \xi_i \mathbf{V}_i \mathbf{V}_i^\top$. Moreover, considering that the identity matrix $\mathbf{I} = \mathbf{V}\mathbf{V}^\top = \sum_{i=1}^{n} \mathbf{V}_i\mathbf{V}_i^\top$, we have

$$\begin{aligned} \left\|(\mathbf{G}-\mathbf{I})\,\mathbf{G}^t\right\|_{\mathrm{F}} &= \left\|\left(\sum_{i=1}^{n}\xi_i\mathbf{V}_i\mathbf{V}_i^\top - \mathbf{I}\right)\left(\sum_{i=1}^{n}\lambda_i^t\mathbf{V}_i\mathbf{V}_i^\top\right)\right\|_{\mathrm{F}} \\ &= \sqrt{\sum_{i=1}^{n}\xi_i^{2t}(\xi_i-1)^2 tr(\mathbf{V}_i\mathbf{V}_i^\top)}. \end{aligned} \tag{3.47}$$

Because $\|\mathbf{V}_i\|_2 = 1$, so $tr(\mathbf{V}_i\mathbf{V}_i^\top) = 1$ for $1 \leq i \leq n$. Hence (3.47) can be further simplified as

$$\left\|(\mathbf{G}-\mathbf{I})\,\mathbf{G}^t\right\|_{\mathrm{F}} = \sqrt{\sum\nolimits_{i=1}^{n} \xi_i^{2t}(\xi_i-1)^2}. \tag{3.48}$$

Finally, by substituting (3.48) into (3.46) we have

$$\sum\nolimits_{i=1}^{n} (\alpha\xi_i)^{2t_{\max}}(\xi_i-1)^2 = \frac{\varepsilon^2}{l\alpha^2}. \tag{3.49}$$

This reveals the relationship between the upper bound of iteration times $t_{\max}$ and the eigenvalues of the iteration matrix $\mathbf{G}$. □

According to Lemma 3.3, we investigate what requirement of the eigenvalues of the iteration matrix should meet to obtain the minimum $t_{\max}$. In fact, this question is equal to the following optimization problem

$$\begin{array}{ll} \min & t_{\max} \\ s.t. & \sum_{i=1}^{n} (\alpha\xi_i)^{2t_{\max}}(\xi_i-1)^2 = \frac{\varepsilon^2}{l\alpha^2} \end{array}, \tag{3.50}$$

which can be rewritten as

$$\min \;\; t_{\max} + \rho_0 \left(\sum\nolimits_{i=1}^{n} (\alpha\xi_i)^{2t_{\max}}(\xi_i-1)^2 - \frac{\varepsilon^2}{l\alpha^2} \right), \tag{3.51}$$

where ρ_0 is the Lagrange multiplier. If we denote

$$F(\xi_1, \cdots, \xi_n, \rho_0) = t_{\max} + \rho_0 \left(\sum_{i=1}^{n} (\alpha\xi_i)^{2t_{\max}}(\xi_i-1)^2 - \frac{\varepsilon^2}{l\alpha^2} \right), \tag{3.52}$$

then solving (3.50) equals to finding the solution of

$$\begin{cases} \frac{\partial F}{\partial \rho_0} = \sum_{i=1}^{n} (\alpha\xi_i)^{2t_{\max}}(\xi_i-1)^2 - \frac{\varepsilon^2}{l\alpha^2} = 0 \\ \frac{\partial F}{\partial \lambda_1} = 4\rho_0 t_{\max}\alpha^{2t_{\max}}\xi_1^{2t_{\max}-1}(\xi_1-1) = 0 \\ \qquad\qquad \vdots \\ \frac{\partial F}{\partial \lambda_n} = 4\rho_0 t_{\max}\alpha^{2t_{\max}}\xi_n^{2t_{\max}-1}(\xi_n-1) = 0 \end{cases}. \tag{3.53}$$

It can be concluded that Eq. (3.53) is maximally satisfied if ξ_i equals to 0 or 1 (note that $\varepsilon^2/l\alpha^2 \approx 0$). Fortunately, Theorem 3.4 below guarantees that all the eigenvalues of FLAP's iteration matrix $\mathbf{P}$ are almost equivalent to 1 by choosing a small η:

Theorem 3.4. *Let λ_i ($1 \leq i \leq n$) be the eigenvalues of FLAP's iteration matrix, then $1-2\eta \leq \lambda_i \leq 1$ where $\eta = \gamma \max\limits_{j} \sum_{k=1,k\neq j}^{n} d_{kj}^{-2}$ as defined in the beginning of Section 3.4.*

Proof. To prove Theorem 3.4, we need a lemma from Huang and Yang [2007].

Lemma 3.5. *Huang and Yang [2007] Suppose* $\mathbf{A} = (a_{ij})$ *is a stochastic matrix,* $q = \min\{a_{ii}, i \in N\}$*, then all the eigenvalues of* $\mathbf{A}$ *satisfy* $|\lambda_i - q| \leq 1 - q$*.*

Now we begin to prove the Theorem 3.4. Since the iteration matrix $\mathbf{P}$ of FLAP is a symmetric stochastic matrix, all its eigenvalues are in the real range $[-1, 1]$, so according to Lemma 3.5, q in our case satisfies $q = 1 - \gamma \max_j \sum_{k=1,k\neq j}^{n} d_{kj}^{-2} = 1 - \eta \left(\max_j \sum_{k=1,k\neq j}^{n} d_{kj}^{-2} \right)^{-1} \max_j \sum_{k=1,k\neq j}^{n} d_{kj}^{-2} = 1 - \eta$. By substituting the expression of q into Lemma 3.5, we complete the proof. □

Above analyses explain why we set the parameter η (or γ) to a relatively small positive number (*e.g.* $\eta = 0.1$ in *Iris*, $\eta = 0.01$ in *Wine*, and $\eta = 0.001$ in *BreastCancer*, etc.). By choosing a small η, $\mathbf{P}$ can be diagonally dominant and its eigenvalues $\lambda_i (1 \leq i \leq n)$ will be distributed very close to 1. Fig. 3.10 shows that the eigenvalues of the FLAP iteration matrix are only slight smaller than 1. In contrast, the eigenvalues of the iteration matrices in LNP and LGC are widely scattered in the range $[-1, 1]$. Therefore, FLAP is able to converge relatively more quickly than LGC and LNP as Table 3.5 illustrates.

On the other hand, if η is set to an extremely small value, then $\mathbf{P}$ will be very close to an identity matrix. Eq. (3.13) reveals that under this situation, the convergence result is almost the same as the initial state $\mathbf{Y}$, which means the label information cannot be thoroughly propagated from initially labeled examples to other unlabeled ones. Therefore, an extremely small η will damage the classification accuracy significantly.

In short, a small η leads to higher convergence rate, but decreases the classification accuracy; a large η can boost the accuracy at the cost of high computational time. Therefore, η should be chosen by trading off the accuracy and efficiency. By fixing $l = 12$ on the *Iris* dataset, we plot the accuracy and iteration times of FLAP w.r.t. the variations of η in Figs. 3.11 (a) and (b). These two figures also indicate that both the accuracy and iteration times will rise by gradually increasing η. In the *Iris* dataset, we set η to 0.1 because the accuracy is relatively high and the iteration times are also acceptable under this setting.

3.4.9.2 Choosing K

A suitable graph is very important for improving the performance. As mentioned above, K is a critical parameter determining the number of neighborhoods and the sparsity of an established graph. This section studies how K influences the classification accuracy and iteration times. In Figs. 3.11 (c) and (d), we fix $\eta = 0.1$ and change the value of K from small to large. It can be observed that if the graph is too sparse (*e.g.* $K = 2$), FLAP will not achieve satisfying performance. However,

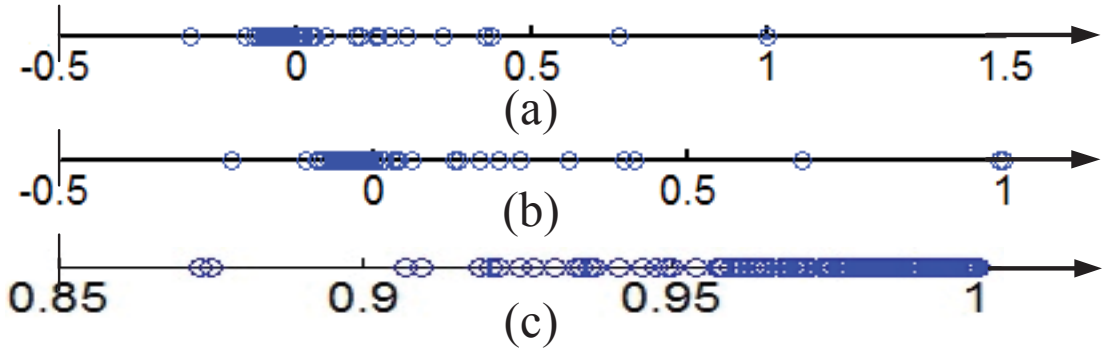

Figure 3.10: Distribution of eigenvalues on *Iris*. (a) denotes LNP, (b) denotes LGC and (c) denotes FLAP. Note that the ranges of the three x-axes are different.

when K is larger than a certain value (*e.g.* $K = 6$), FLAP functions properly and produces encouraging results. Besides, Fig. 3.11 (d) reveals that the choice of K will not influence the iteration times significantly. Therefore, both the accuracy and iteration times are not sensitive to the choice of K if K is not too small. In other words, this parameter can be easily tuned.

3.5 Summary of This Chapter

We presented the FLAP algorithm to propagate labels by adopting Fick's First Law of Diffusion in fluid mechanics. We showed FLAP can also be derived in the context of the traditional regularization theory, which not only relates the FLAP algorithm to existing algorithms but also implies that the convergent point of FLAP is globally optimal. It was also demonstrated that the parameter η played an important role in determining classification performance and iteration times. Comprehensive experimental results suggest that FLAP obtains competitive performance when compared with state-of-the-art transduction algorithms.

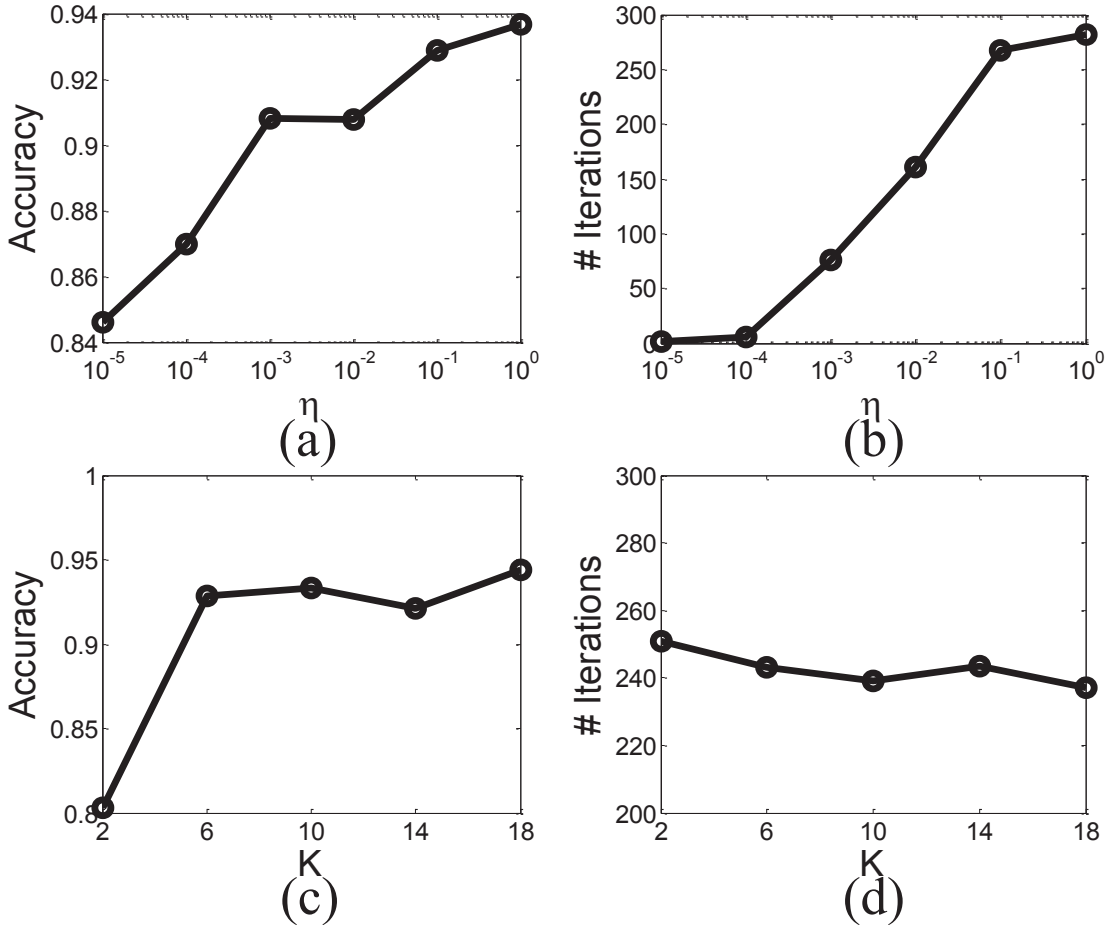

Figure 3.11: The impact of parametric settings on accuracy and iteration times. (a), (b) investigate η, and (c), (d) evaluate K.

Table 3.4: F-statistics values of baselines versus FLAP on four UCI datasets. (The records smaller than 4.74 are marked in red, which mean that 1) the null hypothesis is accepted, and 2) the corresponding baseline algorithm performs comparably to FLAP.)

	l	LGC	LNP	MinCut	HF	NTK
Iris	6	1.28	3.84	1.25	1.84	8.23
	12	20.71	12.93	5.69	6.76	12.46
	18	18.05	5.02	5.06	4.80	9.76
	24	13.77	7.17	2.14	4.60	13.08
	30	14.42	11.49	2.57	2.92	10.91
Wine	6	4.99	41.65	3.56	9.21	39.99
	12	3.29	16.81	3.13	5.58	208.63
	18	1.52	55.17	2.93	2.61	192.36
	24	1.51	24.65	2.74	2.77	222.16
	30	5.55	26.33	8.50	7.23	864.96
BreastCancer	4	2.64	14.39	2.08	8.13	37.77
	8	4.21	13.88	9.58	22.92	701.85
	12	6.86	27.37	6.80	18.29	939.93
	16	7.89	8.22	78.31	27.53	8132.42
	20	5.25	7.00	3.78	6.86	2747.30
CNAE-9	18	44.70	98.95	100.32	13.59	1434.33
	36	8.27	102.22	42.59	61.96	10596.37
	54	20.88	54.82	21.22	24.10	2107.28
	72	14.29	94.07	58.05	28.98	7535.61
	90	10.14	103.75	68.64	32.27	3852.48
Digits	60	101.51	14.53	7.14	6.12	153.55
	80	78.99	41.57	13.05	4.06	2211.52
	100	63.08	21.67	8.27	7.34	503.88
	120	73.60	118.29	7.13	7.68	438.75
	140	24.31	19.33	8.36	8.52	672.71
Teapot	4	2.94	7.39	16.51	8.41	9.54
	8	3.89	4.49	9.07	10.76	10.37
	12	4.23	3.81	18.65	31.21	12.98
	16	10.89	1.61	7.82	7.05	2.21
	20	8.40	2.42	7.66	5.91	3.48
LFW	20	17.41	267.00	3.20	7.07	1252.94
	40	33.76	410.25	8.99	12.16	5420.41
	60	28.69	129.32	6.08	11.84	3327.88
	80	18.23	106.75	19.82	7.34	4424.79
	100	24.00	113.43	4.02	12.09	2139.00

Table 3.5: CPU time (unit: seconds) of various methods. (For each l, the smallest record among iterative methods is marked in red, while the smallest record among non-iterative methods is highlighted in blue.)

		Non-iterative methods			Iterative methods		
	l	MinCut	HF	NTK	LGC	LNP	FLAP
Iris	6	0.006	0.002	0.324	0.023	0.025	0.021
	12	0.008	0.002	0.356	0.034	0.040	0.024
	18	0.009	0.002	0.356	0.034	0.046	0.026
	24	0.008	0.002	0.412	0.038	0.062	0.029
	30	0.008	0.002	0.416	0.048	0.056	0.029
Wine	6	0.012	0.004	0.395	0.102	0.019	0.038
	12	0.015	0.002	0.313	0.105	0.016	0.040
	18	0.010	0.003	0.377	0.110	0.020	0.043
	24	0.023	0.002	0.386	0.111	0.018	0.045
	30	0.017	0.002	0.426	0.111	0.031	0.045
BreastCancer	4	0.122	0.059	0.422	0.254	0.746	0.226
	8	0.111	0.078	0.507	0.330	0.827	0.310
	12	0.109	0.061	0.493	0.376	1.156	0.373
	16	0.122	0.057	0.544	0.401	1.771	0.405
	20	0.123	0.057	0.542	0.422	1.768	0.420
CNAE-9	18	0.352	0.151	1.207	3.877	3.904	2.967
	36	0.380	0.147	1.689	4.015	6.148	3.146
	54	0.381	0.147	2.752	4.156	6.622	3.235
	72	0.379	0.149	1.717	4.209	6.844	3.369
	90	0.318	0.150	2.112	4.218	7.120	3.484
Digits	60	40.659	33.811	35.057	41.903	38.486	12.124
	80	39.154	34.344	34.268	45.747	41.473	14.490
	100	40.027	34.086	38.052	48.374	43.471	16.658
	120	40.006	34.117	37.615	50.508	46.011	18.518
	140	42.180	33.902	40.585	52.173	47.958	19.757
Teapot	4	0.026	0.011	0.422	0.088	0.481	0.010
	8	0.027	0.014	0.473	0.106	0.477	0.031
	12	0.027	0.013	0.450	0.118	0.488	0.043
	16	0.029	0.012	0.461	0.115	0.493	0.049
	20	0.028	0.012	0.468	0.124	0.506	0.058
LFW	20	0.050	0.012	0.993	0.295	0.180	0.169
	40	0.036	0.015	1.110	0.213	0.202	0.209
	60	0.060	0.014	1.751	0.219	0.219	0.213
	80	0.036	0.012	2.780	0.228	0.239	0.245
	100	0.038	0.015	4.261	0.235	0.252	0.269

Chapter 4

Label Propagation Via Teaching-to-Learn and Learning-to-Teach

In this chapter, we aim to design a robust iterative label propagation approach for graph transduction. Existing graph-based propagation algorithms usually treat unlabeled examples equally, and transmit seed labels to the unlabeled examples as long as they are connected to the labeled examples. However, such a popular propagation scheme is very likely to yield inaccurate propagation, because it falls short of tackling ambiguous but critical data points (*e.g.*, outliers). To address this shortcoming, in this chapter we treat the unlabeled examples in different levels of difficulties by assessing their reliability and discriminability, and explicitly optimize the propagation quality by manipulating the propagation sequence to move from simple to difficult examples.

So far, machine learning has been studied from different aspects for a long history, and numerous learning algorithms have been developed to tackle different types of problems. However, as an important counterpart of machine learning, machine teaching has been largely ignored by the majority of researchers. Therefore, we incorporate a "teacher" (*i.e.* a teaching algorithm) and a "learner" (*i.e.* a learning algorithm) into one framework to achieve accurate propagation.

Specifically, we propose a novel iterative label propagation algorithm in which each propagation alternates between two paradigms, teaching-to-learn and learning-to-teach. In the teaching-to-learn step, the teacher picks up a set of the simplest examples for the learner to conduct the propagation. In the learning-to-teach step, the teacher absorbs the learner's learning feedback to adjust the choice of the simplest examples for the next learning round. The proposed **T**eaching-to-**L**earn and **L**earning-to-**T**each (TLLT) strategy has two major advantages: 1) it helps to boost the accuracy of label propagation, and 2) TLLT makes the entire propagation process very robust to the disturbance of the Gaussian kernel width for graph construction.

4.1 A Brief Introduction to Machine Teaching

The early works related to machine teaching mainly focus on the "teaching dimension" theoryBalbach and Zeugmann [2006]; Goldman and Kearns [1995]; Shinohara and Miyano [1991]. Recently, some teaching algorithms have been developed such as Dekel et al. [2012]; Mei and Zhu [2015]; Patil et al. [2014]; Singla et al. [2014]; Zhu [2013, 2015]; Zilles et al. [2011] . In above literatures, a teacher is supposed to know the exact labels of a curriculum. However, in our case a teacher is assumed to only know the difficulties of examples without accessing their real labels, which poses a great challenge to teaching and learning. Besides, different from Dekel and Shamir [2009] that considers the teachers may be malicious, the teacher incorporated by our algorithm is supposed to be always helpful.

In 2009, Bengio et al. Bengio et al. [2009] proposed "curriculum learning", which treats available examples as curriculums with different levels of difficulties in running a stepwise learner. In each step, the simplest set of examples should be "taught" to the learner. Therefore, we also regard curriculum learning as a branch of machine teaching, even though the teacher does not care about explicit labels of the training examples. Instead, the teacher only knows the difficulty of the available examples. Actually, the argument of learning from simple to difficult levels has been broadly acknowledged in the human cognitive domain Elman [1993]; Khan et al. [2011], and also gradually applied to advance the existing machine learning algorithms in recent years. For example, Kumar et al. [2010] proposed "self-paced learning", which adaptively decides which and how many examples are taken as the curriculum according to the learner's ability. The self-paced learning can be regarded as an implementation of the curriculum learning. This learning strategy has been applied to visual category discovery Lee and Grauman [2011], object tracking Supancic and Ramanan [2013], detection Tang et al. [2012], and multimedia retrieval Jiang et al. [2014a]. By introducing the anti-group-sparsity term, Jiang et al. Jiang et al. [2014b] picked up curriculums that are not only simple but also diverse. Jiang et al. Jiang et al. [2015] also combined curriculum learning with self-paced learning so that the proposed model can exploit both the estimation of example difficulty before learning and information about the dynamic difficulty rendered during learning.

Although various teaching algorithms have been proposed in recent years, none of them focus on the graph transduction problem. To the best of our knowledge, this is the first work to model label propagation as a teaching and learning framework, so that abundant unlabeled examples are activated to receive the propagated labels in a well-organized sequence. We employ the state-of-the-art label propagation algorithm Zhu and Ghahramani [2002] as the learner, because it is naturally incremental without retraining when a new curriculum comes.

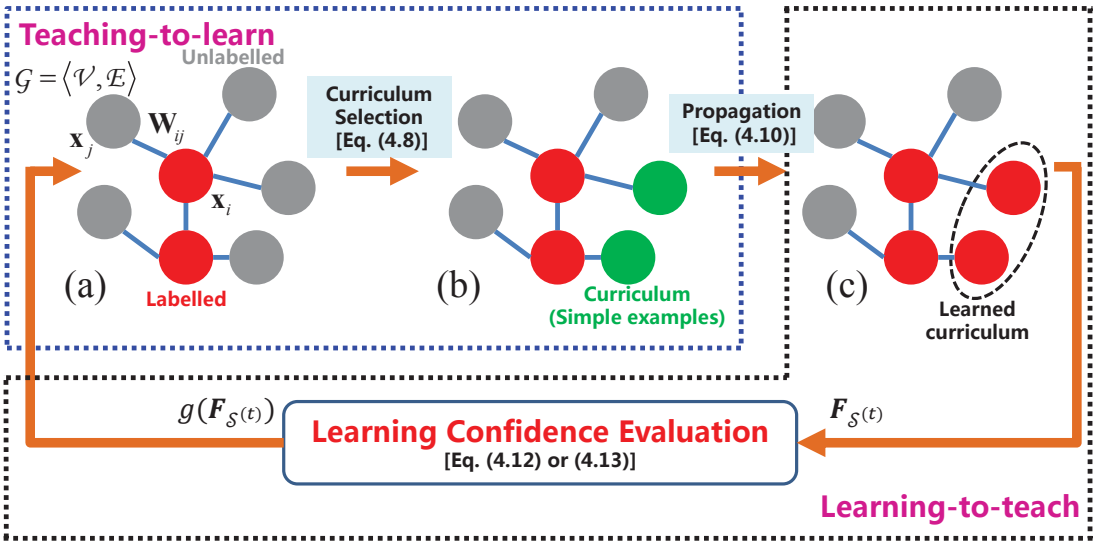

Figure 4.1: The TLLT framework for label propagation. The labeled examples, unlabeled examples, and curriculum are represented by red, grey, and green balls, respectively. The steps of Teaching-to-Learn and Leaning-to-Teach are marked with blue and black dashed boxes.

4.2 Overview of the Proposed Framework

The framework of our proposed TLLT is shown in Fig. 4.1. In the teaching-to-learn step, a teaching model that serves as a "teacher" is established to select the simplest examples (*i.e.*, a curriculum, see green balls) from the pool of unlabeled examples (grey balls) for the current propagation. This selection is performed by solving an optimization problem that integrates the reliability and discriminability of each unlabeled example. In the learning-to-teach step, a "learner" activates the simplest examples to conduct label propagation using the classical method presented in Zhu and Ghahramani [2002], and meanwhile delivers its learning confidence to the teacher in order to assist the teacher in deciding the subsequent simplest examples. Such a two-step procedure iterates until all the unlabeled examples are properly handled. As a result of the interactions between the teacher and learner, the originally difficult (*i.e.*, ambiguous) examples are handled at a late time, so that they can be reliably labeled via leveraging the previously learned knowledge.

We provide a toy example to further illustrate the propagation strategy of our algorithm. In Fig. 4.2, we aim to identify whether the orientation of the spout in each image is right or left Weinberger et al. [2004]. According to the labeled positive (red box) and negative (blue box) images, the classification difficulty increases as the spout gradually turns towards the middle. We observe the propagation process

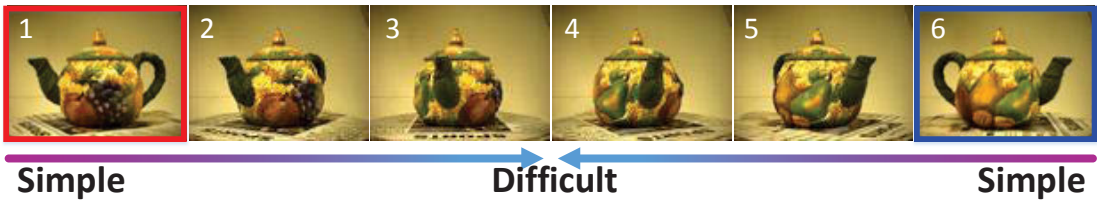

Figure 4.2: The toy example to illustrate our motivation. The orientation (left or right) of the spout in each image is to be determined. Labeled positive and negative examples are marked with red and blue boxes, respectively. The difficulties of these examples are illustrated by the two arrows below the images. The propagation sequence generated by the conventional methods Gong et al. [2014b]; Zhou and Bousquet [2003]; Zhu and Ghahramani [2002] is $\{1,6\} \rightarrow \{2,3,4,5\}$, while TLLT operates in the sequence $\{1,6\} \rightarrow \{2,5\} \rightarrow \{3\} \rightarrow \{4\}$. As a consequence, only the proposed TLLT can correctly classify the most difficult images 3 and 4.

from the beginning to the situation where all the images are exactly propagated. Our TLLT differs from existing methods Gong et al. [2014b]; Zhou and Bousquet [2003]; Zhu and Ghahramani [2002] in this process, which brings about completely different final iterative results from Gong et al. [2014b]; Zhou and Bousquet [2003]; Zhu and Ghahramani [2002]. Working on a fully connected graph, Gong et al. [2014b]; Zhou and Bousquet [2003]; Zhu and Ghahramani [2002] activate and immediately transmit the seed labels to all the unlabeled images $\{2,3,4,5\}$ in once propagation, as these images are directly linked to the labeled images 1 and 6 in the graph. In contrast, TLLT activates the unlabeled images in a three-stage sequence $\{2,5\} \rightarrow \{3\} \rightarrow \{4\}$ so that the ambiguous images 3 and 4 are handled later with a low error risk. As a result, the accuracy generated by TLLT is 100%, while the traditional methods Gong et al. [2014b]; Zhu and Ghahramani [2002] incorrectly classify the ambiguous 4th images, and Zhou and Bousquet [2003] fails to make accurate classifications on both the 3rd and 4th images.

4.3 Teaching-to-Learn Step

Suppose that a set of $n = l + u$ examples $\Psi = \{\mathbf{x}_1, \cdots, \mathbf{x}_l, \mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\}$ are given, where the first l elements constitute the labeled set $\mathcal{L}$ and the remaining u examples form the unlabeled set $\mathcal{U}$ with typically $l \ll u$. The purpose of label propagation is to iteratively propagate the label information from $\mathcal{L}$ to $\mathcal{U}$. In a generic multi-class setting, the label matrix is defined as $\mathbf{F} = \left(\mathbf{F}_1^\top, \cdots, \mathbf{F}_l^\top, \mathbf{F}_{l+1}^\top, \cdots, \mathbf{F}_n^\top\right)^\top$, where the i-th row vector $\mathbf{F}_i \in \{1,0\}^{1\times c}$ (c is the number of classes) satisfying $\sum_{j=1}^{c} \mathbf{F}_{ij} = 1$ denotes $\mathbf{x}_i$'s soft labels with $\mathbf{F}_{ij}$ being the probability of $\mathbf{x}_i$ belonging to the j-th class $\mathcal{C}_j$. In addition, we

define a set $\mathcal{S}$ to denote the curriculum which contains s unlabeled examples selected in one propagation iteration. When one iteration of label propagation is completed, $\mathcal{L}$ and $\mathcal{U}$ are updated by $\mathcal{L} := \mathcal{L} \cup \mathcal{S}$ and $\mathcal{U} := \mathcal{U} - \mathcal{S}$, respectively[1].

The adjacency (or affinity) matrix $\mathbf{W}$ of the graph $\mathcal{G}$ showed in Fig. 4.1(a) is formed by $\mathbf{W}_{ij} = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\xi^2)\right)$ (ξ is the Gaussian kernel width) if $\mathbf{x}_i$ and $\mathbf{x}_j$ are linked by an edge in $\mathcal{G}$, and $\mathbf{W}_{ij} = 0$ otherwise. Based upon $\mathbf{W}$, we introduce the diagonal degree matrix $\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{W}_{ij}$ and graph Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$.

4.3.1 Curriculum Selection

This section introduces a "teacher", which is essentially a teaching model, to decide the curriculum $\mathcal{S}$ for each iteration of propagation.

Above all, we define a random variable y_i associated with each example $\mathbf{x}_i$, and view the propagations on the graph as a Gaussian process Zhu et al. [2003b], which is modeled as a multivariate Gaussian distribution over the random variables $\mathbf{y} = (y_1, \cdots, y_n)^\top$, that is

$$p(\mathbf{y}) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top\left(\mathbf{L} + \mathbf{I}/\kappa^2\right)\mathbf{y}\right). \tag{4.1}$$

In Eq. (4.1), $\mathbf{L} + \mathbf{I}/\kappa^2$ (κ^2 is fixed to 100) is the regularized graph Laplacian Zhu et al. [2003b]. The modeled Gaussian process has a concise form $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ with its covariance matrix being $\mathbf{\Sigma} = (\mathbf{L} + \mathbf{I}/\kappa^2)^{-1}$.

Then, we define *reliability* and *discriminability* to assess the difficulty of the examples selected in $\mathcal{S} \subseteq \mathcal{U}$.

Definition 4.1. ***(Reliability)*** *A curriculum $\mathcal{S} \subseteq \mathcal{U}$ is reliable with respect to the labeled set $\mathcal{L}$ if the conditional entropy $H(\mathbf{y}_\mathcal{S}|\mathbf{y}_\mathcal{L})$ is small, where $\mathbf{y}_\mathcal{S}$ and $\mathbf{y}_\mathcal{L}$ represent the subvectors of $\mathbf{y}$ corresponding to the sets $\mathcal{S}$ and $\mathcal{L}$, respectively.*

Definition 4.2. ***(Discriminability)*** *A curriculum $\mathcal{S} \subseteq \mathcal{U}$ is discriminative if $\forall \mathbf{x}_i \in \mathcal{S}$, the value of*

$$\min_{j' \in \{1,\cdots,c\} \setminus \{q\}} \bar{T}(\mathbf{x}_i, \mathcal{C}_{j'}) - \min_{j \in \{1,\cdots,c\}} \bar{T}(\mathbf{x}_i, \mathcal{C}_j)$$

is large, where $\bar{T}(\mathbf{x}_i, \mathcal{C}_j)$ denotes the average commute time between $\mathbf{x}_i$ and all the labeled examples in class $\mathcal{C}_j$, and $q = \arg\min_{j \in \{1,\cdots,c\}} \bar{T}(\mathbf{x}_i, \mathcal{C}_j)$.

In Definition 4.1, we use reliability to measure the correlation between a curriculum $\mathcal{S}$ and the labeled set $\mathcal{L}$. The curriculum examples highly correlated to the labeled set

[1]Note that the notations such as l, u, s, $\mathcal{L}$, $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{C}_j$ are all related to the iteration number t. We drop the superscript (t) for simplicity if no confusion is incurred.

are obviously simple and reliable to classify. Such reliability is modeled as the entropy of $\mathcal{S}$ conditioned on $\mathcal{L}$, which implies that the simple examples in $\mathcal{S}$ should have small conditional entropy since they come as no "surprise" to the labeled examples. In Definition 4.2, we introduce the discriminability to investigate the tendency of $\mathbf{x}_i \in \mathcal{S}$ belonging to certain classes. An example $\mathbf{x}_i$ is easy to classify if it is significantly inclined to a category. Generally speaking, Definition 4.1 considers the hybrid relationship between $\mathcal{S}$ and $\mathcal{L}$, while Definition 4.2 associates the examples in $\mathcal{S}$ with the concrete class information, so they complement to each other in optimally selecting the simplest examples.

According to Definition 4.1, we aim to find a reliable set $\mathcal{S}$ such that it is most deterministic with respect to the labeled set $\mathcal{L}$, which is formulated as

$$\mathcal{S}^* = \arg\min_{\mathcal{S}\subseteq\mathcal{U}} H(\mathbf{y}_\mathcal{S}|\mathbf{y}_\mathcal{L}) := H(\mathbf{y}_{\mathcal{S}\cup\mathcal{L}}) - H(\mathbf{y}_\mathcal{L}). \tag{4.2}$$

Using the Gaussian process model in Eq. (4.1), we deduce Eq. (4.2) as follows

$$\begin{aligned}\mathcal{S}^* =& \arg\min_{\mathcal{S}\subseteq\mathcal{U}} \left(\frac{s+l}{2}\big(1+\ln 2\pi\big) + \frac{1}{2}\ln\left|\mathbf{\Sigma}_{\mathcal{S}\cup\mathcal{L},\mathcal{S}\cup\mathcal{L}}\right|\right) \\ &- \left(\frac{l}{2}\big(1+\ln 2\pi\big) + \frac{1}{2}\ln\left|\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}\right|\right) \\ =& \arg\min_{\mathcal{S}\subseteq\mathcal{U}} \frac{s}{2}\big(1+\ln 2\pi\big) + \frac{1}{2}\ln\frac{\left|\mathbf{\Sigma}_{\mathcal{S}\cup\mathcal{L},\mathcal{S}\cup\mathcal{L}}\right|}{\left|\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}\right|},\end{aligned}$$

where $\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}$, $\mathbf{\Sigma}_{\mathcal{S}\cup\mathcal{L},\mathcal{S}\cup\mathcal{L}}$ are submatrices of $\mathbf{\Sigma}$ associated with the corresponding subscripts. By further partitioning $\mathbf{\Sigma}_{\mathcal{S}\cup\mathcal{L},\mathcal{S}\cup\mathcal{L}} = \begin{pmatrix}\mathbf{\Sigma}_{\mathcal{S},\mathcal{S}} & \mathbf{\Sigma}_{\mathcal{S},\mathcal{L}} \\ \mathbf{\Sigma}_{\mathcal{L},\mathcal{S}} & \mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}\end{pmatrix}$, we have

$$\frac{\left|\mathbf{\Sigma}_{\mathcal{S}\cup\mathcal{L},\mathcal{S}\cup\mathcal{L}}\right|}{\left|\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}\right|} = \frac{\left|\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}\right|\left|\mathbf{\Sigma}_{\mathcal{S},\mathcal{S}} - \mathbf{\Sigma}_{\mathcal{S},\mathcal{L}}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}\mathbf{\Sigma}_{\mathcal{L},\mathcal{S}}\right|}{\left|\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}\right|} = \left|\mathbf{\Sigma}_{\mathcal{S}|\mathcal{L}}\right|,$$

where $\mathbf{\Sigma}_{\mathcal{S}|\mathcal{L}}$ is the covariance matrix of the conditional distribution $p(\mathbf{y}_\mathcal{S}|\mathbf{y}_\mathcal{L})$ and is naturally positive semidefinite. Therefore, minimizing $\ln\left|\mathbf{\Sigma}_{\mathcal{S}|\mathcal{L}}\right| = \ln\left|\mathbf{\Sigma}_{\mathcal{S},\mathcal{S}} - \mathbf{\Sigma}_{\mathcal{S},\mathcal{L}}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}\mathbf{\Sigma}_{\mathcal{L},\mathcal{S}}\right|$ is equivalent to minimizing $\mathrm{tr}\big(\mathbf{\Sigma}_{\mathcal{S},\mathcal{S}} - \mathbf{\Sigma}_{\mathcal{S},\mathcal{L}}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}\mathbf{\Sigma}_{\mathcal{L},\mathcal{S}}\big)$. Given a fixed s (we defer its determination to Section 4.4), the most reliable curriculum $\mathcal{S}$ is then found by

$$\mathcal{S}^* = \arg\min_{\mathcal{S}\subseteq\mathcal{U}} \mathrm{tr}\big(\mathbf{\Sigma}_{\mathcal{S},\mathcal{S}} - \mathbf{\Sigma}_{\mathcal{S},\mathcal{L}}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}\mathbf{\Sigma}_{\mathcal{L},\mathcal{S}}\big). \tag{4.3}$$

In Definition 4.2, the commute time between two examples $\mathbf{x}_i$ and $\mathbf{x}_j$ is the expected number of steps starting from $\mathbf{x}_i$, reaching $\mathbf{x}_j$, and then returning to $\mathbf{x}_i$ again,

which is computed by Qiu and Hancock [2007][1]:

$$T(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{n} h(\lambda_k)\big(u_{ki} - u_{kj}\big)^2. \tag{4.4}$$

In Eq. (4.4), $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are the eigenvalues of $\mathbf{L}$, and $\mathbf{u}_1, \cdots, \mathbf{u}_n$ are the associated eigenvectors; u_{ki} denotes the i-th element of $\mathbf{u}_k$; $h(\lambda_k) = 1/\lambda_k$ if $\lambda_k \neq 0$ and $h(\lambda_k) = 0$ otherwise. Based on Eq. (4.4), the average commute time between $\mathbf{x}_i$ and the examples in the j-th class $\mathcal{C}_j$ is calculated as

$$\bar{T}(\mathbf{x}_i, \mathcal{C}_j) = \frac{1}{|\mathcal{C}_j|} \sum_{\mathbf{x}_{i'} \in \mathcal{C}_j} T(\mathbf{x}_i, \mathbf{x}_{i'}). \tag{4.5}$$

Definition 4.2 characterizes the discriminability of an unlabeled example $\mathbf{x}_i \in \mathcal{U}$ as the average commute time difference between $\mathbf{x}_i$'s two closest classes $\mathcal{C}_{j_1}$ and $\mathcal{C}_{j_2}$, that is, $M(\mathbf{x}_i) = \bar{T}(\mathbf{x}_i, \mathcal{C}_{j_2}) - \bar{T}(\mathbf{x}_i, \mathcal{C}_{j_1})$. $\mathbf{x}_i$ is thought of as discriminative if it is significantly inclined to a certain class, namely it has a large $M(\mathbf{x}_i)$. From Definition 4.2, the most discriminative curriculum that consists of s discriminative examples is equivalently found by

$$\mathcal{S}^* = \arg \min_{\mathcal{S} = \{\mathbf{x}_{i_k} \in \mathcal{U}\}_{k=1}^{s}} \sum_{k=1}^{s} 1/M(\mathbf{x}_{i_k}). \tag{4.6}$$

Now, we propose that the simplest curriculum is not only reliable but also discriminative. Hence, we combine Eqs. (4.3) and (4.6) to arrive at the following curriculum selection criterion:

$$\mathcal{S}^* = \arg \min_{\mathcal{S} = \{\mathbf{x}_{i_k} \in \mathcal{U}\}_{k=1}^{s}} \operatorname{tr}\big(\mathbf{\Sigma}_{\mathcal{S},\mathcal{S}} - \mathbf{\Sigma}_{\mathcal{S},\mathcal{L}} \mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1} \mathbf{\Sigma}_{\mathcal{L},\mathcal{S}}\big) + \alpha \sum_{k=1}^{s} 1/M(\mathbf{x}_{i_k}),$$

where $\alpha > 0$ is the trade-off parameter.

Considering that the seed labels will be first propagated to the unlabeled examples which are direct neighbors of the labeled examples in $\mathcal{L}$, we collect such unlabeled examples in a set $\mathcal{B}$ with cardinality b. Since only s ($s < b$) distinct examples from $\mathcal{B}$ are needed, we introduce a binary selection matrix $\mathbf{S} \in \{1, 0\}^{b \times s}$ such that $\mathbf{S}^\top \mathbf{S} = \mathbf{I}_{s \times s}$ ($\mathbf{I}_{s \times s}$ denotes the $s \times s$ identity matrix). The element $\mathbf{S}_{ik} = 1$ means that the i-th example in $\mathcal{B}$ is selected as the k-th example in the curriculum $\mathcal{S}$. The orthogonality constraint $\mathbf{S}^\top \mathbf{S} = \mathbf{I}_{s \times s}$ imposed on $\mathbf{S}$ ensures that no repetitive example is included in $\mathcal{S}$.

[1] Strictly, the original commute time is $T(\mathbf{x}_i, \mathbf{x}_j) = vol(\mathcal{G}) \sum_{k=1}^{n} h(\lambda_k)(u_{ki} - u_{kj})^2$, where $vol(\mathcal{G})$ is a constant denoting the volume of graph $\mathcal{G}$. Here we drop this term since it will not influence our derivations.

We then reformulate problem (4.7) in the following matrix form:

$$\begin{aligned}\mathbf{S}^* = \arg\min_{\mathbf{S}}\ & \mathrm{tr}\big(\mathbf{S}^\top \mathbf{\Sigma}_{\mathcal{B},\mathcal{B}}\mathbf{S} - \mathbf{S}^\top \mathbf{\Sigma}_{\mathcal{B},\mathcal{L}}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}\mathbf{\Sigma}_{\mathcal{L},\mathcal{B}}\mathbf{S}\big) + \alpha \mathrm{tr}\big(\mathbf{S}^\top \mathbf{M}\mathbf{S}\big), \\ \text{s.t. } & \mathbf{S} \in \{1,0\}^{b\times s},\ \mathbf{S}^\top\mathbf{S} = \mathbf{I}_{s\times s},\end{aligned} \tag{4.7}$$

where $\mathbf{M} \in \mathbb{R}^{b\times b}$ is a diagonal matrix whose diagonal elements are $\mathbf{M}_{ii} = 1/M(\mathbf{x}_{i_k})$ for $k = 1, \cdots, b$. We notice that problem (4.7) falls into an integer program and is generally NP-hard. To make problem (4.7) tractable, we relax the discrete constraint $\mathbf{S} \in \{1,0\}^{b\times s}$ to be a continuous nonnegative constraint $\mathbf{S} \geq \mathbf{O}$. By doing so, we pursue to solve a simpler problem:

$$\begin{aligned}\mathbf{S}^* = \arg\min_{\mathbf{S}}\ & \mathrm{tr}\big(\mathbf{S}^\top \mathbf{R}\mathbf{S}\big), \\ \text{s.t. } & \mathbf{S} \geq \mathbf{O},\ \mathbf{S}^\top\mathbf{S} = \mathbf{I}_{s\times s},\end{aligned} \tag{4.8}$$

where $\mathbf{R} = \mathbf{\Sigma}_{\mathcal{B},\mathcal{B}} - \mathbf{\Sigma}_{\mathcal{B},\mathcal{L}}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}\mathbf{\Sigma}_{\mathcal{L},\mathcal{B}} + \alpha\mathbf{M}$ is a positive definite matrix.

4.3.2 Optimization

Note that Eq. (4.8) is a nonconvex optimization problem because of the orthogonal constraint. In fact, the feasible solution region is on the Stiefel manifold, which makes conventional gradient methods easily trapped into local minima. Instead, we adopt the method of partial augmented Lagrangian multiplier (PALM) Bertsekas [2014] to solve problem (4.8). Specifically, only the nonnegative constraint is incorporated into the objective function of the augmented Lagrangian expression, while the orthogonal constraint is explicitly retained and imposed on the subproblem for updating $\mathbf{S}$. As such, the $\mathbf{S}$-subproblem is a Stiefel-manifold constrained optimization problem, and can be efficiently solved by the curvilinear search method Wen and Yin [2013].

Updating S: By degenerating the nonnegative constraint and preserving the orthogonal constraint in problem (4.8), the partial augmented Lagrangian function is

$$L(\mathbf{S}, \mathbf{\Lambda}, \mathbf{T}, \sigma) := \mathrm{tr}\big(\mathbf{S}^\top\mathbf{R}\mathbf{S}\big) + \mathrm{tr}\big(\mathbf{\Lambda}^\top(\mathbf{S}-\mathbf{T})\big) + \frac{\sigma}{2}\big\|\mathbf{S}-\mathbf{T}\big\|_{\mathrm{F}}^2, \tag{4.9}$$

where $\mathbf{\Lambda} \in \mathbb{R}^{b\times s}$ is the Lagrangian multiplier, $\mathbf{T} \in \mathbb{R}^{b\times s}$ is a nonnegative matrix, and $\sigma > 0$ is the penalty coefficient. Therefore, $\mathbf{S}$ is updated by minimizing Eq. (4.9) subject to $\mathbf{S}^\top\mathbf{S} = \mathbf{I}_{s\times s}$ using the curvilinear search method Wen and Yin [2013] (see Algorithm 1).

In Algorithm 1, $\nabla L(\mathbf{S}) = 2\mathbf{R}\mathbf{S} + \mathbf{\Lambda} + \sigma(\mathbf{S}-\mathbf{T})$ is the gradient of $L(\mathbf{S}, \mathbf{\Lambda}, \mathbf{T}, \sigma)$ w.r.t. $\mathbf{S}$, and $L'\big(\bar{\mathbf{P}}(\tau)\big) = \mathrm{tr}\big(\nabla L(\mathbf{S})^\top \bar{\mathbf{P}}'(\tau)\big)$ calculates the derivate of $L(\mathbf{S}, \mathbf{\Lambda}, \mathbf{T}, \sigma)$ w.r.t. the step size τ, in which $\bar{\mathbf{P}}'(\tau) = -\big(\mathbf{I} + \frac{\tau}{2}\mathbf{A}\big)^{-1}\mathbf{A}\left(\frac{\mathbf{S}+\bar{\mathbf{P}}(\tau)}{2}\right)$. Algorithm 1 works by finding the gradient of L in the tangent plane of the manifold at the point $\mathbf{S}^{(iter)}$

Algorithm 1 A curvilinear search method for solving $\mathbf{S}$-subproblem

1: **Input:** $\mathbf{S}$ satisfying $\mathbf{S}^\top\mathbf{S} = \mathbf{I}$, $\varepsilon = 10^{-5}$, $\tau = 10^{-3}$, $\vartheta = 0.2$, $\eta = 0.85$, $Q = 1$, $\nu = L(\mathbf{S})$, $iter = 0$
2: **repeat**
3: // Define searching path $\bar{\mathbf{P}}(\tau)$ and step size on the Stiefel manifold
4: $\mathbf{A} = \nabla L(\mathbf{S}) \cdot \mathbf{S}^\top - \mathbf{S} \cdot (\nabla L(\mathbf{S}))^\top$;
5: **repeat**
6: $\bar{\mathbf{P}}(\tau) = \left(\mathbf{I} + \frac{\tau}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I} - \frac{\tau}{2}\mathbf{A}\right)\mathbf{S}$;
7: $\tau := \vartheta \cdot \tau$;
8: // Check Barzilai-Borwein condition
9: **until** $L(\bar{\mathbf{P}}(\tau)) \leq \nu - \tau L'(\bar{\mathbf{P}}(0))$
10: // Update variables
11: $\mathbf{S} := \bar{\mathbf{P}}(\tau)$;
12: $Q := \eta Q + 1$; $\nu := (\eta Q \nu + L(\mathbf{S}))/Q$;
13: $iter := iter + 1$;
14: **until** $\|\nabla L(\mathbf{S})\|_{\mathrm{F}} < \varepsilon$
15: **Output:** $\mathbf{S}$

(Line 4), based on which a curve is obtained on the manifold that proceeds along the projected negative gradient (Line 6). A curvilinear search is then made along the curve towards the optimal $\mathbf{S}^{(iter+1)}$.

Algorithm 1 preserves the orthogonal constraint through the skew-symmetric matrix $\mathbf{A}$ based Cayley transformation $\left(\mathbf{I}+\frac{\tau}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I}-\frac{\tau}{2}\mathbf{A}\right)$, which transforms $\mathbf{S}$ to $\bar{\mathbf{P}}(\tau)$ to guarantee that $\bar{\mathbf{P}}(\tau)^\top\bar{\mathbf{P}}(\tau) = \mathbf{I}$ always holds. The step size τ is adaptively determined by the Barzilai-Borwein method Fletcher [2005].

Updating T: In Eq. (4.9), $\mathbf{T}$ is the auxiliary variable to enforce $\mathbf{S}$ nonnegative, whose update is the same as that in the traditional augmented Lagrangian multiplier (ALM) method, namely $\mathbf{T}_{ik} = \max(0,\ \mathbf{S}_{ik} + \mathbf{\Lambda}_{ik}/\sigma)$.

We summarize the complete optimization procedure of PALM in Algorithm 2, by which a local minimizer can be efficiently obtained. PALM inherits the merits of conventional ALM such as the non-necessity for driving the penalty coefficient to infinity, and is also guaranteed to be convergent Conn et al. [1996].

Note that the solution $\mathbf{S}^*$ generated by Algorithm 2 is continuous, which does not comply with the original binary constraint in problem (4.7). Therefore, we discretize $\mathbf{S}^*$ to binary values via a simple greedy procedure. In detail, we find the largest element in $\mathbf{S}^*$, and record its row and column; then from the unrecorded columns and rows we search the largest element and mark it again; this procedure repeats until s elements are found. The rows of these s elements indicate the selected simplest examples to be propagated.

Algorithm 2 PALM for solving problem (4.8)

1: **Input:** $\mathbf{R}$, $\mathbf{S}$ satisfying $\mathbf{S}^\top\mathbf{S} = \mathbf{I}$, $\mathbf{\Lambda} = \mathbf{O}$, $\sigma = 1$, $\rho = 1.2$, $iter = 0$
2: **repeat**
3: // Compute $\mathbf{T}$
4: $\mathbf{T}_{ik} = \max(0,\ \mathbf{S}_{ik} + \mathbf{\Lambda}_{ik}/\sigma)$;
5: // Update $\mathbf{S}$ by minimizing Eq. (4.9) using Algorithm 1
6: $\mathbf{S} := \arg\min_{\mathbf{S}^\top\mathbf{S}=\mathbf{I}_{s\times s}} \mathrm{tr}(\mathbf{S}^\top\mathbf{R}\mathbf{S}) + \mathrm{tr}\left[\mathbf{\Lambda}^\top(\mathbf{S}-\mathbf{T})\right] + \frac{\sigma}{2}\|\mathbf{S}-\mathbf{T}\|_{\mathrm{F}}^2$;
7: // Update variables
8: $\mathbf{\Lambda}_{ik} := \max(0, \mathbf{\Lambda}_{ik} - \sigma\mathbf{S}_{ik})$; $\sigma := \min(\rho\sigma, 10^{10})$; $iter := iter + 1$;
9: **until** Convergence
10: **Output:** $\mathbf{S}^*$

In Algorithm 1, the complexities for obtaining $\mathbf{A}$, inverting $\mathbf{I} + \frac{\tau}{2}\mathbf{A}$, and computing the value of objective function $L(\mathbf{S}, \mathbf{\Lambda}, \mathbf{T}, \sigma)$ are $\mathcal{O}(b^2 s)$, $\mathcal{O}(b^3)$, and $\mathcal{O}(bs^2)$, respectively. Therefore, suppose the Lines 5∼9 are repeated T_1 times, and the Lines 2∼14 are iterated T_2 times, then the complexity of Algorithm 1 is $\mathcal{O}\left([b^2 s + (b^3 + bs^2)T_1]T_2\right)$. As a result, the entire PALM takes $\mathcal{O}\left([b^2 s + (b^3 + bs^2)T_1]T_2 T_3\right)$ complexity where T_3 is the iteration times of Lines 2∼9 in Algorithm 2. Because the established k-NN graph $\mathcal{G}$ is very sparse, the amount of examples directly linked to the labeled set (*i.e.*, b) will not be extremely large. Besides, s will also be very small since we only select a small proportion of unlabeled examples in each propagation. Therefore, the computational cost is acceptable though the complexity of our optimization is approximately cubic to b and quadric to s.

4.4 Learning-to-Teach Step

This section first introduces a "learner", which is a propagation model, and then elaborates how the learning feedback is established for the subsequent teaching.

Suppose that the curriculum in the t-th propagation iteration is $\mathcal{S}^{(t)}$. The learner "learns" (*i.e.*, labels) the $s^{(t)}$ examples in $\mathcal{S}^{(t)}$ by propagating the labels of the labeled examples in $\mathcal{L}^{(t)}$ to $\mathcal{S}^{(t)}$. We adopt the following iterative propagation model Zhu and Ghahramani [2002]:

$$\mathbf{F}_i^{(t)} := \begin{cases} \mathbf{F}_i^{(0)}, & \mathbf{x}_i \in \mathcal{L}^{(0)} \\ \mathbf{P}_{i\cdot}\mathbf{F}^{(t-1)}, & \mathbf{x}_i \in \mathcal{S}^{(1:t-1)} \cup \mathcal{S}^{(t)} \end{cases} \tag{4.10}$$

where $\mathbf{P}_{i\cdot}$ represents the i-th row of the *transition matrix* $\mathbf{P}$ calculated by $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$, and $\mathcal{S}^{(1:t-1)}$ denotes the set $\mathcal{S}^{(1)} \cup \cdots \cup \mathcal{S}^{(t-1)}$. Eq. (4.10) reveals that the labels of the t-th curriculum $\mathcal{S}^{(t)}$ along with the previously learned examples $\mathcal{S}^{(1:t-1)}$ will change during the propagation, while the labels of the initially labeled examples in $\mathcal{L}^{(0)}$ are

clamped, as suggested by Zhu and Ghahramani [2002]. The initial state for $\mathbf{x}_i$'s label vector $\mathbf{F}_i^{(0)}$ is

$$\mathbf{F}_i^{(0)} := \begin{cases} \underbrace{(1/c, \cdots, 1/c)}_{c}, & \mathbf{x}_i \in \mathcal{U}^{(0)} \\ \left(0, \cdots, \underset{\substack{\downarrow \\ j-th\ element}}{1}, \cdots, 0\right), & \mathbf{x}_i \in \mathcal{C}_j \in \mathcal{L}^{(0)} \end{cases} \tag{4.11}$$

The formulations of Eqs. (4.10) and (4.11) maintain the probability interpretation $\sum_{j=1}^{c} \mathbf{F}_{ij}^{(t)} = 1$ for any example $\mathbf{x}_i$ and all iterations $t = 0, 1, 2, \cdots$.

After the t-th propagation iteration, the learner should deliver a learning feedback to the teacher and assist the teacher to determine the $(t+1)$-th curriculum $\mathcal{S}^{(t+1)}$. If the t-th learning result is correct, the teacher may assign a "heavier" curriculum to the learner for the next propagation. In this sense, the teacher should also "learn" the learner's feedback to arrange the proper $(t+1)$-th curriculum, which is a "learning-to-teach" mechanism. However, the correctness of the propagated labels generated by the t-th iteration remains unknown to the teacher, so the learning confidence is explored to blindly evaluate the t-th learning performance.

To be specific, we restrict the learning confidence to the range $[0, 1]$, in which 1 is achieved if all the curriculum examples in $\mathcal{S}^{(t)}$ obtain definite label vectors, and 0 is reached if the curriculum examples are assigned similar label values over all the possible classes. For example, suppose we have $c = 3$ classes in total, then for a single example $\mathbf{x}_i$, it is "well-learned" if it has a label vector $\mathbf{F}_i = [1, 0, 0]$, $[0, 1, 0]$, or $[0, 0, 1]$ which means that $\mathbf{x}_i$ definitely belongs to the class 1, 2 or 3, respectively. In contrast, if $\mathbf{x}_i$'s label vector is $\mathbf{F}_i = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$, it will be an "ill-learned" example because $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ cannot provide any cue for determining its class. Therefore, we integrate the learning confidence of all the examples in $\mathcal{S}^{(t)}$ and define a *Learning Evaluation Function* $g(\mathbf{F}_{\mathcal{S}^{(t)}}) : \mathbb{R}^{s^{(t)} \times c} \to \mathbb{R}$ to assess the t-th propagation quality, based on which the number of examples $s^{(t+1)}$ for the $(t+1)$-th iteration can be adaptively decided. Here $\mathbf{F}_{\mathcal{S}^{(t)}}$ denotes the obtained label matrix of the t-th curriculum $\mathcal{S}^{(t)}$. A valid $g(\mathbf{F}_{\mathcal{S}^{(t)}})$ is formally described by Definition 4.3.

Definition 4.3. ***(Learning Evaluation Function)*** *A learning evaluation function $g(\mathbf{F}_{\mathcal{S}^{(t)}}) : \mathbb{R}^{s^{(t)} \times c} \to \mathbb{R}$ assesses the t-th learning confidence revealed by the label matrix $\mathbf{F}_{\mathcal{S}^{(t)}}$, which satisfies: 1) $0 \leq g(\mathbf{F}_{\mathcal{S}^{(t)}}) \leq 1$; 2) $g(\mathbf{F}_{\mathcal{S}^{(t)}}) \to 1$ if $\forall \mathbf{x}_i \in \mathcal{S}^{(t)}$, $\mathbf{F}_{ij} \to 1$ while $\mathbf{F}_{ik} \to 0$ for $k \neq j$; and $g(\mathbf{F}_{\mathcal{S}^{(t)}}) \to 0$ if $\mathbf{F}_{ij} \to 1/c$ for $i = 1, \cdots, s^{(t)}$, $j = 1, \cdots, c$.*

Definition 4.3 suggests that a large $g(\mathbf{F}_{\mathcal{S}^{(t)}})$ can be achieved if the label vectors $\mathbf{F}_{i_k}$ $(k = 1, 2, \cdots, s^{(t)})$ in $\mathbf{F}_{\mathcal{S}^{(t)}}$ are almost binary. In contrast, the ambiguous label vectors $\mathbf{F}_{i_k}$ with all entries around $1/c$ cause $\mathbf{F}_{\mathcal{S}^{(t)}}$ to obtain a rather low confidence evaluation

$g(\mathbf{F}_{\mathcal{S}^{(t)}})$. According to Definition 4.3, we propose two learning evaluation functions by respectively utilizing $\mathbf{F}_{\mathcal{S}^{(t)}}$'s norm and entropy:

$$g_1(\mathbf{F}_{\mathcal{S}^{(t)}}) = \frac{2}{1+\exp\left[-\gamma_1\left(\|\mathbf{F}_{\mathcal{S}^{(t)}}\|_{\mathrm{F}}^2 - s^{(t)}/c\right)\right]} - 1, \tag{4.12}$$

$$\begin{aligned} g_2(\mathbf{F}_{\mathcal{S}^{(t)}}) &= \exp\left[-\gamma_2 \frac{1}{s^{(t)}} H\left(\mathbf{F}_{\mathcal{S}^{(t)}}\right)\right] \\ &= \exp\left[\frac{\gamma_2}{s^{(t)}} \sum_{k=1}^{s^{(t)}} \sum_{j=1}^{c} \left(\mathbf{F}_{\mathcal{S}^{(t)}}\right)_{kj} \log_c \left(\mathbf{F}_{\mathcal{S}^{(t)}}\right)_{kj}\right], \end{aligned} \tag{4.13}$$

where γ_1 and γ_2 are the parameters controlling the learning rate. Increasing γ_1 in Eq. (4.12) or decreasing γ_2 in Eq. (4.13) will incorporate more examples into one curriculum. However, the complexity analysis in Section 4.3.2 reveals that including too many examples in one curriculum (*i.e.* increasing s) will make solving problem (4.8) very inefficient.

It can be easily verified that both (4.12) and (4.13) satisfy the two requirements in Definition 4.3. For (4.12), we may write $g_1(\mathbf{F}_{\mathcal{S}^{(t)}})$ as $g_1(\mathbf{F}_{\mathcal{S}^{(t)}}) = 2\bar{g}_1(\mathbf{F}_{\mathcal{S}^{(t)}}) - 1$ where $\bar{g}_1(\mathbf{F}_{\mathcal{S}^{(t)}}) = \frac{1}{1+\exp\left[-\gamma_1\left(\left\|\mathbf{F}_{\mathcal{S}^{(t)}}\right\|_{\mathrm{F}}^2 - s^{(t)}/c\right)\right]}$ is a monotonically increasing logistic function with respect to $\|\mathbf{F}_{\mathcal{S}^{(t)}}\|_{\mathrm{F}}$. Therefore, $\bar{g}_1(\mathbf{F}_{\mathcal{S}^{(t)}})$ reaches its minimum value $1/2$ when $\|\mathbf{F}_{\mathcal{S}^{(t)}}\|_{\mathrm{F}}^2 = s^{(t)}/c$, which means that all the elements in $\mathbf{F}_{\mathcal{S}^{(t)}}$ equal to $1/c$. The value of $\bar{g}_1(\mathbf{F}_{\mathcal{S}^{(t)}})$ gradually approaches 1 when $\|\mathbf{F}_{\mathcal{S}^{(t)}}\|_{\mathrm{F}}$ becomes larger, which requires that all the row vectors in $\mathbf{F}_{\mathcal{S}^{(t)}}$ are almost binary. Therefore, $\bar{g}_1(\mathbf{F}_{\mathcal{S}^{(t)}}) \in [1/2, 1)$ and $g_1(\mathbf{F}_{\mathcal{S}^{(t)}})$ maps $\bar{g}_1(\mathbf{F}_{\mathcal{S}^{(t)}})$ to $[0, 1)$ so that the two requirements in Definition 4.3 are satisfied.

For (4.13), it is evident that the entropy $H\left(\mathbf{F}_{\mathcal{S}^{(t)}}\right) = -\sum_k \sum_j \left(\mathbf{F}_{\mathcal{S}^{(t)}}\right)_{kj} \log_c \left(\mathbf{F}_{\mathcal{S}^{(t)}}\right)_{kj}$ falls into the range $[0, 1]$, where 0 is obtained when each row of $\mathbf{F}_{\mathcal{S}^{(t)}}$ is a $\{0,1\}$-binary vector with only one element 1, and 1 is attained if every element in $\mathbf{F}_{\mathcal{S}^{(t)}}$ is $1/c$. As a result, $g_2(\mathbf{F}_{\mathcal{S}^{(t)}})$ is valid as a learning evaluation function.

Based on a defined learning evaluation function, the number of examples included in the $(t+1)$-th curriculum is:

$$s^{(t+1)} = \left\lceil b^{(t+1)} \cdot g(\mathbf{F}_{\mathcal{S}^{(t)}}) \right\rceil, \tag{4.14}$$

where $b^{(t+1)}$ is the size of set $\mathcal{B}^{(t+1)}$ in the $(t+1)$-th iteration, $\lceil \cdot \rceil$ rounds up the element to the nearest integer, and $g(\cdot)$ can be either $g_1(\cdot)$ or $g_2(\cdot)$. Note that $g(\cdot)$ is simply set to a very small number, *e.g.* 0.05, for the first propagation, because no feedback is available at the beginning of the propagation process.

TLLT proceeds until all the unlabeled examples are learned, and the obtained label matrix is denoted as $\bar{\mathbf{F}}$. Then we set $\bar{\mathbf{F}}^{(0)} := \bar{\mathbf{F}}$ and use the following iterative formula to drive the entire propagation process to the steady state:

$$\bar{\mathbf{F}}^{(t)} = \theta \mathbf{P} \bar{\mathbf{F}}^{(t-1)} + (1-\theta)\bar{\mathbf{F}}, \tag{4.15}$$

where $\theta > 0$ is the weighting parameter balancing the labels propagated from other examples, and $\bar{\mathbf{F}}$ that is produced by the teaching-to-learn and learning-to-teach process. We set $\theta = 0.05$ to enforce the final result to maximally preserve the labels generated by teaching and learning. By employing the Perron-Frobenius Theorem Golub and Loan [1996], we take the limit of $\bar{\mathbf{F}}^{(t)}$ as follows

$$\begin{aligned}\bar{\mathbf{F}}^{*} &= \lim_{t\to\infty} \bar{\mathbf{F}}^{(t)} = \lim_{t\to\infty} (\theta\mathbf{P})^{t}\bar{\mathbf{F}} + (1-\theta)\sum\nolimits_{i=0}^{t-1}(\theta\mathbf{P})^{i}\bar{\mathbf{F}} \\ &= (1-\theta)(\mathbf{I}-\theta\mathbf{P})^{-1}\bar{\mathbf{F}}. \end{aligned} \tag{4.16}$$

Eventually, $\mathbf{x}_i$ is assigned to the j^*-th class such that $j^* = \arg\max_{j\in\{1,\cdots,c\}} \bar{\mathbf{F}}^{*}_{ij}$.

4.5 Efficient Computations

The computational bottlenecks of TLLT are the calculation of pairwise commute time in Eq. (4.4) and the updating of $\mathbf{\Sigma}^{-1}_{\mathcal{L},\mathcal{L}}$ in Eq. (4.3) for each propagation. The former can be accelerated by applying the Nyström approximation Fowlkes et al. [2004] to efficiently compute $\mathbf{L}$'s eigenvectors, while the latter can be efficiently implemented via matrix permutation and blockwise inversion Hager [1989].

4.5.1 Commute Time

Note that the Eq. (4.4) involves computing the eigenvectors of $\mathbf{L}$, which is time-consuming when n is large. Here we apply the Nyström approximation to reduce the computational burden. Specifically, we uniformly sample q ($q = 10\% \cdot n$ throughout this chapter) rows/columns of the original $\mathbf{L}$ to form a submatrix $\mathbf{L}_{q,q}$, and then $\mathbf{L}$ can be approximated by $\tilde{\mathbf{L}} = \mathbf{L}_{n,q}\mathbf{L}^{-1}_{q,q}\mathbf{L}_{q,n}$, where $\mathbf{L}_{n,q}$ represents the $n \times q$ block of $\mathbf{L}$ and $\mathbf{L}_{q,n} = \mathbf{L}^{\top}_{n,q}$. By defining $\mathbf{V} \in \mathbb{R}^{q\times q}$ as an orthogonal matrix, $\tilde{\mathbf{\Theta}}$ as a $q \times q$ diagonal matrix, and

$$\mathbf{U} = \begin{pmatrix}\mathbf{L}_{q,q}\\ \mathbf{L}_{n-q,q}\end{pmatrix}\mathbf{L}^{-1/2}_{q,q}\mathbf{V}\tilde{\mathbf{\Theta}}^{-1/2}, \tag{4.17}$$

we have (4.18) according to Fowlkes et al. [2004]:

$$\begin{aligned}\tilde{\mathbf{L}} &= \mathbf{L}_{n,q}\mathbf{L}^{-1}_{q,q}\mathbf{L}_{q,n} = \begin{pmatrix}\mathbf{L}_{q,q}\\ \mathbf{L}_{n-q,q}\end{pmatrix}\mathbf{L}^{-1}_{q,q}\begin{pmatrix}\mathbf{L}^{\top}_{q,q} & \mathbf{L}^{\top}_{n-q,q}\end{pmatrix} \\ &= \begin{pmatrix}\mathbf{L}_{q,q}\\ \mathbf{L}_{n-q,q}\end{pmatrix}\mathbf{L}^{-1/2}_{q,q}\mathbf{V}\tilde{\mathbf{\Theta}}^{-1/2}\tilde{\mathbf{\Theta}}\tilde{\mathbf{\Theta}}^{-1/2}\mathbf{V}^{\top}\mathbf{L}^{-1/2}_{q,q}\begin{pmatrix}\mathbf{L}^{\top}_{q,q} & \mathbf{L}^{\top}_{n-q,q}\end{pmatrix}\cdot \\ &= \mathbf{U}\tilde{\mathbf{\Theta}}\mathbf{U}^{\top}\end{aligned} \tag{4.18}$$

Since $\tilde{\mathbf{L}}$ is positive semi-definite, then according to Eq. (4.17), we require

$$\begin{aligned}\mathbf{I} &= \mathbf{U}^\top \mathbf{U} \\ &= \tilde{\mathbf{\Theta}}^{-1/2}\mathbf{V}^\top \mathbf{L}_{q,q}^{-1/2} \begin{pmatrix}\mathbf{L}_{q,q}^\top & \mathbf{L}_{n-q,q}^\top\end{pmatrix} \cdot \begin{pmatrix}\mathbf{L}_{q,q} \\ \mathbf{L}_{n-q,q}\end{pmatrix} \mathbf{L}_{q,q}^{-1/2}\mathbf{V}\tilde{\mathbf{\Theta}}^{-1/2}.\end{aligned} \tag{4.19}$$

Multiplying from the left by $\mathbf{V}\tilde{\mathbf{\Theta}}^{1/2}$ and from the right by $\tilde{\mathbf{\Theta}}^{1/2}\mathbf{V}^\top$, we have

$$\begin{aligned}\mathbf{V}\tilde{\Theta}\mathbf{V}^\top &= \mathbf{L}_{q,q}^{-1/2} \begin{pmatrix}\mathbf{L}_{q,q}^\top & \mathbf{L}_{n-q,q}^\top\end{pmatrix} \cdot \begin{pmatrix}\mathbf{L}_{q,q} \\ \mathbf{L}_{n-q,q}\end{pmatrix} \mathbf{L}_{q,q}^{-1/2} \\ &= \mathbf{L}_{q,q} + \mathbf{L}_{q,q}^{-1/2}\mathbf{L}_{n-q,q}^\top \mathbf{L}_{n-q,q}\mathbf{L}_{q,q}^{-1/2}.\end{aligned} \tag{4.20}$$

Therefore, by comparing (4.18) and (4.20) we know that the matrix $\mathbf{U}$ containing all the eigenvectors $\mathbf{u}_i$ $(i = 1, \cdots, n)$ can be obtained by conducting singular value decomposition (SVD) on $\mathbf{L}_{q,q} + \mathbf{L}_{q,q}^{-1/2}\mathbf{L}_{n-q,q}^\top \mathbf{L}_{n-q,q}\mathbf{L}_{q,q}^{-1/2}$, and then plugging $\mathbf{V}$ and $\tilde{\mathbf{\Theta}}$ to (4.17).

The complexity for computing the commute time between examples via Nyström approximation is $\mathcal{O}(q^3)$, which is caused by finding $\mathbf{L}_{q,q}^{-1/2}$ in (4.20) and the SVD on $\mathbf{L}_{q,q} + \mathbf{L}_{q,q}^{-1/2}\mathbf{L}_{n-q,q}^\top \mathbf{L}_{n-q,q}\mathbf{L}_{q,q}^{-1/2}$. This significantly reduces the cost of directly solving the original eigen-system that takes $\mathcal{O}(n^3)$ $(n \gg q)$ complexity.

4.5.2 Updating $\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}$

It is very inefficient if $\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}$ in Eq. (4.3) is computed from scratch in each propagation, so we develop an incremental way to update the inversion based on the blockwise inversion equation Hager [1989] and matrix permutation technique.

Suppose the updated kernel matrix on the labeled set $\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}$ is permuted by $\widehat{\mathbf{\Sigma}}_{\mathcal{L},\mathcal{L}}$, namely $\widehat{\mathbf{\Sigma}}_{\mathcal{L},\mathcal{L}} = perm(\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}) = \mathbf{\Delta}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}\mathbf{\Delta}^\top$ and $\mathbf{\Delta}$ is the orthogonal binary permutation matrix producing the permutation on the rows or columns of the matrix $\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}$ Petersen and Pedersen, then we have the following lemma:

Lemma 4.4. *The inverse of the updated $\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}$ equals to the inverse of its permutation $\widehat{\mathbf{\Sigma}}_{\mathcal{L},\mathcal{L}} = \mathbf{\Delta}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}\mathbf{\Delta}^\top$ followed by the reversed permutation, namely $\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1} = perm^{-1}(\widehat{\mathbf{\Sigma}}_{\mathcal{L},\mathcal{L}}^{-1})$ where $perm^{-1}(\widehat{\mathbf{\Sigma}}_{\mathcal{L},\mathcal{L}}^{-1}) = \mathbf{\Delta}^\top\widehat{\mathbf{\Sigma}}_{\mathcal{L},\mathcal{L}}^{-1}\mathbf{\Delta}$.*

Proof. Note that $\mathbf{\Delta}$ is an orthogonal matrix, so it is easy to verify that $perm^{-1}(\widehat{\mathbf{\Sigma}}_{\mathcal{L},\mathcal{L}}^{-1}) = \mathbf{\Delta}^\top(\mathbf{\Delta}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}\mathbf{\Delta}^\top)^{-1}\mathbf{\Delta} = \mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}$. This completes the proof. □

Therefore, we permute the new kernel matrix on the labeled set after each propagation, so that the submatrices corresponding to the previous labeled set $\mathcal{L}$

and the current curriculum $\mathcal{S}$ are respectively arranged together, which is formally represented as

$$perm(\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}) = \mathbf{\Delta}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}\mathbf{\Delta}^\top = \widehat{\mathbf{\Sigma}}_{\mathcal{L},\mathcal{L}} := \begin{pmatrix} \mathbf{\Sigma}_{\mathcal{L},\mathcal{L}} & \mathbf{\Sigma}_{\mathcal{L},\mathcal{S}} \\ \mathbf{\Sigma}_{\mathcal{S},\mathcal{L}} & \mathbf{\Sigma}_{\mathcal{S},\mathcal{S}} \end{pmatrix}. \tag{4.21}$$

The permutation on $\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}$ aims to reformat $\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}$ so that its rows/columns are arranged according to the ascending order of the indices of the updated labeled examples. If we do not permute $\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}$ and directly plug the un-permuted $\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}$ into Eq. (4.7) for the next propagation, the generated $\mathbf{S}^*$ will not render the real indices of curriculum examples in the entire dataset $\Psi = \mathcal{L} \cup \mathcal{U}$. This permutation renders the matrix $\widehat{\mathbf{\Sigma}}_{\mathcal{L},\mathcal{L}}$, of which the inverse can be easily computed via Eq. (4.22) as suggested by the blockwise inversion equation Hager [1989].

$$\widehat{\mathbf{\Sigma}}_{\mathcal{L},\mathcal{L}}^{-1} := \begin{pmatrix} \mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1} + \mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}\mathbf{\Sigma}_{\mathcal{L},\mathcal{S}}(\mathbf{\Sigma}_{\mathcal{S},\mathcal{S}} - \mathbf{\Sigma}_{\mathcal{S},\mathcal{L}}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}\mathbf{\Sigma}_{\mathcal{L},\mathcal{S}})^{-1}\mathbf{\Sigma}_{\mathcal{S},\mathcal{L}}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1} & -\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}\mathbf{\Sigma}_{\mathcal{L},\mathcal{S}}(\mathbf{\Sigma}_{\mathcal{S},\mathcal{S}} - \mathbf{\Sigma}_{\mathcal{S},\mathcal{L}}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}\mathbf{\Sigma}_{\mathcal{L},\mathcal{S}})^{-1} \\ -(\mathbf{\Sigma}_{\mathcal{S},\mathcal{S}} - \mathbf{\Sigma}_{\mathcal{S},\mathcal{L}}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}\mathbf{\Sigma}_{\mathcal{L},\mathcal{S}})^{-1}\mathbf{\Sigma}_{\mathcal{S},\mathcal{L}}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1} & (\mathbf{\Sigma}_{\mathcal{S},\mathcal{S}} - \mathbf{\Sigma}_{\mathcal{S},\mathcal{L}}\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}\mathbf{\Sigma}_{\mathcal{L},\mathcal{S}})^{-1} \end{pmatrix}. \tag{4.22}$$

Therefore, according to Lemma 4.4 we can compute the inverse of the new kernel matrix on the labeled set by $\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1} = \mathbf{\Delta}^\top \widehat{\mathbf{\Sigma}}_{\mathcal{L},\mathcal{L}}^{-1} \mathbf{\Delta}$.

In the above manipulations, we only need to invert an $s \times s$ matrix in (4.22), which is much more efficient than inverting the original $l \times l$ ($l \gg s$ in later propagations) matrix. Moreover, s will not be quite large since only a small proportion of unlabeled examples are incorporated into the curriculum per propagation. Therefore, the $\mathbf{\Sigma}_{\mathcal{L},\mathcal{L}}^{-1}$ can be updated efficiently.

4.6 Robustness Analysis

For graph-based learning algorithms, the choice of the Gaussian kernel width ξ is critical to achieving good performance. Unfortunately, tuning this parameter is usually nontrivial because a slight perturbation of ξ will lead to a big change in the model output. Several methods Karasuyama and Mamitsuka [2013a]; Zhu et al. [2003a] have been proposed to decide the optimal ξ via entropy minimization Zhu et al. [2003a] or local reconstruction Karasuyama and Mamitsuka [2013a]. However, they are heuristic and not guaranteed to always obtain the optimal ξ. Here we demonstrate that TLLT is very robust to the variation of ξ, which implies that ξ in our method can be easily tuned.

Theorem 4.5. *Suppose that the adjacency matrix $\tilde{\mathbf{W}}$ of graph $\mathcal{G}$ is perturbed from $\mathbf{W}$ due to the variation of ξ, such that for some $\delta > 1$, $\forall i, j$, $\mathbf{W}_{ij}/\delta \leq \tilde{\mathbf{W}}_{ij} \leq \delta \mathbf{W}_{ij}$. The deviation of the t-th propagation result on the initial unlabeled examples $\tilde{\mathbf{F}}_{\mathcal{U}}^{(t)}$ from the*

accurate $\mathbf{F}_{\mathcal{U}}^{(t)}$[1] *is bounded by* $\left\|\tilde{\mathbf{F}}_{\mathcal{U}}^{(t)} - \mathbf{F}_{\mathcal{U}}^{(t)}\right\|_{\mathrm{F}} \leq \mathcal{O}(\delta^2 - 1)$.

Proof. Given $\mathbf{W}_{ij}/\delta \leq \tilde{\mathbf{W}}_{ij} \leq \delta \mathbf{W}_{ij}$ for $\delta > 1$, the bound for the (i, j)-th element in the perturbed transition matrix $\tilde{\mathbf{P}}$ is $\mathbf{P}_{ij}/\delta^2 \leq \tilde{\mathbf{P}}_{ij} \leq \delta^2 \mathbf{P}_{ij}$. Besides, by recalling that $\mathbf{P}_{ij} \leq 1$ as $\mathbf{P}$ has been row normalized, so the difference between $\mathbf{P}_{ij}$ and $\tilde{\mathbf{P}}_{ij}$ satisfies

$$\left|\tilde{\mathbf{P}}_{ij} - \mathbf{P}_{ij}\right| \leq (\delta^2 - 1)\mathbf{P}_{ij} \leq \delta^2 - 1. \tag{4.23}$$

For the ease of analysis, we rewrite the learning model (4.10) in a more compact form. Suppose $\mathbf{Q}_{\mathcal{S}^{(1:t)}} \in \{0, 1\}^{u^{(0)} \times u^{(0)}}$ ($\mathcal{S}^{(1:t)} = \mathcal{S}^{(1)} \cup \cdots \cup \mathcal{S}^{(t)}$ as defined in Section 4.4) is a binary diagonal matrix where the diagonal elements are set to 1 if they correspond to the examples in the set $\mathcal{S}^{(1:t)}$, then Eq. (4.10) can be reformulated as

$$\mathbf{F}_{\mathcal{U}}^{(t)} = \mathbf{Q}_{\mathcal{S}^{(1:t)}} \mathbf{P}_{\mathcal{U},\cdot} \mathbf{F}^{(t-1)} + (\mathbf{I} - \mathbf{Q}_{\mathcal{S}^{(1:t)}}) \mathbf{F}_{\mathcal{U}}^{(t-1)}, \tag{4.24}$$

where $\mathbf{P}_{\mathcal{U},\cdot} = \begin{pmatrix} \mathbf{P}_{\mathcal{U},\mathcal{L}} & \mathbf{P}_{\mathcal{U},\mathcal{U}} \end{pmatrix}$ denotes the rows in $\mathbf{P}$ corresponding to $\mathcal{U}$. Similarly, the perturbed $\tilde{\mathbf{F}}_{\mathcal{U}}^{(t)}$ is

$$\tilde{\mathbf{F}}_{\mathcal{U}}^{(t)} = \mathbf{Q}_{\mathcal{S}^{(1:t)}} \tilde{\mathbf{P}}_{\mathcal{U},\cdot} \mathbf{F}^{(t-1)} + (\mathbf{I} - \mathbf{Q}_{\mathcal{S}^{(1:t)}}) \mathbf{F}_{\mathcal{U}}^{(t-1)}. \tag{4.25}$$

As a result, the difference between $\mathbf{F}_{\mathcal{U}}^{(t)}$ and $\tilde{\mathbf{F}}_{\mathcal{U}}^{(t)}$ is computed by

$$\begin{aligned} \left\|\tilde{\mathbf{F}}_{\mathcal{U}}^{(t)} - \mathbf{F}_{\mathcal{U}}^{(t)}\right\|_{\mathrm{F}} &= \left\|\mathbf{Q}_{\mathcal{S}^{(1:t)}} (\tilde{\mathbf{P}}_{\mathcal{U},\cdot} - \mathbf{P}_{\mathcal{U},\cdot}) \mathbf{F}^{(t-1)}\right\|_{\mathrm{F}} \\ &\leq \left\|\mathbf{Q}_{\mathcal{S}^{(1:t)}} (\tilde{\mathbf{P}}_{\mathcal{U},\cdot} - \mathbf{P}_{\mathcal{U},\cdot})\right\|_{\mathrm{F}} \left\|\mathbf{F}^{(t-1)}\right\|_{\mathrm{F}} \end{aligned} \tag{4.26}$$

By employing (4.23), we arrive at

$$\left\|\mathbf{Q}_{\mathcal{S}^{(1:t)}} (\tilde{\mathbf{P}}_{\mathcal{U},\cdot} - \mathbf{P}_{\mathcal{U},\cdot})\right\|_{\mathrm{F}} \leq (\delta^2 - 1) \sqrt{n \sum\nolimits_{i=1}^{t} s^{(i)}} \tag{4.27}$$

Additionally, since the sum of every row in $\mathbf{F}^{(t-1)} \in [0, 1]^{n \times c}$ is 1, we know that

$$\left\|\mathbf{F}^{(t-1)}\right\|_{\mathrm{F}} \leq \sqrt{n}. \tag{4.28}$$

Finally, by plugging (4.27) and (4.28) into (4.26) and noting that $\sum_{i=1}^{t} s^{(i)} \leq u^{(0)}$, we obtain

$$\left\|\tilde{\mathbf{F}}_{\mathcal{U}}^{(t)} - \mathbf{F}_{\mathcal{U}}^{(t)}\right\|_{\mathrm{F}} \leq (\delta^2 - 1) \sqrt{n \sum\nolimits_{i=1}^{t} s^{(i)}} \leq (\delta^2 - 1) n \sqrt{u^{(0)}}. \tag{4.29}$$

Since $n\sqrt{u^{(0)}}$ is a constant, Theorem 4.5 is proved, which reveals that our algorithm is insensitive to the perturbation of Gaussian kernel width ξ in one propagation. □

[1] For ease of explanation, we slightly abuse the notations in this section by using $\mathcal{L}$ and $\mathcal{U}$ to represent the initial labeled set $\mathcal{L}^{(0)}$ and unlabeled set $\mathcal{U}^{(0)}$. They are not time-varying variables as previously defined. Therefore, the notation $\mathbf{F}_{\mathcal{U}}^{(t)}$ represents the labels of initial unlabeled examples produced by the t-th propagation.

However, one may argue that the error introduced in every propagation will accumulate and degrade the final parametric stability of TLLT. To show that the error will not be significantly amplified, the error bound between successive propagations is presented in Theorem 4.6.

Theorem 4.6. *Let $\tilde{\mathbf{F}}_{\mathcal{U}}^{(t-1)}$ be the perturbed label matrix $\mathbf{F}_{\mathcal{U}}^{(t-1)}$ generated by the $(t-1)$-th propagation, which satisfies $\left\|\tilde{\mathbf{F}}_{\mathcal{U}}^{(t-1)} - \mathbf{F}_{\mathcal{U}}^{(t-1)}\right\|_{\mathrm{F}} \leq \mathcal{O}(\delta^2 - 1)$. Let $\tilde{\mathbf{P}}$ be the perturbed transition matrix $\mathbf{P}$. Then after the t-th propagation, the inaccurate output $\tilde{\mathbf{F}}_{\mathcal{U}}^{(t)}$ will deviate from its real value $\mathbf{F}_{\mathcal{U}}^{(t)}$ by $\left\|\tilde{\mathbf{F}}_{\mathcal{U}}^{(t)} - \mathbf{F}_{\mathcal{U}}^{(t)}\right\|_{\mathrm{F}} \leq \mathcal{O}(\delta^2 - 1)$.*

Proof. Theorem 4.6 can be proved based on the following existing result:

Lemma 4.7. ***Taylor and Thompson [1998]:*** *Suppose p_1 and p_2 are two uncertain variables with possible errors Δp_1 and Δp_2, then the deviation Δp_3 of their multiplication $p_3 = p_1 \cdot p_2$ satisfies $\Delta p_3 = p_1 \cdot \Delta p_2 + \Delta p_1 \cdot p_2$.*

Based on above lemma, next we bound the error accumulation between successive propagations under the perturbed Gaussian kernel width ξ. The Eq. (4.24) can be rearranged as:

$$\begin{aligned}
\mathbf{F}_{\mathcal{U}}^{(t)} &= \mathbf{Q}_{\mathcal{S}^{(1:t)}}\mathbf{P}_{\mathcal{U},\cdot}\mathbf{F}^{(t-1)} + (\mathbf{I} - \mathbf{Q}_{\mathcal{S}^{(1:t)}})\mathbf{F}_{\mathcal{U}}^{(t-1)} \\
&= \mathbf{Q}_{\mathcal{S}^{(1:t)}}(\mathbf{P}_{\mathcal{U},\mathcal{L}}\mathbf{F}_{\mathcal{L}}^{(t-1)} + \mathbf{P}_{\mathcal{U},\mathcal{U}}\mathbf{F}_{\mathcal{U}}^{(t-1)}) + (\mathbf{I} - \mathbf{Q}_{\mathcal{S}^{(1:t)}})\mathbf{F}_{\mathcal{U}}^{(t-1)} \\
&= \mathbf{Q}_{\mathcal{S}^{(1:t)}}\mathbf{P}_{\mathcal{U},\mathcal{L}}\mathbf{F}_{\mathcal{L}}^{(t-1)} + (\mathbf{Q}_{\mathcal{S}^{(1:t)}}\mathbf{P}_{\mathcal{U},\mathcal{U}} + \mathbf{I} - \mathbf{Q}_{\mathcal{S}^{(1:t)}})\mathbf{F}_{\mathcal{U}}^{(t-1)} \\
&= \begin{pmatrix}\mathbf{Q}_{\mathcal{S}^{(1:t)}}\mathbf{P}_{\mathcal{U},\mathcal{L}} & \mathbf{Q}_{\mathcal{S}^{(1:t)}}\mathbf{P}_{\mathcal{U},\mathcal{U}} + \mathbf{I} - \mathbf{Q}_{\mathcal{S}^{(1:t)}}\end{pmatrix}\begin{pmatrix}\mathbf{F}_{\mathcal{L}}^{(t-1)} \\ \mathbf{F}_{\mathcal{U}}^{(t-1)}\end{pmatrix} \\
&= \mathbf{\Phi}^{(t)}\mathbf{F}^{(t-1)},
\end{aligned} \tag{4.30}$$

where $\mathbf{\Phi}^{(t)} = \begin{pmatrix}\mathbf{Q}_{\mathcal{S}^{(1:t)}}\mathbf{P}_{\mathcal{U},\mathcal{L}} & \mathbf{Q}_{\mathcal{S}^{(1:t)}}\mathbf{P}_{\mathcal{U},\mathcal{U}} + \mathbf{I} - \mathbf{Q}_{\mathcal{S}^{(1:t)}}\end{pmatrix}$ and $\mathbf{F}^{(t-1)} = \begin{pmatrix}\mathbf{F}_{\mathcal{L}}^{(t-1)\top} & \mathbf{F}_{\mathcal{U}}^{(t-1)\top}\end{pmatrix}^{\top}$. Therefore, by leveraging Lemma 4.7, we know that

$$\tilde{\mathbf{F}}_{\mathcal{U}}^{(t)} - \mathbf{F}_{\mathcal{U}}^{(t)} = (\tilde{\mathbf{\Phi}}^{(t)} - \mathbf{\Phi}^{(t)})\mathbf{F}^{(t-1)} + \mathbf{\Phi}^{(t)}(\tilde{\mathbf{F}}^{(t-1)} - \mathbf{F}^{(t-1)}), \tag{4.31}$$

where $\tilde{\mathbf{\Phi}}^{(t)} = \begin{pmatrix}\mathbf{Q}_{\mathcal{S}^{(1:t)}}\tilde{\mathbf{P}}_{\mathcal{U},\mathcal{L}} & \mathbf{Q}_{\mathcal{S}^{(1:t)}}\tilde{\mathbf{P}}_{\mathcal{U},\mathcal{U}} + \mathbf{I} - \mathbf{Q}_{\mathcal{S}^{(1:t)}}\end{pmatrix}$ is the imprecise $\mathbf{\Phi}^{(t)}$ induced by $\tilde{\mathbf{P}}$. Consequently, we obtain

$$\begin{aligned}
\left\|\tilde{\mathbf{F}}_{\mathcal{U}}^{(t)} - \mathbf{F}_{\mathcal{U}}^{(t)}\right\|_{\mathrm{F}} &= \left\|(\tilde{\mathbf{\Phi}}^{(t)} - \mathbf{\Phi}^{(t)})\mathbf{F}^{(t-1)} + \mathbf{\Phi}^{(t)}(\tilde{\mathbf{F}}^{(t-1)} - \mathbf{F}^{(t-1)})\right\|_{\mathrm{F}} \\
&\leq \left\|\tilde{\mathbf{\Phi}}^{(t)} - \mathbf{\Phi}^{(t)}\right\|_{\mathrm{F}}\left\|\mathbf{F}^{(t-1)}\right\|_{\mathrm{F}} + \left\|\mathbf{\Phi}^{(t)}\right\|_{\mathrm{F}}\left\|\tilde{\mathbf{F}}^{(t-1)} - \mathbf{F}^{(t-1)}\right\|_{\mathrm{F}}.
\end{aligned} \tag{4.32}$$

Next we investigate the upper bounds of $\left\|\tilde{\mathbf{\Phi}}^{(t)}-\mathbf{\Phi}^{(t)}\right\|_{\mathrm{F}}$, $\left\|\mathbf{F}^{(t-1)}\right\|_{\mathrm{F}}$, $\left\|\mathbf{\Phi}^{(t)}\right\|_{\mathrm{F}}$, and $\left\|\tilde{\mathbf{F}}^{(t-1)}-\mathbf{F}^{(t-1)}\right\|_{\mathrm{F}}$. Of these, $\left\|\mathbf{F}^{(t-1)}\right\|_{\mathrm{F}}$ has been bounded in (4.28).

It is also straightforward that

$$\begin{aligned}
&\left\|\tilde{\mathbf{\Phi}}^{(t)}-\mathbf{\Phi}^{(t)}\right\|_{\mathrm{F}}\\
=&\left\|\begin{pmatrix}\mathbf{Q}_{\mathcal{S}^{(1:t)}}(\tilde{\mathbf{P}}_{\mathcal{U},\mathcal{L}}-\mathbf{P}_{\mathcal{U},\mathcal{L}}) & \mathbf{Q}_{\mathcal{S}^{(1:t)}}(\tilde{\mathbf{P}}_{\mathcal{U},\mathcal{U}}-\mathbf{P}_{\mathcal{U},\mathcal{U}})\end{pmatrix}\right\|_{\mathrm{F}}\\
=&\left\|\mathbf{Q}_{S^{(1:t)}}(\tilde{\mathbf{P}}_{\mathcal{U},\cdot}-\mathbf{P}_{\mathcal{U},\cdot})\right\|_{\mathrm{F}} \qquad ,\\
\overset{1}{\leq}&(\delta^2-1)\sqrt{n\sum\nolimits_{i=1}^{t}s^{(i)}}\\
\overset{2}{\leq}&(\delta^2-1)\sqrt{nu^{(0)}}
\end{aligned} \tag{4.33}$$

where the inequality 1 is given by (4.27), and the inequality 2 holds because $\sum_{i=1}^{t}s^{(i)}\leq u^{(0)}$.

By further investigating the structure of the $u^{(0)}\times n$ matrix $\mathbf{\Phi}^{(t)}$ in (4.30), it is easy to find that the i-th row of $\mathbf{\Phi}^{(t)}$ (*i.e.* $\mathbf{\Phi}^{(t)}_{i\cdot}$) is

$$\mathbf{\Phi}^{(t)}_{i\cdot}:=\begin{cases}\mathbf{P}_{i\cdot}, & \mathbf{x}_i\in\mathcal{S}^{(1:t)}\\ \left(0,\cdots,\underset{\substack{\downarrow\\ i-th\ element}}{1},\cdots,0\right), & \mathbf{x}_i\notin\mathcal{S}^{(1:t)}\end{cases}, \tag{4.34}$$

where $\mathbf{P}_{i\cdot}$ denotes the i-th row of $\mathbf{P}$. Therefore, the sum of every row in $\mathbf{\Phi}^{(t)}$ is not larger than 1, leading to

$$\left\|\mathbf{\Phi}^{(t)}\right\|_{\mathrm{F}}\leq\sqrt{u^{(0)}}, \tag{4.35}$$

where we again use the fact that $0\leq\mathbf{P}_{ij}\leq 1$.

Recalling that the labels of the original labeled examples are clamped after every propagation (please see Eq. (4.10)), the bound obtained in Theorem 4.5 also applies to $\left\|\tilde{\mathbf{F}}^{(t-1)}-\mathbf{F}^{(t-1)}\right\|_{\mathrm{F}}$, which is

$$\left\|\tilde{\mathbf{F}}^{(t-1)}-\mathbf{F}^{(t-1)}\right\|_{\mathrm{F}}=\left\|\tilde{\mathbf{F}}^{(t-1)}_{\mathcal{U}}-\mathbf{F}^{(t-1)}_{\mathcal{U}}\right\|_{\mathrm{F}}\leq\mathcal{O}(\delta^2-1). \tag{4.36}$$

Because $u^{(0)}$ and n are constants for a given problem, Theorem 4.6 is finally proved by substituting (4.28), (4.33), (4.35) and (4.36) into (4.32). Theorem 4.6 implies that under the perturbed $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{F}}^{(t-1)}$, the error bound after the t-th propagation $\left\|\tilde{\mathbf{F}}^{(t)}_{\mathcal{U}}-\mathbf{F}^{(t)}_{\mathcal{U}}\right\|_{\mathrm{F}}$ is the same as that before the t-th propagation $\left\|\tilde{\mathbf{F}}^{(t-1)}_{\mathcal{U}}-\mathbf{F}^{(t-1)}_{\mathcal{U}}\right\|_{\mathrm{F}}$. Therefore, the labelling error will not be significantly accumulated when the propagations proceed. □

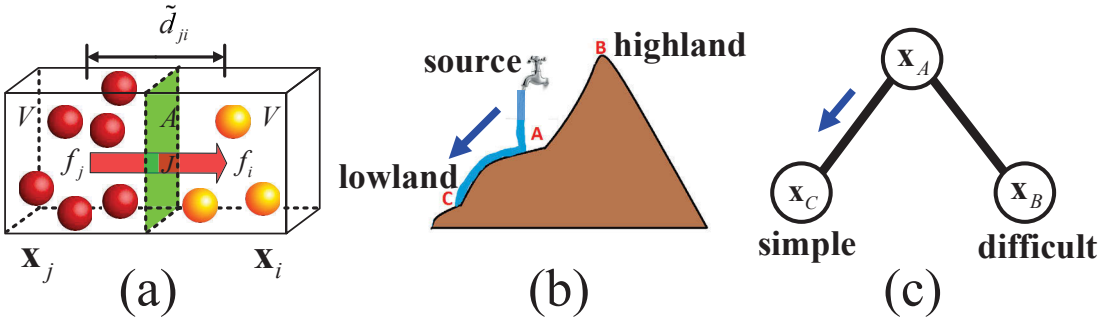

Figure 4.3: The physical interpretation of our TLLT label propagation algorithm. (a) compares the propagation between two examples with equal difficulty to the fluid diffusion between two cubes with same altitude. The left cube with more balls is compared to the examples with larger label value. The right cube with fewer balls is compared to the examples with less label information. The red arrow indicates the diffusion direction. (b) and (c) draw the parallel between fluid diffusion with different altitudes and label propagation guided by curriculums. The lowland "C", highland "B", and source "A" in (b) correspond to the simple vertex $\mathbf{x}_C$, difficult vertex $\mathbf{x}_B$, and labeled vertex $\mathbf{x}_A$ in (c), respectively. Like the fluid can only flow from "A" to the lowland "C" in (b), $\mathbf{x}_A$ in (c) also tends to transfer the label information to the simple vertex $\mathbf{x}_C$.

Taking into account Theorems 4.5 and 4.6 together, we conclude that a small variation of ξ will not greatly influence the performance of TLLT, so the robustness of the entire propagation algorithm is guaranteed. Accordingly, the parameter ξ used in our method can be easily tuned. An empirical demonstration of parametric insensitivity can be found in the Section 4.8.6.

4.7 Physical Interpretation

A key factor to the effectiveness of our TLLT method is the well-ordered learning sequence from simple to difficult, which is also considered by curriculum learning Bengio et al. [2009] and self-paced learning Kumar et al. [2010]. Recall that in Chapter 3, we draw an analogy between label propagation on the graph and the fluid diffusion on a plane, and apply the Fick's First Law of Diffusion to design a "natural" graph transduction algorithm. The diffusion between a pair of examples is illustrated in Fig. 4.3(a) for the ease of the following explanations.

Similar to Chapter 3, we also regard the labeled examples as sources to emit the fluid, and the remaining unlabeled examples are to be diffused. Differently, here the simple and difficult unlabeled examples are compared to lowlands and highlands, respectively (see Figs. 4.3(b)(c)). There are two obvious facts here: 1) the lowlands

will be propagated prior to the highlands, and 2) fluid cannot be transmitted from lowlands to highlands. Therefore, by treating γ of Eq. (3.2) as the propagation coefficient, f as the label, and $\widetilde{d}$ as the propagation distance defined by $\widetilde{d}_{ji} = 1/\sqrt{\omega_{ji}}$, Eq. (3.2) explains the process of label propagation from $\mathbf{x}_j$ to $\mathbf{x}_i$ as

$$J_{ji} = -m_i\gamma\frac{f_i^{(t)} - f_j^{(t)}}{\widetilde{d}_{ji}} = -m_i\gamma\sqrt{\omega_{ji}}(f_i^{(t)} - f_j^{(t)}). \tag{4.37}$$

The parameter m_i in (4.37) denotes the "altitude" of $\mathbf{x}_i$, which is different from Eq. (3.2) in Chapter 3. It equals to 1 if $\mathbf{x}_i$ corresponds to a lowland, and 0 if $\mathbf{x}_i$ represents a highland. Note that if $\mathbf{x}_i$ is higher than $\mathbf{x}_j$, the flux $J_{ji} = 0$ because the fluid cannot transfer from lowland to highland. Given (4.37), we have the following theorem:

Theorem 4.8. *Suppose all the examples $\mathbf{x}_1, \cdots, \mathbf{x}_n$ are modeled as the cubes with volume V, and the area of their interface is A (see Fig. 4.3(a)). By using m_i to indicate the altitude of $\mathbf{x}_i$ and setting the propagation coefficient $\gamma = 1$, the proposed propagation process can be derived from the fluid transmission modeled by Fick's Law of Diffusion.*

Proof. The propagation process from example $\mathbf{x}_j$ to $\mathbf{x}_i$ is illustrated in Fig. 4.3(a). Since both examples are regarded as cubes with volume V, and the area of their interface is A, so during an unit time from t to $t+1$, the label information (similar to the number of molecules in fluids) received by $\mathbf{x}_i$ satisfies Eq. (3.3). By replacing J_{ji} with the Eq. (4.37) and considering $V = A/\sqrt{\omega_{ji}}$, we have the basic propagation model between two examples expressed as

$$f_i^{(t+1)} - f_i^{(t)} = -\gamma m_i\omega_{ji}(f_i^{(t)} - f_j^{(t)}). \tag{4.38}$$

Practically, an example receives the labels from all its neighbors rather than only one as modelled by (4.38), so the label information propagated to $\mathbf{x}_i$ should be summed over multiple examples. Therefore, by treating $\omega_{ji} = 0$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are not directly linked by an edge on $\mathcal{G}$, (4.38) is extended to

$$f_i^{(t+1)} - f_i^{(t)} = -\gamma m_i \sum\nolimits_{j=1\sim n, j\neq i} \omega_{ji}(f_i^{(t)} - f_j^{(t)}), \tag{4.39}$$

where n is the total amount of examples on the graph. After re-arranging (4.39), we obtain the following model explaining the diffusions among multiple examples:

$$f_i^{(t+1)} = \left(1 - \gamma m_i \sum_{j=1\sim n,\ j\neq i} \omega_{ji}\right) f_i^{(t)} + \gamma m_i \sum_{j=1\sim n,\ j\neq i} \omega_{ji} f_j^{(t)}. \tag{4.40}$$

By applying (4.40) to all the n examples $\{\mathbf{x}_i\}_{i=1}^n$, the propagation on graph $\mathcal{G}$ can be reformulated into a compact formation

$$\mathbf{f}^{(t+1)} = \mathbf{\Psi}\mathbf{f}^{(t)}, \tag{4.41}$$

where $\mathbf{f}^{(t)} = (f_1^{(t)}, f_2^{(t)}, \cdots, f_n^{(t)})^\top$, and

$$\mathbf{\Psi} = \begin{pmatrix} 1-\gamma m_1 \sum\limits_{j=1\sim n,\ j\neq 1} \omega_{j1} & \gamma m_1\omega_{21} & \cdots & \gamma m_1\omega_{n1} \\ \gamma m_2\omega_{12} & 1-\gamma m_2 \sum\limits_{j=1\sim n,\ j\neq 2} \omega_{j2} & \cdots & \gamma m_2\omega_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma m_n\omega_{1n} & \gamma m_n\omega_{2n} & \cdots & 1-\gamma m_n \sum\limits_{j=1\sim n,\ j\neq n} \omega_{jn} \end{pmatrix}.$$

For propagation purpose, the diagonal elements in $\mathbf{\Psi}$ are set to 0 to avoid the self-loop on graph $\mathcal{G}$ Zhou and Bousquet [2003]. Therefore, Eq. (4.41) can be rewritten as

$$\mathbf{f}^{(t+1)} = \gamma\mathbf{M}^{(t)}\mathbf{W}\mathbf{f}^{(t)}. \tag{4.42}$$

By row-normalizing $\mathbf{W}$ as $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ and setting the propagation coefficient $\gamma = 1$, we achieve the employed propagation model that is essentially the same as Eq. (4.10). Consequently, our propagation strategy from simple examples to more difficult ones has a close relationship with the practical fluid diffusion with highlands and lowlands. This completes the proof. □

4.8 Experimental Results

In this section, we compare the proposed TLLT with several representative label propagation methods on both synthetic and practical datasets. In particular, we implement TLLT with two different learning-to-teach strategies presented by Eqs. (4.12) and (4.13), and term them "TLLT (Norm)" and "TLLT (Entropy)", respectively. The compared methods include Gaussian Field and Harmonic Functions (GFHF) Zhu and Ghahramani [2002], Local and Global Consistency (LGC) Zhou and Bousquet [2003], Graph Transduction via Alternating Minimization (GTAM) Wang et al. [2008b], Linear Neighbourhood Propagation Wang et al. [2009b] (LNP), and Dynamic Label Propagation (DLP) Wang et al. [2013]. Note that GFHF is the learning model (*i.e.*, learner) used by our proposed TLLT, which is not instructed by a teacher.

4.8.1 Synthetic Data

We begin by leveraging the two-dimensional *DoubleMoon* dataset to visualize the propagation process of different methods. *DoubleMoon* consists of 640 examples,

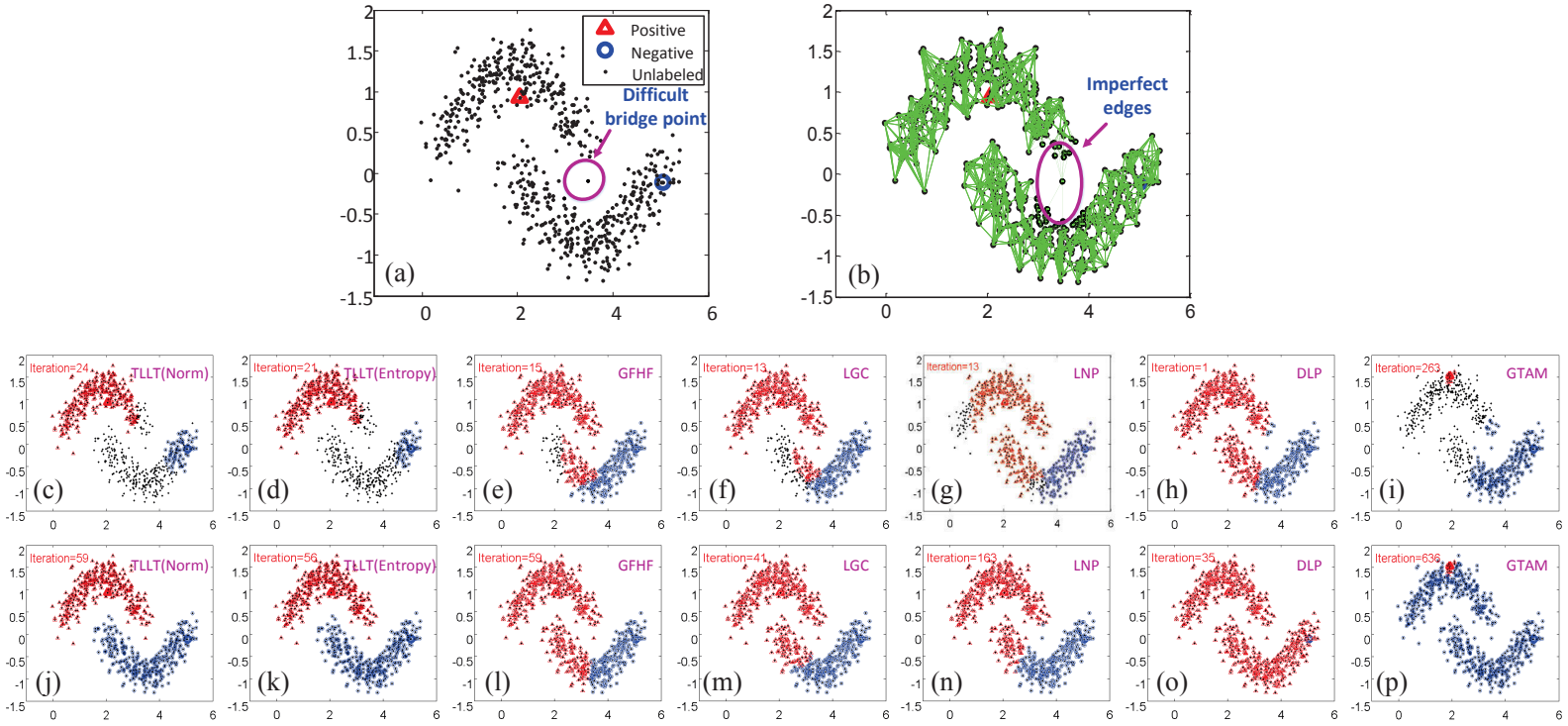

Figure 4.4: The propagation process of the methods on the *DoubleMoon* dataset. (a) is the initial state with marked labeled examples and difficult bridge point. (b) shows the imperfect edges during graph construction caused by the bridge point in (a). These unsuitable edges pose a difficulty for all the compared methods to achieve accurate propagation. The second row (c)∼(i) shows the intermediate propagations of TLLT (Norm), TLLT (Entropy), GFHF, LGC, LNP, DLP, and GTAM. The third row (j)∼(p) compares the results achieved by all the algorithms, which reveals that only the proposed TLLT achieves perfect classification while the other methods are misled by the ambiguous bridge point.

which are equally divided into two moons. This dataset was contaminated by Gaussian noise with standard variance 0.15, and each class had only one initial labeled example (see Fig. 4.4(a)). The 8-NN graph with the Gaussian kernel width $\xi = 1$ is established for TLLT, GFHF, LGC, GTAM, and DLP. The parameter μ in GTAM is set to 99 according to Wang et al. [2008b]. The number of neighbors k for LNP is adjusted to 10. We set $\gamma_1 = 1$ for TLLT (Norm) and $\gamma_2 = 2$ for TLLT (Entropy). The trade-off parameter α in Eq. (4.7) is fixed to 1 throughout the chapter, and we show that the result is insensitive to the variation of this parameter in Section 4.8.6.

From Fig. 4.4(a), we observe that the distance between the two classes is very small, and that a difficult bridge point is distributed over the intersection region between the two moons. Therefore, the improper edges (see Fig. 4.4(b)) caused by the bridge point may lead to the mutual transmission of labels from different classes. As a result, previous label propagation methods like GFHF, LGC, LNP, DLP and GTAM generate unsatisfactory results, as shown by Figs. 4.4(l)(m)(n)(o)(p). In contrast, only our proposed TLLT (including TLLT (Norm) and TLLT (Entropy)) achieves perfect classification without any confusion (see Figs. 4.4(j)(k)). The reason for our accurate propagation can be found in Figs. 4.4(c)(d), which indicate that the propagation to ambiguous bridge point is postponed due to the careful curriculum selection. On the contrary, this critical but difficult example is propagated by GFHF, LGC, LNP, DLP and GTAM at an early stage as long as it is connected to the initial labeled examples (see Figs. 4.4(e)(f)(g)(h)(i)), resulting in the mutual label transmission between the two moons. This experiment highlights the importance of our teaching-guided label propagation.

4.8.2 UCI Benchmark Data

In this section, we compare TLLT with GFHF, LGC, GTAM, LNP, and DLP on four UCI benchmark datasets Frank and Asuncion [2010]. Of these four datasets, *SPECTF* concerns about the binary classification while *Iris*, *Wine* and *Seeds* are multi-class classification problems. For each dataset, all the algorithms are tested with different numbers of initial labeled examples, such as $l^{(0)} = 40$ and $l^{(0)} = 80$ for *SPECTF*, and $l^{(0)} = 60$ and $l^{(0)} = 120$ for *Iris*, *Wine* and *Seeds*. In order to suppress the influence of different initial labeled sets to the final performance, the accuracies are reported as the mean values of the outputs of ten independent runs. In each run, the labeled examples are randomly selected. However, the ten different partitions of labeled and unlabeled sets are identical for all the compared algorithms.

On each dataset, we established the same 5-NN graph (*i.e.* $k = 5$) for GFHF, LGC, GTAM and DLP to achieve fair comparison. For LNP, we respectively set k to 10, 50, 30 and 50 on *Iris*, *Wine*, *Seeds* and *SPECTF* since the graph required by LNP is different from other methods. The learning rates γ_1 and γ_2 in TLLT were simply tuned to 1, and the classification accuracies of all the compared methods are presented by

Table 4.1: Experimental results of the compared methods on four UCI benchmark datasets. (Each record in the table represents the "accuracy ± standard deviation". The highest result obtained on each dataset is marked in bold.)

Datasets	*Iris*		*Wine*		*Seeds*		*SPECTF*	
$l^{(0)}$	60	120	60	120	60	120	40	80
GFHF	0.962±0.011	0.988±0.004	0.962±0.009	0.979±0.009	0.918±0.011	0.949±0.005	0.667±0.029	0.739±0.013
LGC	0.940±0.017	0.957±0.013	0.898±0.014	0.908±0.022	0.906±0.007	0.907±0.008	0.515±0.088	0.548±0.052
GTAM	0.867±0.026	0.969±0.015	0.897±0.035	0.924±0.054	0.828±0.041	0.914±0.036	0.482±0.002	0.556±0.003
LNP	0.832±0.054	0.905±0.059	0.749±0.133	0.780±0.130	0.702±0.080	0.865±0.055	0.434±0.061	0.525±0.050
DLP	0.951±0.008	0.980±0.005	0.875±0.029	0.927±0.026	0.919±0.006	0.942±0.009	0.512±0.131	0.611±0.035
TLLT (Norm)	**0.979±0.003**	0.991±0.004	**0.965±0.011**	0.990±0.007	**0.927±0.009**	0.952±0.006	**0.672±0.051**	**0.751±0.024**
TLLT (Entropy)	0.975±0.010	**0.996±0.006**	**0.965±0.007**	**0.991±0.009**	0.919±0.014	**0.956±0.010**	0.667±0.028	0.742±0.030

Table 4.1.

From Table 4.1, we observe that the proposed TLLT (Norm) and TLLT (Entropy) yield better performance than other baselines on all the adopted datasets. Another fact is that both TLLT (Norm) and TLLT (Entropy) consistently outperform GFHF, which is the plain learning model adopted by TLLT. Therefore, the well-organized learning sequence produced by our teaching and learning strategy does help to improve the propagation performance. Notably, TLLT (Entropy) achieves more than 99% accuracy on *Iris* and *Wine* datasets when $l^{(0)} = 120$.

4.8.3 Text Categorization

To demonstrate the superiority of TLLT in dealing with practical problems, we compare TLLT against GFHF, LGC, GTAM, LNP, and DLP in terms of text categorization. A subset of *RCV1* Lewis et al. [2004] with 2000 documents is employed for our experiment. These documents were classified into four categories: "C15", "ECAT", "GCAT", and "MCAT", and each category contained 500 examples. The standard TF-IDF weighting scheme was adopted to generate the feature vector for each document. The parameters of the graph used for GFHF, LGC, GTAM, DLP and TLLT are fixed to $k = 10$ and $\xi = 2$. The k for LNP is adjusted to 20. The learning rates for TLLT (Norm) and TLLT (Entropy) are optimally tuned to $\gamma_1 = 0.2$ and $\gamma_2 = 2$ via grid search hereinafter. Specifically, γ_1 and γ_2 are chosen from 0.1 to 2 with the interval 0.05. We implement all the methods with $l^{(0)} = 200, 400, 600, 800$ initial labeled examples, and report the classification accuracy averaged over the outputs of ten independent runs. In each run, the labeled and unlabeled examples are randomly generated; however, the ten different partitions of labeled and unlabeled sets are identical for all the compared methods. The experimental results are summarized in Table 4.2, in which the highest records are marked with boldface. We observe that both TLLT (Norm) and TLLT (Entropy) outperform the other competing methods when $l^{(0)}$ varies from 200 to 800.

Table 4.2: Accuracy of all methods on the *RCV1* dataset (the highest records are marked in bold).

	$l^{(0)} = 200$	$l^{(0)} = 400$	$l^{(0)} = 600$	$l^{(0)} = 800$
GFHF	0.848 ± 0.017	0.872 ± 0.007	0.905 ± 0.008	0.930 ± 0.004
LGC	0.832 ± 0.018	0.862 ± 0.013	0.879 ± 0.010	0.889 ± 0.008
GTAM	0.500 ± 0.002	0.501 ± 0.002	0.576 ± 0.004	0.651 ± 0.001
LNP	0.515 ± 0.079	0.582 ± 0.089	0.575 ± 0.047	0.599 ± 0.055
DLP	0.752 ± 0.044	0.858 ± 0.024	0.900 ± 0.008	0.921 ± 0.004
TLLT (Norm)	$\mathbf{0.860 \pm 0.012}$	0.879 ± 0.006	0.917 ± 0.008	0.923 ± 0.006
TLLT (Entropy)	0.856 ± 0.008	$\mathbf{0.887 \pm 0.003}$	$\mathbf{0.920 \pm 0.003}$	$\mathbf{0.933 \pm 0.007}$

Figure 4.5: Example images of *COIL20* dataset.

4.8.4 Object Recognition

We also apply the proposed TLLT to typical computer vision problems. *COIL20* is a popular public dataset for object recognition which contains 1440 object images belonging to 20 classes (see Fig. 4.5 for examples), and each object has 72 images shot from different angles. The resolution of each image is 32×32, with 256 grey levels per pixel. Thus, each image is represented by a 1024-dimensional element-wise vector. We built a 5-NN graph with $\xi = 50$ for GFHF, LGC, GTAM, DLP and TLLT. The number of neighbours k for LNP was tuned to 10. Other parameters were $\gamma_1 = 1$ for TLLT (Norm) and $\gamma_2 = 0.5$ for TLLT (Entropy). All the algorithms were implemented under $l^{(0)} = 100, 200, 300, 400$ initial labeled examples, and the reported accuracies are mean values of the outputs of ten independent runs. Table 4.3 shows

Table 4.3: Accuracy of all methods on the *COIL20* dataset (the highest records are marked in bold).

	$l^{(0)} = 100$	$l^{(0)} = 200$	$l^{(0)} = 300$	$l^{(0)} = 400$
GFHF	0.886 ± 0.009	0.899 ± 0.006	0.935 ± 0.003	0.942 ± 0.007
LGC	0.884 ± 0.013	0.895 ± 0.009	0.898 ± 0.004	0.902 ± 0.005
GTAM	0.716 ± 0.026	0.781 ± 0.019	0.841 ± 0.029	0.889 ± 0.025
LNP	0.872 ± 0.011	0.873 ± 0.011	0.870 ± 0.005	0.867 ± 0.004
DLP	0.778 ± 0.019	0.838 ± 0.022	0.870 ± 0.003	0.881 ± 0.005
TLLT (Norm)	$\mathbf{0.897 \pm 0.007}$	$\mathbf{0.922 \pm 0.007}$	$\mathbf{0.950 \pm 0.006}$	$\mathbf{0.957 \pm 0.005}$
TLLT (Entropy)	$\mathbf{0.897 \pm 0.008}$	0.920 ± 0.002	0.943 ± 0.009	0.955 ± 0.005

Figure 4.6: Example images of *UT Zappos* dataset.

the comparison results, from which we observe that TLLT consistently hits the highest records among all comparators. Though the number of initial labeled examples is very limited, our method can still achieve impressive output.

4.8.5 Fine-grained Image Classification

To further demonstrate the strength of TLLT, we apply the proposed algorithm to identify the fine-grained visual differences. We use a subset of the *UT Zappos* dataset Yu and Grauman [2014] to classify 4000 shoes belonging to 4 common categories: "shoe", "sandal", "slipper", and "boot". The orientation of the shoes is consistent and the shoes are centred on a white background image as shown by Fig. 4.6. It can be observed that the colour and style of shoes within a category differ significantly, which makes accurate classification extremely challenging.

Each example is represented by a 960-dimensional GIST feature. The established

Table 4.4: Accuracy of all methods on *UT Zappos* dataset (highest records are marked in bold).

	$l^{(0)} = 160$	$l^{(0)} = 240$	$l^{(0)} = 320$	$l^{(0)} = 400$
GFHF	0.771 ± 0.014	0.803 ± 0.012	0.804 ± 0.002	0.818 ± 0.009
LGC	0.773 ± 0.021	0.490 ± 0.007	0.492 ± 0.013	0.810 ± 0.016
GTAM	0.680 ± 0.001	0.695 ± 0.002	0.710 ± 0.001	0.725 ± 0.001
LNP	0.654 ± 0.042	0.664 ± 0.084	0.725 ± 0.037	0.745 ± 0.044
DLP	0.505 ± 0.183	0.719 ± 0.072	0.756 ± 0.033	0.804 ± 0.162
TLLT (Norm)	$\mathbf{0.793 \pm 0.011}$	0.809 ± 0.011	0.823 ± 0.009	0.840 ± 0.007
TLLT (Entropy)	0.786 ± 0.012	$\mathbf{0.817 \pm 0.012}$	$\mathbf{0.826 \pm 0.009}$	$\mathbf{0.845 \pm 0.007}$

graph for GFHF, LGC, GTAM, DLP and TLLT was parameterized by $k = 5$ and $\xi = 1$, and the k for LNP was set to 10. We select the value of k from $\{5, 6, \cdots, 15\}$ so that the highest accuracy is obtained. The learning rates of TLLT (Norm) and TLLT (Entropy) were $\gamma_1 = 0.15$ and $\gamma_2 = 2$, respectively.

The accuracy achieved by all the methods under $l^{(0)} = 160, 240, 320, 400$ initial labeled examples is reported in Table 4.4. Similar to above experiments, each reported accuracy is averaged over ten implementations with different labeled sets. Table 4.4 suggests that TLLT (Norm) and TLLT (Entropy) outperform other comparators with the increase of $l^{(0)}$. We also observe that both TLLT (Entropy) and TLLT (Norm) are superior to GFHF, therefore the strength of our interactive teaching and learning is demonstrated. Other baselines like LGC, GTAM, LNP and DLP also perform worse than TLLT.

4.8.6 Parametric Sensitivity

In Section 4.6, we theoretically verify that TLLT is insensitive to the change of Gaussian kernel width ξ. Besides, the weighing parameter α in Eq. (4.7) is also a key parameter to be tuned in our method. In this section, we investigate the parametric sensitivity of each of the parameters ξ and α by examining classification performance of one while the other is fixed. The above three practical datasets *RCV1*, *COIL20*, and *UT Zappos* are adopted here, and the results are illustrated in Fig. 4.7.

Fig. 4.7 reveals that TLLT is very robust to the variations of these two parameters, so they can be easily tuned for practical use. The results in Figs. 4.7 (a), (c) and (e) are also consistent with the theoretical validation in Section 4.6.

4.9 Summary of This Chapter

This chapter proposed a novel label propagation algorithm through iteratively employing a Teaching-to-Learn and Learning-to-Teach (TLLT) scheme. The main ingredients contributing to the success of TLLT are 1) explicitly manipulating the propagation sequence to move from the simple to difficult examples, and 2) adaptively determining the feedback-driven curriculums. These two contributions collaboratively lead to higher classification accuracy than the existing algorithms, and exhibit the robustness to the choice of graph parameters. Empirical studies reveal that TLLT can accomplish the state-of-the-art performance in various applications.

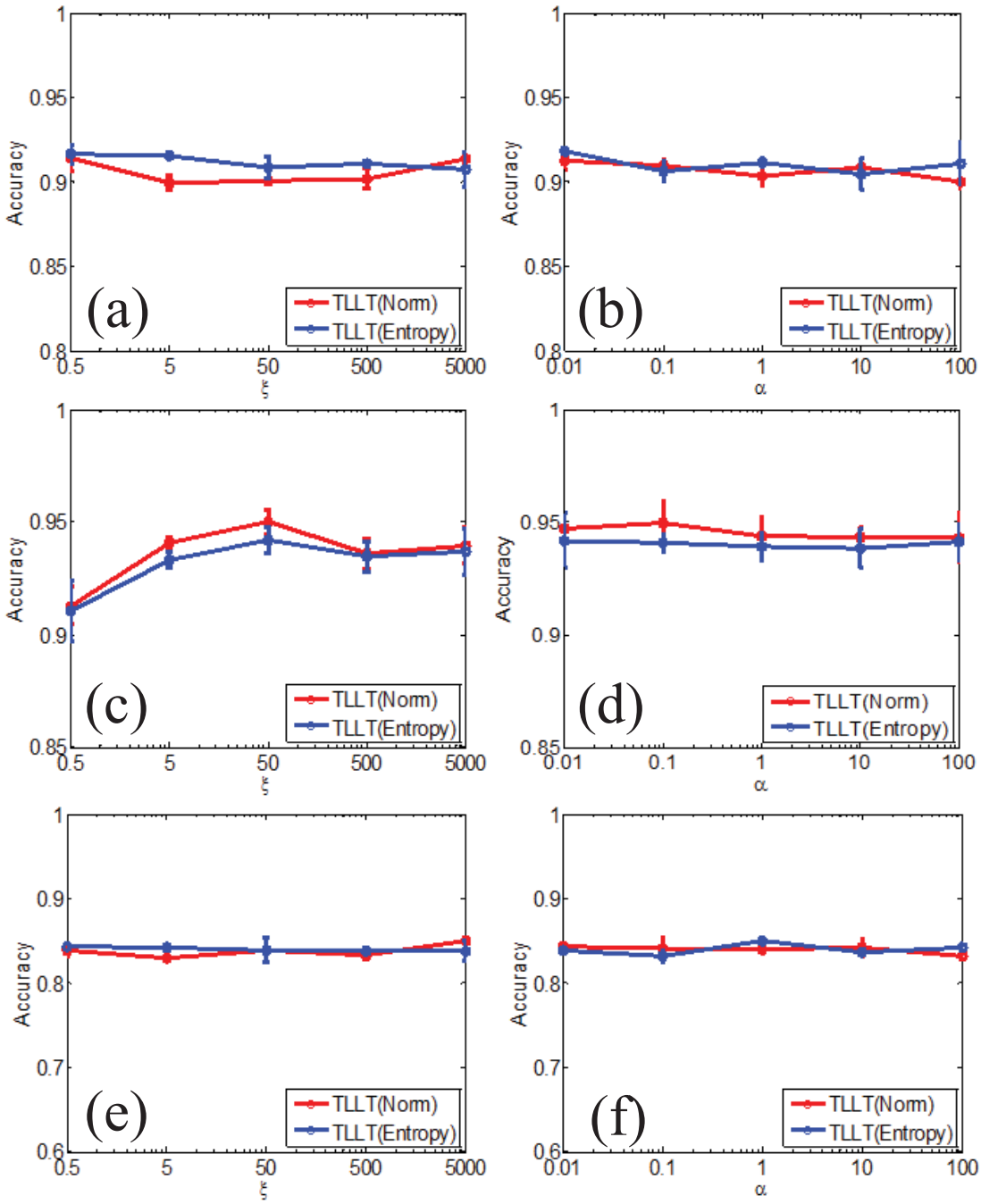

Figure 4.7: Parametric sensitivity of TLLT. The first, second and third rows correspond to *RCV1*, *COIL20* and *UT Zappos* datasets, respectively. (a), (c) and (e) show the variation of accuracy w.r.t. the kernel width ξ when α is fixed to 1, and (b), (d) and (f) evaluate the influence of the trade-off α to final accuracy under $\xi = 10$.

Chapter 5

Teaching-to-Learn and Learning-to-Teach For Saliency Detection

Saliency detection has been widely adopted for identifying the most attractive object in an image. Recently, label propagation has been introduced for saliency detection and achieved encouraging performance. The propagation sequence generated by existing saliency detection methods is governed by the spatial relationships of image regions, *i.e.*, the saliency value is transmitted between two adjacent regions. However, for the inhomogeneous difficult adjacent regions, such a sequence may incur wrong propagations. In this chapter, we apply the framework of teaching-to-learn and learning-to-teach proposed in Chapter 4, and attempt to manipulate the propagation sequence for optimizing the propagation quality. Intuitively, we postpone the propagations to difficult regions and meanwhile advance the propagations to less ambiguous simple regions. In the teaching-to-learn step, a teacher is designed to arrange the regions from simple to difficult and then assign the simplest regions to the learner. In the learning-to-teach step, the learner delivers its learning confidence to the teacher to assist the teacher to choose the subsequent simple regions. Due to the interactions between the teacher and learner, the uncertainty of original difficult regions is gradually reduced, yielding manifest salient objects with optimized background suppression. Extensive experimental results on benchmark saliency datasets demonstrate the superiority of the proposed algorithm over twelve representative saliency detectors.

5.1 A Brief Introduction to Saliency Detection

Saliency detection has attracted intensive attention and achieved considerable progress during the past two decades. Up to now, a great number of detectors based on

Figure 5.1: The results achieved by typical propagation methods and our method on two example images. From left to right: input images, results of Yang et al. [2013b], Jiang et al. [2013], and our method.

computational intelligence have been proposed. They can be roughly divided into two categories: *bottom-up* methods that are data and stimulus driven, and *top-down* methods that are task and knowledge driven.

Top-down methods are usually related to the subsequent applications. For example, Maybank [2013] proposed a probabilistic definition of salient image regions for image matching. Yang and Yang [2012] combined dictionary learning and Conditional Random Fields (CRFs) to generate discriminative representation of target-specific objects. Moreover, the work in Gao et al. [2009] and Zhu et al. [2014a] also belong to typical top-down methods.

Different from top-down methods, bottom-up methods use low-level cues, such as contrast and spectral information, to recognize the most salient regions without realizing content or specific prior knowledge about the targets. The representatives include Cheng et al. [2011]; Fu et al. [2014]; Hou and Zhang [2007]; Itti et al. [1998]; Jiang et al. [2011]; Kim et al. [2014]; Lee et al. [2005]; Li et al. [2014]; Margolin et al. [2013]; Perazzi et al. [2012].

Recently, propagation methods have gained much popularity in bottom-up saliency detection and achieved state-of-the-art performance. To conduct saliency propagations, an input image is represented by a graph over the segmented superpixels, in which the adjacent superpixels in the image are connected by weighted edges. The saliency values are then iteratively diffused along these edges from the labeled superpixels to their unlabeled neighbors. However, such propagations may incur errors if the unlabeled adjacent superpixels are inhomogeneous or very dissimilar to the labeled ones. For example, Gopalakrishnan et al. [2009] and Jiang et al. [2013] formulate the

saliency propagation process as random walks on the graph. Ren et al. [2010] and Yang et al. [2013b] conduct the propagations by employing personalized PageRank Zhou et al. [2004b] and manifold based diffusion Zhou et al. [2004b], respectively. All these methods generate similar propagation sequences which are heavily influenced by the superpixels' spatial relationships. However, once encountering the inhomogeneous or incoherent adjacent superpixels, the propagation sequences are misleading and likely to lead to inaccurate detection results (see Fig. 5.1).

Based on the above observations, we argue that not all neighbors are suitable to participate in the propagation process, especially when they are inhomogeneous or visually different from the labeled superpixels. Therefore, we assume different superpixels have different difficulties, and measure the saliency values of the simple superpixels prior to the difficult ones. This modification to the traditional scheme of generating propagation sequences is very critical, because in this modification the previously attained knowledge can ease the learning burden associated with complex superpixels afterwards, so that the difficult regions can be precisely discovered. Such a "starting simple" strategy conforms to the widely acknowledged theoretical results in pedagogic and cognitive areas Elman [1993]; Khan et al. [2011]; Rohde and Plaut [1999], which emphasize the importance of teachers for human's acquisitions of knowledge from the childish stage to the mature stage.

By taking advantage of these psychological opinions, we propose a novel approach for saliency propagation by leveraging a teaching-to-learn and learning-to-teach paradigm. This paradigm plays two key roles: a teacher behaving as a superpixel selection procedure, and a learner working as a saliency propagation procedure. In the teaching-to-learn step of the t-th propagation, the teacher assigns the simplest superpixels (*i.e.*, curriculum) to the learner in order to avoid the erroneous propagations to the difficult regions. The *informativity*, *individuality*, *inhomogeneity*, and *connectivity* of the candidate superpixels are comprehensively evaluated by the teacher to decide the proper curriculum. In the learning-to-teach step, the learner reports its t-th performance to the teacher in order to assist the teacher in wisely deciding the $(t+1)$-th curriculum. If the performance is satisfactory, the teacher will choose more superpixels into the curriculum for the following learning process. Owing to the interactions between the teacher and learner, the superpixels are logically propagated from simple to difficult with the updated curriculum, resulting in more confident and accurate saliency maps than those of typical methods (see Fig. 5.1).

5.2 Saliency Detection Algorithm

This section details our saliency detection scheme (see Fig. 5.2). When an input image is given, it is pre-processed by computing the convex hull, segmenting into superpixels, and constructing the graph over these superpixels. After that, the saliency values

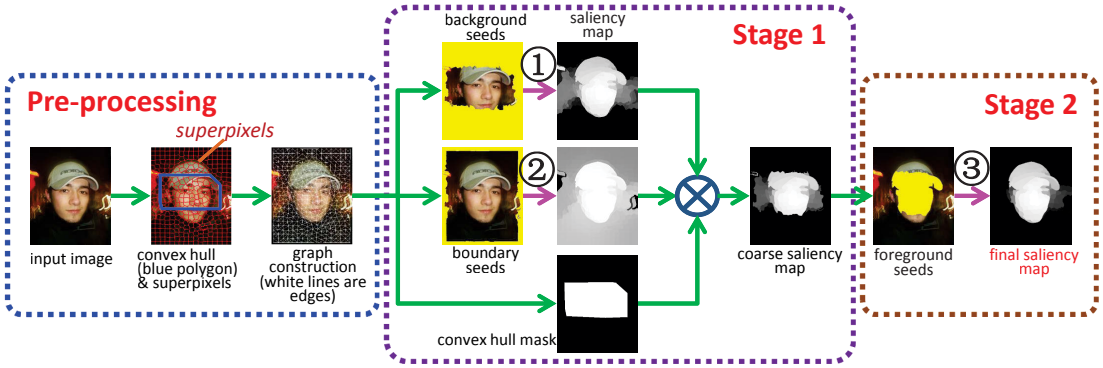

Figure 5.2: The diagram of our detection algorithm. The magenta arrows annotated with numbers denote the implementations of teaching-to-learn and learning-to-teach propagation shown in Fig. 5.3.

are propagated from the background seeds to form a coarse map (Stage 1). Finally, this map is refined by propagating the saliency information from the most confident foreground regions to the remaining superpixels (Stage 2). In the above stages, all the propagations are implemented under the proposed teaching-to-learn and learning-to-teach paradigm (see the magenta arrows in Fig. 5.2), which will be concretely introduced in Section 5.3.

5.2.1 Image Pre-processing

Given an input image, a convex hull $\mathcal{H}$ is constructed to estimate the target's location Yang et al. [2013a]. This is done by detecting some key points in the image via Harris corner detector. Because most key points locate within the target region, we link the outer key points to a convex hull to roughly enclose the target (see Fig. 5.2).

We proceed by using the SLIC Achanta et al. [2012] algorithm to over-segment the input image into N small superpixels (see Fig. 5.2), then an undirected graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is built where $\mathcal{V}$ is the vertex set consisted of these superpixels and $\mathcal{E}$ is the edge set encoding the similarity between them. In our work, we link two vertices[1] $\mathbf{s}_i$ and $\mathbf{s}_j$ by an edge if they are spatially adjacent in the image or both of them correspond to the boundary superpixels. Then their similarity is computed by the Gaussian kernel function $\omega_{ij} = \exp\left(-\|\mathbf{s}_i - \mathbf{s}_j\|^2/(2\sigma^2)\right)$, where σ is the kernel width and $\mathbf{s}_i$ is the feature vector of the i-th superpixel represented in the LAB-XY space (*i.e.* $\mathbf{s}_i = (\mathbf{s}_i^{color}; \mathbf{s}_i^{position})$). Therefore, the $\mathcal{G}$'s associated adjacency matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ is defined by $\mathbf{W}_{ij} = \omega_{ij}$ if $i \neq j$, and $\mathbf{W}_{ij} = 0$ otherwise. The diagonal degree matrix is $\mathbf{D}$ with $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$.

[1]In this chapter, "superpixel" and "vertex" refer to the same thing. We use them interchangeably for different explanation purposes.

5.2.2 Coarse Map Establishment

A coarse saliency map is built from the perspective of background, to assess how these superpixels are distinct from the background. To this end, some regions that are probably background should be determined as seeds for the saliency propagation. Two background priors are adopted to initialize the background propagations. The first one is the *convex hull* prior Yang et al. [2013a] that assumes the pixels outside the convex hull are very likely to be the background; and the second one is the *boundary prior* Wei et al. [2012]; Yang et al. [2013b] which indicates the regions along the image's four boundaries are usually non-salient. Note that the two priors are not strong because the specified non-salient regions will be further refined or modified by the subsequent propagation process, and every superpixel will receive a reasonable saliency value as a result.

For employing the convex hull prior, the superpixels outside $\mathcal{H}$ are regarded as background seeds (marked with yellow in Fig. 5.2) for saliency propagation. Suppose the propagation result is expressed by an N-dimensional vector $\mathbf{f}^* = \begin{pmatrix} f_1^* & \cdots & f_N^* \end{pmatrix}^\top$, where f_i^* $(i = 1, \cdots, N)$ are obtained saliency values corresponding to the superpixels $\mathbf{s}_i$, then after scaling $\mathbf{f}^*$ to $[0, 1]$ (denoted as $\mathbf{f}^*_{normalized}$), the value of the i-th superpixel in the saliency map $S_{ConvexHull}$ is

$$S_{ConvexHull}(i) = 1 - \mathbf{f}^*_{normalized}(i), i = 1, 2, \cdots, N, \tag{5.1}$$

Similarly, we treat the superpixels of four boundaries as seeds, and implement the propagation again. A saliency map based on the boundary prior can then be generated, which is denoted as $S_{Boundary}$. Furthermore, we establish a binary mask S_{mask} Fu et al. [2013] to indicate whether the i-th superpixel is inside ($S_{Mask}(i) = 1$) or outside ($S_{Mask}(i) = 0$) the convex hull $\mathcal{H}$. Finally, the saliency map of Stage 1 is obtained by integrating $S_{ConvexHull}$, $S_{Boundary}$, and S_{Mask} as

$$S_{Stage1} = S_{ConvexHull} \otimes S_{Boundary} \otimes S_{Mask}, \tag{5.2}$$

where "$\otimes$" is the element-wise product between matrices.

5.2.3 Map Refinement

After the Stage 1, the dominant object can be roughly highlighted. However, S_{Stage1} may still contain some background noise that should be suppressed. Therefore, we need to propagate the saliency information from the potential foreground regions to further improve S_{Stage1}.

Intuitively, we may choose the superpixels with large saliency values in S_{Stage1} as foreground seeds. In order to avoid erroneously taking background as seeds, we carefully pick up a small number of superpixels as seeds that are in the set:

$$\left\{\mathbf{s}_i |\ S_{Stage1}(i) \geq \eta \max_{1 \leq j \leq N}(S_{Stage1}(j))\right\}, \tag{5.3}$$

where η is set to 0.7. Finally, by setting the labels of seeds to 1 and conducting the teaching-to-learn and learning-to-teach propagation, we achieve the final saliency map S_{Stage2}. Fig. 5.2 illustrates that S_{Stage2} successfully highlights the foreground regions while removes the background noise appeared in S_{Stage1}.

5.3 Teaching-to-learn and Learning-to-teach For Saliency Propagation

Saliency propagation plays an important role in our algorithm. Suppose we have l seed vertices $\mathbf{s}_1, \cdots, \mathbf{s}_l$ on $\mathcal{G}$ with saliency values $f_1 = \cdots = f_l = 1$, the task of saliency propagation is to reliably and accurately transmit these values from the l labeled vertices to the remaining $u = N - l$ unlabeled superpixels.

As mentioned in Section 5.1, the propagation sequence in existing methods Gopalakrishnan et al. [2009]; Jiang et al. [2013]; Yang et al. [2013b] may incur imperfect results on difficult superpixels, so we propose a novel teaching-to-learn and learning-to-teach framework to optimize the learning sequence (see Fig. 5.3). To be specific, this framework consists of a learner and a teacher. Given the labeled set and unlabeled set at time t denoted as $\mathcal{L}^{(t)}$ and $\mathcal{U}^{(t)}$, the teacher selects a set of simple superpixels from $\mathcal{U}^{(t)}$ as curriculum $\mathcal{T}^{(t)}$. Then, the learner will learn $\mathcal{T}^{(t)}$, and return a feedback to the teacher to help the teacher update the curriculum for the $(t+1)$-th learning. This process iterates until all the superpixels in $\mathcal{U}^{(t)}$ are properly propagated.

5.3.1 Teaching-to-learn

The core of teaching-to-learn is to design a teacher deciding which unlabeled superpixels are to be learned. For the t-th propagation, a candidate set $\mathcal{C}^{(t)}$ is firstly established, in which the elements are vertices directly connected to the labeled set $\mathcal{L}^{(t)}$ on $\mathcal{G}$. Then the teacher chooses the simplest superpixels from $\mathcal{C}^{(t)}$ as the t-th curriculum. To evaluate the propagation difficulty of an unlabeled superpixel $\mathbf{s}_i \in \mathcal{C}^{(t)}$, the difficulty score DS_i is defined by combining informativity INF_i, individuality IND_i, inhomogeneity IHM_i, and connectivity CON_i, namely:

$$DS_i = INF_i + IND_i + IHM_i + CON_i. \tag{5.4}$$

Next we will detail the definitions and computations of INF_i, IND_i, IHM_i, and CON_i, respectively.

Informativity: The simple superpixel should not contain too much information given the labeled set $\mathcal{L}$[1]. Therefore, the informativity of a superpixel $\mathbf{s}_i \in \mathcal{C}$ is

[1]For simplicity, the superscript t is omitted for all the notations hereinafter unless otherwise specified.

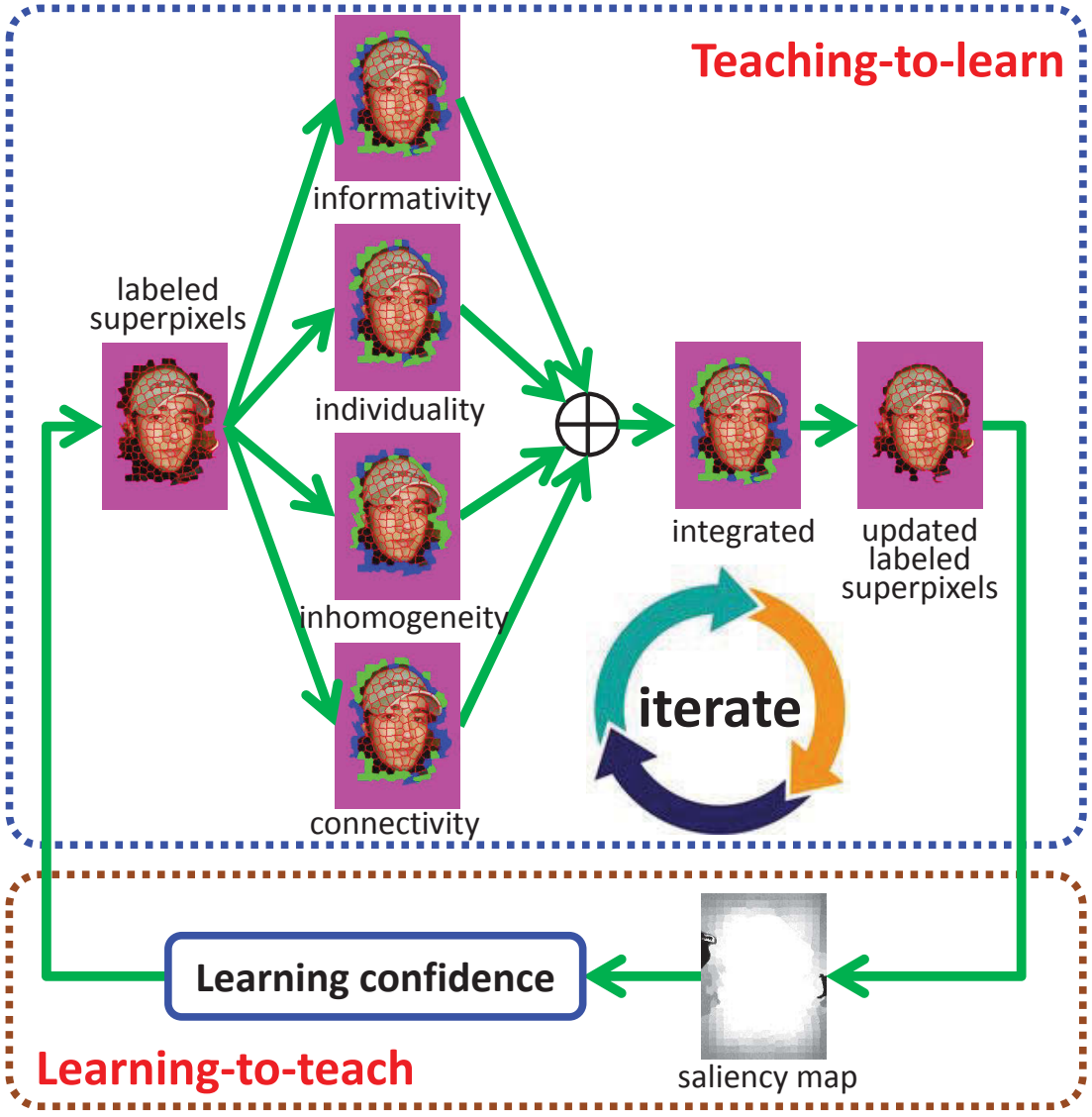

Figure 5.3: An illustration of our teaching-to-learn and learning-to-teach paradigm. In the teaching-to-learn step, based on a set of labeled superpixels (magenta) in an image, the teacher discriminates the adjacent unlabeled superpixels as difficult (blue superpixels) or simple (green superpixels) by fusing their informativity, individuality, inhomogeneity, and connectivity. Then simple superpixels are learned by the learner, and the labeled set is updated correspondingly. In the learning-to-teach step, the learner provides a learning feedback to the teacher to help decide the next curriculum.

straightforwardly modeled by the conditional entropy $H(\mathbf{s}_i|\mathcal{L})$, namely:

$$INF_i = H(\mathbf{s}_i|\mathcal{L}). \tag{5.5}$$

The propagations on the graph follow the multivariate Gaussian process Zhu et al. [2003a], with the elements f_i $(i = 1, \cdots, N)$ in the random vector $\mathbf{f} = \begin{pmatrix} f_1 & \cdots & f_N \end{pmatrix}^\top$ denoting the saliency values of superpixels $\mathbf{s}_i$. The associated covariance matrix $\mathbf{K}$ equals to the adjacency matrix $\mathbf{W}$ except the diagonal elements are set to 1.

For the multivariate Gaussian, the closed-form solution of $H(\mathbf{s}_i|\mathcal{L})$ is Bishop [2006]:

$$H(\mathbf{s}_i|\mathcal{L}) = \frac{1}{2}\ln(2\pi e\sigma^2_{i|\mathcal{L}}), \tag{5.6}$$

where $\sigma^2_{i|\mathcal{L}}$ denotes the conditional covariance of f_i given $\mathcal{L}$. Considering that the conditional distribution is a multivariate Gaussian, $\sigma^2_{i|\mathcal{L}}$ in Eq. (5.6) can be represented by

$$\sigma^2_{i|\mathcal{L}} = \mathbf{K}^2_{ii} - \mathbf{K}_{i,\mathcal{L}}\mathbf{K}^{-1}_{\mathcal{L},\mathcal{L}}\mathbf{K}_{\mathcal{L},i}, \tag{5.7}$$

in which $\mathbf{K}_{i,\mathcal{L}}$ and $\mathbf{K}_{\mathcal{L},\mathcal{L}}$ denote the sub-matrices of $\mathbf{K}$ indexed by the corresponding subscripts. By plugging Eqs. (5.6) and (5.7) into Eq. (5.5), we obtain the informativity of $\mathbf{s}_i$.

In Eq. (5.7), the inverse of an $l \times l$ (l is the size of gradually expanded labeled set $\mathcal{L}$) matrix $\mathbf{K}_{\mathcal{L},\mathcal{L}}$ should be computed in every iteration. As l becomes larger and larger, directly inverting this matrix can be time-consuming. Therefore, we adopt the efficient updating technique designed in Chapter 4 based on the blockwise inversion equation.

Individuality: Individuality measures how distinct of a superpixel to its surrounding superpixels. We consider a superpixel simple if it is similar to the nearby superpixels in the LAB color space. This is because such superpixel is very likely to share the similar saliency value with its neighbors, thus can be easily identified as either foreground or background. For example, the superpixel $\mathbf{s}_2$ in Fig. 5.4(a) has lower individuality than $\mathbf{s}_1$ since it is more similar to the neighbors than $\mathbf{s}_1$. The equation below quantifies the local individuality of $\mathbf{s}_i$ and its neighboring superpixels $\mathcal{N}(\mathbf{s}_i)$:

$$IND_i = IND(\mathbf{s}_i, \mathcal{N}(\mathbf{s}_i)) = \frac{1}{|\mathcal{N}(\mathbf{s}_i)|}\sum_{j\in\mathcal{N}(\mathbf{s}_i)} \left\|\mathbf{s}_i^{color} - \mathbf{s}_j^{color}\right\|, \tag{5.8}$$

where $|\mathcal{N}(\mathbf{s}_i)|$ denotes the amount of $\mathbf{s}_i$'s neighbors. Consequently, the superpixels with small individuality are preferred for the current learning.

Inhomogeneity: It is obvious that a superpixel is ambiguous if it is not homogenous or compact. Fig. 5.4(b) provides an example that the homogenous $\mathbf{s}_4$ gets smaller IHM value than the complicated $\mathbf{s}_3$. Suppose there are b pixels $\left\{\mathbf{p}_j^{color}\right\}_{j=1}^{b}$ in a superpixel $\mathbf{s}_i$ characterized by the LAB color feature, then their pairwise correlations

are recorded in the $b \times b$ symmetric matrix $\mathbf{\Theta} = \mathbf{P}\mathbf{P}^\top$, where $\mathbf{P}$ is a matrix with each row representing a pixel $\mathbf{p}_j^{color}$. Therefore, the inhomogeneity of a superpixel $\mathbf{s}_i$ is defined by the reciprocal of mean value of all the pairwise correlations:

$$IHM_i = \left(\frac{2}{b^2 - b} \sum\nolimits_{i=1}^{b} \sum\nolimits_{j=i+1}^{b} \mathbf{\Theta}_{ij} \right)^{-1}, \tag{5.9}$$

where $\mathbf{\Theta}_{ij}$ is the (i, j)-th element of matrix $\mathbf{\Theta}$. Small IHM_i means that all the pixels in $\mathbf{s}_i$ are much correlated with others, so $\mathbf{s}_i$ is homogenous and can be easily learned.

Connectivity: For the established graph $\mathcal{G}$, a simple intuition is that the vertices strongly connected to the labeled set $\mathcal{L}$ are not difficult to propagate. Such strength of connectivity is inversely proportional to the averaged geodesic distances between $\mathbf{s}_i \in \mathcal{C}$ and all the elements in $\mathcal{L}$, namely:

$$CON_i = \frac{1}{l} \sum\nolimits_{j \in \mathcal{L}} geo(\mathbf{s}_i, \mathbf{s}_j). \tag{5.10}$$

In Eq. (5.10), $geo(\mathbf{s}_i, \mathbf{s}_j)$ represents the geodesic distance between $\mathbf{s}_i$ and $\mathbf{s}_j$, which can be approximated by their shortest path, namely:

$$\begin{aligned} geo(\mathbf{s}_i, \mathbf{s}_j) = & \min_{R_1 = i, R_2, \cdots, R_n = j} \sum\nolimits_{k=1}^{n-1} \max(E_{R_k, R_{k+1}} - c_0, 0) \\ s.t. \quad & R_k, R_{k+1} \in \mathcal{V}, \ \ R_k \text{ and } R_{k+1} \text{ are connected in } \mathcal{G} \end{aligned}. \tag{5.11}$$

Here $\mathcal{V}$ denotes the vertex set of $\mathcal{G}$, $E_{R_k, R_{k+1}}$ computes the Euclidean distance between R_k and R_{k+1}, and c_0 is an adaptive threshold preventing the "small-weight-accumulation" problem Wei et al. [2012].

Finally, by substituting Eqs. (5.5), (5.8), (5.9) and (5.10) into Eq. (5.4), the difficulty scores of all $\mathbf{s}_i \in \mathcal{C}$ can be calculated, based on which the teacher is able to determine the simple curriculum for the current iteration. With the teacher's effort, the unlabeled superpixels are gradually learned from simple to difficult, which is different from the propagation sequence in many existing methodologies Gopalakrishnan et al. [2009]; Jiang et al. [2013]; Ren et al. [2010]; Yang et al. [2013b].

5.3.2 Learning-to-teach

After the difficulty scores of all candidate superpixels are computed, the next step is to pick up a certain number of superpixels as curriculum based on $DS_1, \cdots, DS_{|\mathcal{C}|}$. A straightforward idea is to sort all the elements in $\mathcal{C}$ so that their difficulty scores satisfying $DS_1 \leq DS_2 \leq \cdots \leq DS_{|\mathcal{C}|}$. Then the first q ($q \leq |\mathcal{C}|$) superpixels are used to establish the curriculum set $\mathcal{T} = \{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_q\}$ according to the pre-defined q. However, we hold that how many superpixels are to be learned at t should depend on the $(t-1)$-th learning performance. If the $(t-1)$-th learning is confident, the teacher

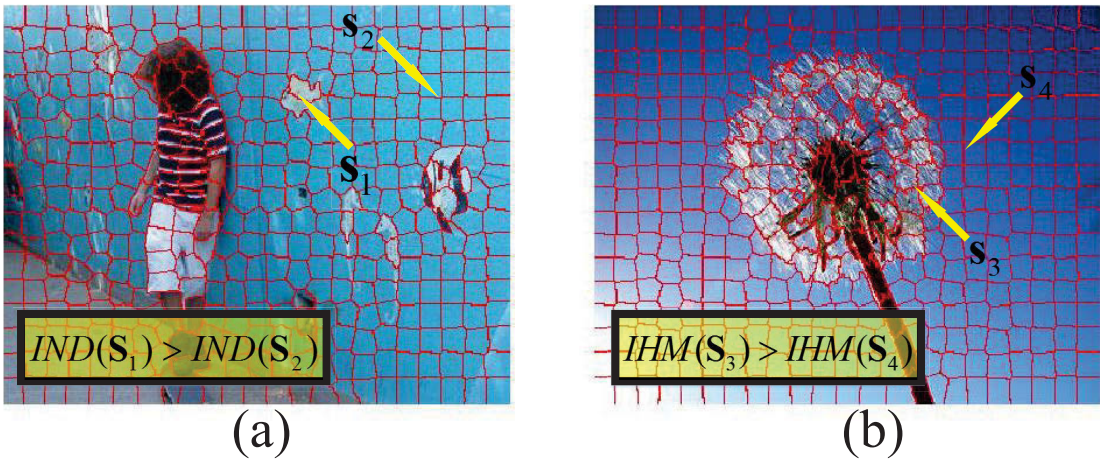

Figure 5.4: The illustrations of individuality (a) and inhomogeneity (b). The region $\mathbf{s}_1$ in (a) obtains larger individuality than $\mathbf{s}_2$, and $\mathbf{s}_3$ in (b) is more inhomogeneous than $\mathbf{s}_4$.

may assign "heavier" curriculum to the learner. In other words, the teacher should also consider the learner's feedback to arrange the proper curriculum, which is a "learning-to-teach" mechanism. Next we will use this mechanism to adaptively decide $q^{(t)}$ for the t-th curriculum.

As mentioned above, $q^{(t)}$ should be adjusted by considering the effect of previous learning. However, since the correctness of the $(t-1)$-th output saliency is unknown, we define a confidence score to blindly evaluate the previous learning performance. Intuitively, the $(t-1)$-th learning is confident if the saliency values $f_1^{(t-1)}, \cdots, f_{q^{(t-1)}}^{(t-1)}$ are close to 0 (very dissimilar to seeds) or 1 (very similar to seeds) after scaling. However, if $f_1^{(t-1)}, \cdots, f_{q^{(t-1)}}^{(t-1)}$ are close to the ambiguous value 0.5, the teacher will rate the last learning as unsatisfactory, and produce a small $q^{(t)}$ to relieve the "burden" for the current learning. Therefore, the confidence score that belongs to $[0, 1]$ is defined by

$$ConfidenceScore = 1 - \frac{2}{q^{(t-1)}} \sum_{i=1}^{q^{(t-1)}} \min(f_i^{(t-1)}, 1 - f_i^{(t-1)}), \tag{5.12}$$

and $q^{(t)}$ is finally computed by

$$q^{(t)} = \left\lceil \left| \mathcal{C}^{(t)} \right| \times ConfidenceScore \right\rceil. \tag{5.13}$$

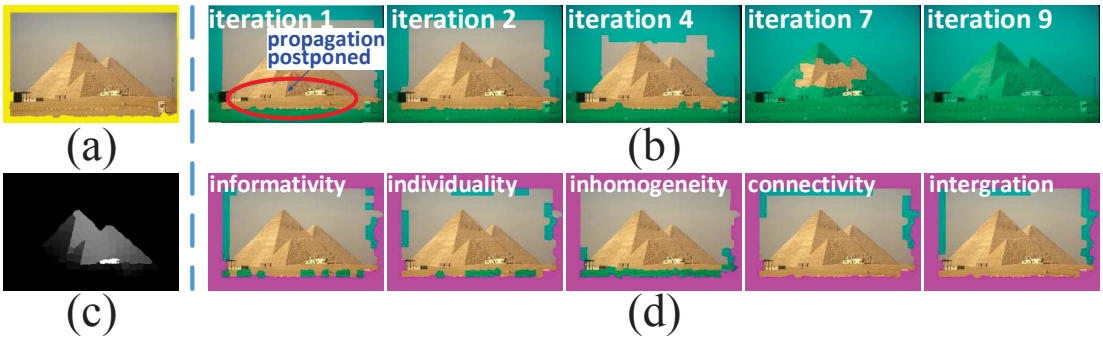

Figure 5.5: Visualization of the designed propagation process. (a) shows the input image with boundary seeds (yellow). (b) displays the propagations in several key iterations, and the expansions of labeled set $\mathcal{L}$ are highlighted with light green masks. (c) is the final saliency map. The curriculum superpixels of the 2nd iteration decided by informativity, individuality, inhomogeneity, connectivity, and the final integrated result are visualized in (d), in which the magenta patches represent the learned superpixels in the 1st propagation, and the regions for the 2nd diffusion are annotated with light green.

5.3.3 Saliency Propagation

After the curriculum $\mathcal{T}^{(t)} = \left\{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{q^{(t)}}\right\}$ is specified, the learner will spread the saliency values from $\mathcal{L}^{(t)}$ to $\mathcal{T}^{(t)}$ via propagation. Particularly, the expression is:

$$\mathbf{f}^{(t+1)} = \mathbf{M}^{(t)}\mathbf{D}^{-1}\mathbf{W}\mathbf{f}^{(t)}, \tag{5.14}$$

where $\mathbf{M}^{(t)}$ is a diagonal matrix with $\mathbf{M}_{ii}^{(t)} = 1$ if $\mathbf{s}_i \in \mathcal{L}^{(t)} \cup \mathcal{T}^{(t)}$, and $\mathbf{M}_{ii}^{(t)} = 0$ otherwise. When the t-th iteration is completed, the labeled and unlabeled sets are updated as $\mathcal{L}^{(t+1)} = \mathcal{L}^{(t)} \cup \mathcal{T}^{(t)}$ and $\mathcal{U}^{(t+1)} = \mathcal{U}^{(t)} - \mathcal{T}^{(t)}$, respectively. Eq. (5.14) initializes from the binary vector $\mathbf{f}^{(0)} = \left(f_1^{(0)}, \cdots, f_N^{(0)}\right)^\top$ ($f_i^{(0)} = 1$ if the i-th superpixel corresponds to seed, and 0 otherwise), terminates when $\mathcal{U}$ becomes an empty set, and the obtained saliency value vector is denoted by $\bar{\mathbf{f}}$. Finally, we smooth $\bar{\mathbf{f}}$ by driving the entire propagation on $\mathcal{G}$ to the stationary state:

$$\mathbf{f}^* = \left(\mathbf{I} - \alpha\mathbf{D}^{-1}\mathbf{W}\right)^{-1}\bar{\mathbf{f}}, \tag{5.15}$$

where α is a parameter set to 0.99 Yang et al. [2013b], and $\mathbf{f}^*$ encodes the saliency information of N superpixels as defined in Section 5.2.2.

One example of the complete propagation process is visualized in Fig. 5.5, in which the superpixels along the image's four boundaries serve as seeds to propagate the saliency information to the remaining superpixels (see Fig. 5.5(a)). In (b), we observe that the sky regions are relatively easy and are firstly learned during the

1st$\sim$4th iterations. In contrast, the land areas are very different from the seeds, so they are difficult and their diffusion should be deferred. Though the labeled set touches the land in a very early time (see the red circle in the 1st iteration), the land superpixels are not diffused until the 4th iteration. This is because the background regions are mostly learned until the 4th iteration, which provide sufficient preliminary knowledge to identify the difficult land regions as foreground or background. As a result, the learner is more confident to assign the correct saliency values to the land after the 4th iteration, and the target (pyramid) is learned in the end during the 7th$\sim$9th iterations. More concretely, the effect of our curriculum selection approach is demonstrated in Fig. 5.5(d). It can be observed that though the curriculum superpixels are differently chosen by their informativity, individuality, inhomogeneity, and connectivity, they are easy to learn based on the previous accumulated knowledge. Particularly, we notice that the final integrated result only preserves the sky regions for the leaner, while discards the land areas though they are recommended by informativity, individuality, and inhomogeneity. This further reduces the erroneous propagation possibility since the land looks differently from the sky and actually more similar to the unlearned pyramid. Therefore, the fusion scheme (5.4) and the proper $q^{(t)}$ decided by the learning-to-teach step are reasonable and they are critical to the successful propagations (see Fig. 5.5(c)).

5.4 Experimental Results

In this section, we qualitatively and quantitatively compare the proposed **T**eaching-to-**L**earn and **L**earning-to-**T**each approach (abbreviated as "TLLT") with twelve popular methods on two popular saliency datasets. The twelve baselines include classical methods (LD Liu et al. [2007], GS Wei et al. [2012]), state-of-the-art methods (SS Fu et al. [2014], PD Margolin et al. [2013], CT Kim et al. [2014], RBD Zhu et al. [2014b], HS Yan et al. [2013], SF Perazzi et al. [2012]), and representative propagation based methods (MR Yang et al. [2013b], GP Fu et al. [2013], AM Jiang et al. [2013], GRD Yang et al. [2013a]). We adopt the weighted precision Precision^w, weighted recall Recall^w and weighted F_β-measure F_β^w proposed by Margolin et al. [2014] for performance evaluation. The parameters in our method are set to $N = 400$ and $\theta = 0.25$ throughout the experiments. The parametric sensitivity and failed cases are also discussed at the end of this section.

5.4.1 Experiments on Public Datasets

The MSRA 1000 dataset Liu et al. [2007], which contains 1000 images with binary pixel-level groundtruth, is firstly adopted for our experiments. The average precision^w, recall^w, and F_β^w of all the methods are illustrated in Fig. 5.6(a). We can observe that

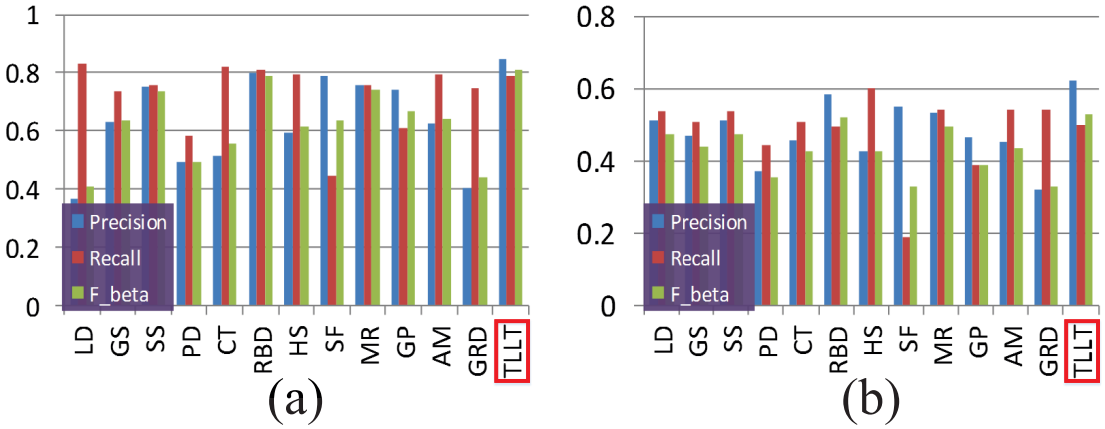

Figure 5.6: Comparison of different methods on two saliency detection datasets. (a) is MSRA 1000, and (b) is ECSSD.

the F_β^w of our TLLT is larger than 0.8, which is the highest record among all the comparators. Another notable fact is that TLLT outperforms other baselines with a large margin in Precision^w. This is because the designed teaching-to-learn and learning-to-teach paradigm propagates the saliency value carefully and accurately. As a result, our approach has less possibility to generate the blurred saliency map with confused foreground. In this way, the Precision^w is significantly improved. More importantly, we note that the Recall^w of our method also touches a relatively high value, although the Precision^w has already obtained an impressive record. This further demonstrates the strength of our innovation.

Although the images from MSRA 1000 dataset have a large variety in their content, the foreground is actually prominent among the simple and structured background. Therefore, a more complicated dataset ECSSD Yan et al. [2013], which represents more general situations that natural images fall into, is adopted to further test all the algorithms. Fig. 5.6(b) shows the result. Generally, all methods perform more poorly on ECSSD than on the MSRA 1000. However, our algorithm still achieves the highest F_β^w and Precision^w when compared with other baselines. RBD obtains slightly lower F_β^w than our method with 0.5215 compared to 0.5284, but the weighted precision is not as good as our approach. Besides, some methods such as HS, SF and GRD obtain very moderate results since they tend to detect the most salient regions at the expense of low precision. As a result, the imbalance between Precision^w and Recall^w will happen, which pulls down the overall F_β^w to a low value. Comparatively, TLLT produces relatively balanced Precision^w and Recall^w on both datasets, therefore higher F_β^w is obtained.

The average CPU seconds of evaluated methods for processing one image in ECSSD are summarized in Table 5.1, on an Intel i5 3.20GHz CPU with 8GB RAM.

Table 5.1: Average CPU seconds of all the approaches on ECSSD dataset

Method	LD	GS	SS	PD	CT	RBD	HS	SF	MR	GP	AM	GRD	TLLT
Duration (s)	7.24	0.18	3.58	2.87	3.53	0.20	0.43	0.19	0.87	3.22	0.15	0.93	2.31
Code	matlab	matlab	matlab	matlab	matlab	matlab	C++	matlab	matlab	matlab	matlab	matlab	matlab

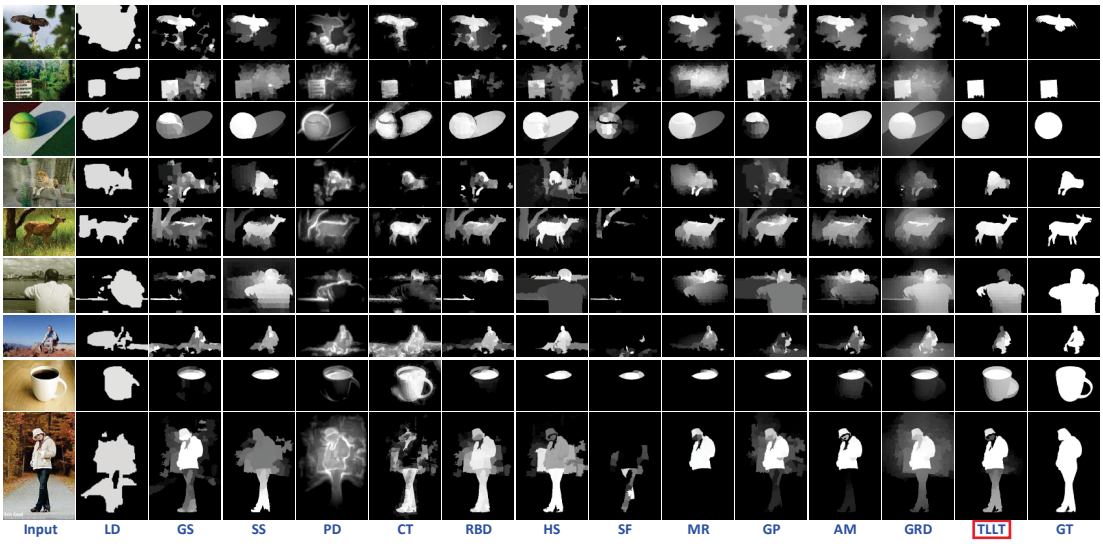

Figure 5.7: Visual comparisons of saliency maps generated by all the methods on some challenging images. The ground truth (GT) is presented in the last column.

our method takes 2.31 seconds per detection, which is slower than GS, RBD, HS, SF, MR, AM, GRD, but faster than LD, SS, PD, CT, and GP. Because our method needs to decide the suitable curriculum in every iteration, it needs relatively longer computational time. The iteration times for a normal image under our parametric settings are usually 5$\sim$15. However, better results can be obtained as shown in Fig. 5.6, at the cost of more computational time.

To further present the merits of the proposed approach, we provide the resulting saliency maps of evaluated methods on several very challenging images from the two datasets (see Fig. 5.7). Though the backgrounds in these images are highly complicated, or very similar to the foregrounds, TLLT is able to generate fairly confident and clean saliency maps. In other words, TLLT is not easily confused by the unstructured background, and can make a clear distinction between the complex background and the regions of interest.

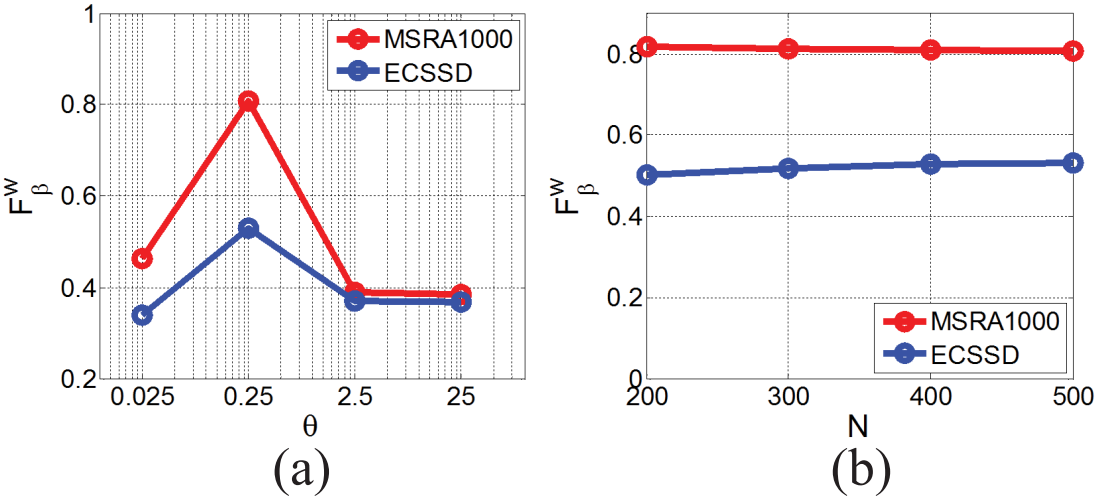

Figure 5.8: Parametric sensitivity analyses: (a) shows the variation of F_β^w w.r.t. θ by fixing $N = 400$; (b) presents the change of F_β^w w.r.t. N by keeping $\theta = 0.25$.

5.4.2 Parametric Sensitivity

There are two free parameters in our algorithm to be manually tuned: Gaussian kernel width θ and the amount of superpixels N. We evaluate each of the parameters θ and N by examining F_β^w with the other one fixed. Fig. 5.8 reveals that F_β^w is not sensitive to the change of N, but heavily depends on the choice of θ. Specifically, it can be observed that the highest records are obtained when $\theta = 0.25$ on both datasets, so we adjust θ to 0.25 for all the experiments.

5.4.3 Failed Cases

Though the proposed TLLT achieves very impressive results in most cases, it may fail if 1) the target is extremely similar to the background which significantly confuses the propagation; and 2) the generated convex hull completely misses the salient object. The two examples corresponding to above situations are presented in Fig. 5.9. In Fig. 5.9(a), the color of the target is very close to the background, therefore the generated saliency map (Fig. 5.9(b)) is far from perfect even though the seed superpixels are correct. In Fig. 5.9(c), the convex hull encloses the non-target regions, so the real targets are not precisely detected in the final saliency map (Fig. 5.9(d)). However, this case seldom occurs according to the extensive experiments in prior works Fu et al. [2013]; Xie et al. [2013]; Yang et al. [2013a]. The failure rates of the convex hull in MSRA 1000 and ECSSD are 6/1000 and 12/1000, respectively. Actually, the two listed situations 1) and 2) are also challenging for the existing saliency algorithms.

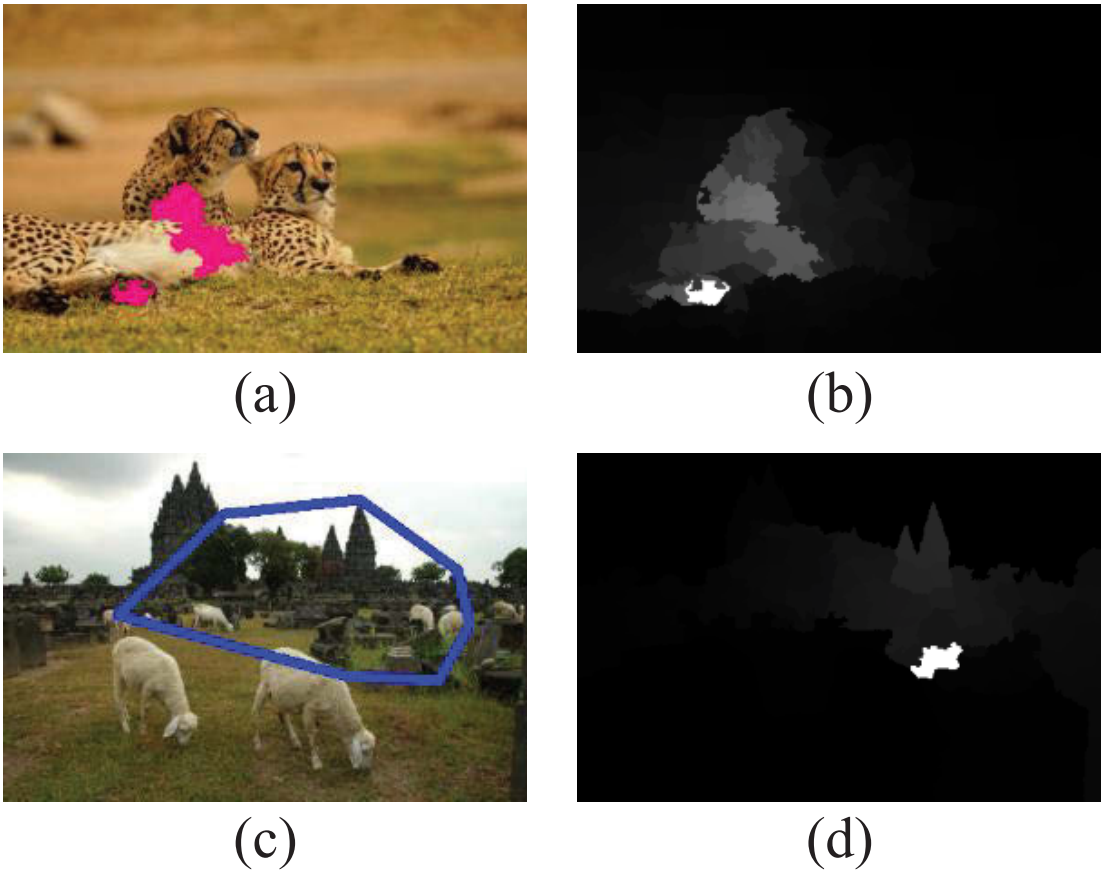

Figure 5.9: Failed cases of our method. (a) shows an example that the object is very similar to the background, in which the correct seed superpixels are marked with magenta. (b) is the imperfect saliency map corresponding to the image in (a). In (c), the targets are completely missed by the convex hull (blue polygon), which leads to the detection failure as revealed by (d).

5.5 Summary of This Chapter

This chapter proposed a novel approach for saliency propagation through leveraging a teaching-to-learn and learning-to-teach paradigm. Different from the existing methods that propagated the saliency information entirely depending on the relationships among adjacent image regions, the proposed approach manipulated the propagation sequence from simple regions to difficult regions, thus leading to more reliable propagations. Consequently, our approach can render a more confident saliency map with higher background suppression, yielding a better popping out of objects of interest.

Chapter 6

Conclusion and Future Work

In this section, we first conclude the entire thesis, and then elaborate the possible trends for future research.

6.1 Thesis Summarization

This thesis addresses the problem of robust graph transduction. In practical situations, there are usually a large number of noisy data, which significantly impair the performance of existing graph transduction methods. Therefore, we designed various robust algorithms that can suppress the adverse impact of the noisy data from different aspects. Specifically, one non-iterative method LPDGL, and two iterative approaches FLAP and TLLT were proposed, among which TLLT was further shown to be effective for saliency detection tasks.

In LPDGL (Chapter 2), we introduced a deformed graph Laplacian, which originates from the deformed differential operator developed in mathematical physics. As a consequence, the resulting smoothness term not only contains the existing pairwise term, but also incorporates a novel local smoothness term. This local smoothness term assigns confident labels to the examples with large degree, as well as allocates "weak labels" to the uncertain examples with small degree. Therefore, the negative effects of outliers or "bridge points" can be decreased, leading to robust transduction performance. The free parameters appeared in our regularization framework are also easy to tune because the final performance is proved to be insensitive to their changes. Moreover, we show that our transductive model can be easily extended to inductive settings, and the robustness and generalizability are also theoretically guaranteed. LPDGL has a variety of applications such as handwritten digit recognition, face recognition and violent behavior detection.

In FLAP (Chapter 3), we drew an analogy between label propagation and practical

fluid diffusion, and leveraged a well-known physical theory called Fick's Law of Diffusion to achieve "natural" label propagation. In other words, when and how much label information is received or transferred by an example, or where these labels should be propagated to, are naturally governed. This is very different from most of the existing machine learning algorithms which are established based on some heuristic and ad hoc criteria. As a result, FLAP yields more robust propagation result than the existing methods. Besides, we show that another advantage of our FLAP is that the eigenvalues of the associated iteration matrix are regularly distributed, leading to faster convergence rate than the existing propagation algorithms. Furthermore, we prove that although FLAP is derived from the viewpoint of physical theory, and it has a close relationship with the traditional regularization networks, Markov random fields, and graph kernels. Comprehensive experimental results on both synthetic and practical datasets suggest that FLAP obtains very encouraging performance when compared with state-of-the-art algorithms.

In TLLT (Chapter 4), we considered that the the traditional propagation methods have some defects for handling the difficult examples with uncertain labels. Therefore, we proposed to establish a logical propagation sequence from simple examples to more difficult ones under the guidance of a knowledgable "teacher". The teacher can assess the difficulty (or classification reliability) of each unlabeled example, and only arrange the simplest unlabeled examples (*i.e.* a curriculum) to the learner for the current propagation. After "learning" this curriculum, the learner delivers a learning feedback to the teacher to assist it in deciding the next simplest curriculum. Due to the interactions between the teacher and learner, the improved accuracy over existing methods is achieved, and the classification performance is also shown to be robust to the choice of graph parameter.

Finally, in Chapter 5 we applied the framework of TLLT to detect the most salient objects in an image. The difficulty of a superpixel is judged by its informativity, individuality, inhomogeneity, and connectivity. In the designed saliency propagation process, the simple superpixels are decided as foreground or background prior to the difficult superpixels. As a result, our detector generates manifest saliency maps, and meanwhile outperforms baseline methods revealed by both qualitative and quantitative comparisons.

6.2 Relationship and Differences among Algorithms

Although the proposed three algorithms (LPDGL, FLAP, and TLLT) in this paper are able to conduct graph transduction, they have different purposes and are applicable to different scenarios. When the dataset is small, such as the *Iris* and *Wine* datasets appeared in Sections 2.4.3, 3.4.3 and 4.8.2, the classification accuracies of LPDGL

and TLLT are slightly higher than FLAP. This is because some specific measures are taken to deal with the noisy data in LPDGL and TLLT. For example, LPDGL utilizes the degree information of every example, while TLLT identifies the difficulty levels of unlabeled examples and explicitly establishes a classification sequence from simple examples to more difficult ones. In contrast, FLAP does not have a specific mechanism to handle noise, so it is slightly worse than LPDGL and TLLT in terms of classification accuracy. However, LPDGL and TLLT are only suitable for small-scale datasets, while FLAP is applicable to relatively large datasets. This is because the eigenvalues of the iteration matrix in FLAP are distributed regularly, so FLAP is able to converge to the stationary point efficiently. This has been theoretically verified in Section 3.2, and empirically demonstrated in Figs. 3.6, 3.7, 3.8, 3.9 and Table 3.5. Although LPDGL has a closed-form solution as indicated in (2.7), it has to compute the inverse of an $n \times n$ matrix (n is the number of examples), therefore the optimal solution of LPDGL can be efficiently obtained when the amount of examples is not large. However, if we are faced with a large dataset, such matrix inversion will be quite time-consuming, and at this time LPDGL will be less efficient that FLAP. For TLLT, it has to solve an optimization problem (4.8) to generate the optimal curriculum examples in each iteration, so TLLT takes longer computational time than FLAP when the size of dataset is large. Besides, Section 2.2 indicates that LPDGL can be easily extended to obtain a general decision function to cope with induction cases, and this is the advantage of LPDGL over TLLT and FLAP. Therefore, LPDGL can be an ideal choice if we want to accomplish graph transduction as well as obtain a general decision function to predict the labels of unseen examples in the future.

6.3 Future Work

Although the algorithms proposed in this thesis have achieved encouraging results to some extent, some issues still remain open and should be further investigated:

- **Graph construction.** Obviously, all the graph transduction algorithms are based on a established graph. However, how to build a suitable graph that can faithfully reflect the intrinsic structure of the dataset is still to be investigated. A well-constructed graph is beneficial to boosting the performance of our transduction algorithms including LPDGL, FLAP, and TLLT. To this end, we need to take a further insight into the data itself, and also utilize the practical domain knowledge to construct a suitable graph.

- **Scalability.** Almost all the current graph transduction methods are not scalable to large datasets, because they usually involve some computationally heavy factors, such as large matrix multiplication, large matrix inverse or slow

convergence rate. For example, LPDGL contains a large matrix inverse in Eq. (2.7), FLAP needs a large matrix multiplication in Eq. (3.11), and TLLT has a overhead of optimizing the simplest curriculum via solving Eq. (4.7). Therefore, more efficient transductive models and the related solvers are to be developed.

- **Multi-label handling.** All the algorithms proposed in this thesis are only able to handle single-label data. However, it is often the case that the real data can be assigned a set of different labels, such as image annotation. Consequently, our algorithms should be improved to handle multi-label cases by further considering the correlations among the different labels.

- **Heterogeneous data.** Practically, data may come from different sources. For example, an image can be characterized by different feature descriptors like SIFT, RGB, HOG, etc. A webpage usually contains both images and text. Directly concatenating the heterogeneous features into one long feature vector ignores the specific property within each of the features. Therefore, our algorithms should be extended to multi-view settings, so that they can better exploit the complementary information carried by different kinds of features.

- **Label noise handling.** Currently, our algorithms including LPDGL, FLAP and TLLT are only robust to outliers, "bridge points" or the variations of some critical parameters. However, label noise is another important issue that should be taken into consideration for future algorithm design. For example, due to the different expertise of labelers and the difficulty of tasks, the labels collected via crowdsourcing often contain noise, making them not trustable for the following analyses. Therefore, making our algorithms robust to noisy labels is a direction that worths further studying.

- **More applications.** It will be valuable if the proposed algorithms can be applied to more practical problems. Although the experimental results reveal that our algorithms have achieved some initial success in image classification, digit recognition, text classification and saliency detection, we think that our algorithms are also promising for tackling some other applications, such as bioinformatics data analyses, remote sensing image classification, and video processing.

References

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, 2012. 106

E. Alpaydin. Combined 5× 2 cv f test for comparing supervised classification learning algorithms. *Neural computation*, 11(8):1885–1892, 1999. 36, 37, 65

C. Alzate and J. Suykens. A semi-supervised formulation to binary kernel spectral clustering. In *Proc. of the IEEE World Congress on Computational Intelligence*, pages 1992–1999, 2012. 54

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc*, 68(3):337–404, 1950. 22

A. Azran. The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In *Proc. International Conference on Machine Learning*, pages 49–56, Oregon, USA, 2007. 4, 8

X. Bai, X. Yang, L. Jan, W. Liu, and Z. Tu. Learning context-sensitive shape similarity by graph transduction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(5):861–874, 2010. 8

F. Balbach and T. Zeugmann. Teaching randomized learners. In *Learning Theory*, pages 229–243. Springer, 2006. 75

M. Belkin and P. Niyogi. Towards a theoretical foundation for laplacian based manifold methods. *Learning theory*, pages 835–851, 2005. 6

M. Belkin and P. Niyogi. Towards a theoretical foundation for laplacian-based manifold methods. *Journal of Computer and System Sciences*, 74(8):1289–1308, 2008. 6

M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *Annual Conference on Learning Theory*, volume 3120, pages 624–638. Springer, 2004a. 6

M. Belkin, I. Matveeva, and P. Niyogi. Tikhonov regularization and semi-supervised learning on large graphs. In *Acoustics, Speech, and Signal Processing (ICASSP). IEEE International Conference on*, volume 3. IEEE, 2004b. 6, 29

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434, 2006. 4, 6, 7, 18, 23, 30, 31

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proc. International Conference on Machine Learning*, pages 41–48. ACM, 2009. 75, 92

D. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014. 81

C. Bishop. *Pattern recognition and machine learning*, volume 1. springer New York, 2006. 110

A. Blum, J. Lafferty, and R. Rwebangira. Semi-supervised learning using randomized mincuts. In *Proc. International Conference on Machine Learning*, pages 13–20, 2004. 4

X. Cai, F. Nie, W. Cai, and H. Huang. Heterogeneous image features integration via multi-modal semi-supervised learning model. In *IEEE International Conference on Computer Vision*, pages 1737–1744, 2013. 7

H. Chang and D. Yeung. Robust path-based clustering for the unsupervised and semi-supervised learning settings. Technical report, 2004. 4, 30, 31

O. Chapelle, B. Schölkopf, and A. Zien. Semi-supervised learning. 2006. 4, 30, 33, 50, 60

G. Chen, Y. Song, F. Wang, and C. Zhang. Semi-supervised multi-label learning by solving a sylvester equation. In *SIAM International Conference on Data Mining*, pages 410–419. SIAM, 2008. 7

L. Chen, I. Tsang, and D. Xu. Laplacian embedded regression for scalable manifold regularization. *Neural Networks and Learning Systems, IEEE Transactions on*, 23 (6):902–915, 2012a. 30

L. Chen, I. Tsang, and D. Xu. Laplacian embedded regression for scalable manifold regularization. *Neural Networks and Learning Systems, IEEE Transactions on*, 23 (6):902–915, Jun 2012b. 6

X. Chen, Y. Mu, H. Liu, S. Yan, Y. Rui, and T. Chua. Large-scale multilabel propagation based on efficient sparse graph construction. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 10(1):6, 2013. 4

M. Cheng, G. Zhang, N. Mitra, X. Huang, and S. Hu. Global contrast based salient region detection. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 409–416. IEEE, 2011. 104

A. Conn, N. Gould, A. Sartenaer, and P. Toint. Convergence properties of an augmented lagrangian algorithm for optimization with a combination of general equality and linear constraints. *SIAM Journal on Optimization*, 6(3):674–703, 1996. 82

C. André R D. Sousa, R. Solange, and B. Gustavo. Influence of graph construction on semi-supervised learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 160–175. Springer, 2013. 4

O. Dekel and O. Shamir. Good learners for evil teachers. In *Proc. International Conference on Machine Learning*, pages 233–240. ACM, 2009. 75

O. Dekel, C. Gentile, and K. Sridharan. Selective sampling and active learning from single and multiple teachers. *The Journal of Machine Learning Research*, 13(1): 2655–2697, 2012. 75

C. Ding, T. Li, and D. Wang. Label propagation on k-partite graphs. In *International Conference on Machine Learning and Applications*, pages 273–278. IEEE, 2009. 7

J. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993. 75, 105

A. Erdem and M. Pelillo. Graph transduction as a noncooperative game. *Neural Computation*, 24(3):700–723, 2012. 8

Y. Fang, K. Chang, and H. Lauw. Graph-based semi-supervised learning: Realizing pointwise smoothness probabilistically. In *Proc. International Conference on Machine Learning*, pages 406–414, 2014. 9

R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collections. In *Advances in neural information processing systems*, pages 522–530, 2009. 6

R. Fletcher. On the barzilai-borwein method. In *Optimization and control with applications*, pages 235–256. Springer, 2005. 82

C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2): 214–225, 2004. 86

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`. 34, 96

K. Fu, C. Gong, I. Gu, and J. Yang. Geodesic saliency propagation for image salient region detection. In *Image Processing (ICIP), IEEE Conference on*, pages 3278–3282, 2013. 107, 114, 117

K. Fu, C. Gong, I. Gu, J. Yang, and X. He. Spectral salient object detection. In *Multimedia and Expo (ICME), IEEE International Conference on*, 2014. 104, 114

Y. Fujiwara and G. Irie. Efficient label propagation. In *Proc. International Conference on Machine Learning*, pages 784–792, 2014. 8

A. Gammerman, V. Vovk, and V. Vapnik. Learning by transduction. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 148–155. Morgan Kaufmann Publishers Inc., 1998. 1

D. Gao, S. Han, and N. Vasconcelos. Discriminant saliency, the detection of suspicious coincidences, and applications to visual recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(6):989–1005, 2009. 104

B. Gary, R. Manu, B. Tamara, and L. Erik. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, University of Massachusetts, Amherst, 2007. 40

A. Goldberg, X. Zhu, and S. Wright. Dissimilarity in graph-based semi-supervised classification. In *International Conference on Artificial Intelligence and Statistics*, pages 155–162, 2007. 7

A. Goldberg, X. Zhu, A. Singh, Z. Xu, and R. Nowak. Multi-manifold semi-supervised learning. In *International Conference on Artificial Intelligence and Statistics*, pages 169–176, 2009. 7

A. Goldberg, B. Recht, J. Xu, R. Nowak, and X. Zhu. Transduction with matrix completion: Three birds with one stone. In *Advances in Neural Information Processing Systems*, pages 757–765, 2010. 8

S. Goldman and M. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995. 75

G. Golub and V. Loan. *Matrix computations*, volume 3. Johns Hopkins University Press, 1996. 50, 52, 54, 86

C. Gong, K. Fu, Q. Wu, E. Tu, and J. Yang. Semi-supervised classification with pairwise constraints. *Neurocomputing*, 139:130–137, 2014a. 7

C. Gong, D. Tao, K. Fu, and J. Yang. Fick's law assisted propagation for semi-supervised learning. *Neural Networks and Learning Systems, IEEE transactions on*, 2014b. x, 77

V. Gopalakrishnan, Y. Hu, and D. Rajan. Random walks on graphs to model saliency in images. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1698–1705. IEEE, 2009. 104, 108, 111

Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, pages 529–536, 2004. 7

W. Hager. Updating the inverse of a matrix. *SIAM review*, 31(2):221–239, 1989. 86, 87, 88

N. Higham. *Accuracy and stability of numerical algorithms*. Siam, 1996. 51

P. Hislop and I. Sigal. *Introduction to Spectral Theory: With Applications to Schrodinger Operators*. Springer, 1996. 19

T. Hofmann, B. Schölkopf, and A. Smola. A tutorial review of RKHS methods in machine learning. 2005. 22

X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1–8. IEEE, 2007. 104

T. Huang and C. Yang. *The analysis and applications of special matrix*. Science Press, 2007. 69

Y. Huang, D. Xu, and F. Nie. Semi-supervised dimension reduction using trace ratio criterion. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(3): 519–526, 2012. 7

L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(11):1254–1259, 1998. 104

T. Jebara, J. Wang, and S. Chang. Graph construction and b-matching for semi-supervised learning. In *Proc. International Conference on Machine Learning*, pages 441–448. ACM, 2009. 4

M. Ji, T. Yang, B. Lin, R. Jin, and J. Han. A simple algorithm for semisupervised learning with improved generalization error bound. In *Proc. International Conference on Machine Learning*, 2012. 4, 30

B. Jiang, L. Zhang, H. Lu, C. Yang, and M. Yang. Saliency detection via absorbing markov chain. In *Computer Vision (ICCV), IEEE International Conference on*, pages 1665–1672. IEEE, 2013. xi, 104, 108, 111, 114

H. Jiang, J. Wang, Z. Yuan, T. Liu, N. Zheng, and S. Li. Automatic salient object segmentation based on context and shape prior. In *British Machine Vision Conference (BMVC)*, pages 1–12, 2011. 104

L. Jiang, D. Meng, T. Mitamura, and A. Hauptmann. Easy samples first: self-paced reranking for zero-example multimedia search. In *ACM MM*, pages 547–556, 2014a. 75

L. Jiang, D. Meng, S. Yu, Z. Lan, S. Shan, and A. Hauptmann. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pages 2078–2086, 2014b. 75

L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. Hauptmann. Self-paced curriculum learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2015. 75

T. Joachims. Transductive learning via spectral graph partitioning. In *Proc. International Conference on Machine Learning*, pages 290–297, 2003. 6, 29, 46, 57, 62

M. Karasuyama and H. Mamitsuka. Manifold-based similarity adaptation for label propagation. In *Advances in Neural Information Processing Systems*, pages 1547–1555, 2013a. 4, 88

M. Karasuyama and H. Mamitsuka. Multiple graph label propagation by sparse integration. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(12): 1999–2012, 2013b. 4, 7

M. Karlen, J. Weston, A. Erkan, and R. Collobert. Large scale manifold transduction. In *Proc. International Conference on Machine Learning*, Helsinki, Finland, 2008. 6

F. Khan, B. Mutlu, and X. Zhu. How do humans teach: On curriculum learning and teaching dimension. In *Advances in Neural Information Processing Systems*, pages 1449–1457, 2011. 75, 105

J. Kim, D. Han, Y. Tai, and J. Kim. Salient region detection via high-dimensional color transform. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 883–890. IEEE, 2014. 104, 114

K. Kim and J. Theobalt. Semi-supervised learning with explicit relationship regularization. *IEEE International Conference on Computer Vision and Pattern Recognition*, 74(8):1289–1308, 2015. 8

R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proc. International Conference on Machine Learning*, volume 2, pages 315–322, 2002. 56

X. Kong, M. Ng, and Z. Zhou. Transductive multilabel learning via label set propagation. *Knowledge and Data Engineering, IEEE Transactions on*, 25(3):704–719, 2013. 7

M. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2010. 75, 92

N. Kumar, A. Berg, P. Belhumeur, and S. Nayar. Attribute and simile classifiers for face verification. In *Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society Conference on*, pages 365–372. IEEE, 2009. 42, 65

P. Lancaster and M. Tismenetsky. *Theory of matrices*, volume 2. Academic press New York, 1969. 21

C. Lee, A. Varshney, and D. Jacobs. Mesh saliency. In *ACM Transactions on Graphics*, volume 24, pages 659–666. ACM, 2005. 104

Y. Lee and K. Grauman. Learning the easy things first: Self-paced visual category discovery. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1721–1728. IEEE, 2011. 75

D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004. 97

S. Li and Y. Fu. Low-rank coding with b-matching constraint for semi-supervised classification. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1472–1478. AAAI Press, 2013. 4

X. Li and Y. Guo. Adaptive active learning for image classification. In *Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society Conference on*, 2013. 17

Y. Li and R. Zemel. High order regularization for semi-supervised learning of structured output problems. In *Proc. International Conference on Machine Learning*, pages 1368–1376, 2014. 8

Y. Li and Z. Zhou. Towards making unlabeled data never hurt. In *Proc. International Conference on Machine Learning*, pages 1081–1088, 2011. 4, 30, 31

Y. Li and Z. Zhou. Towards making unlabeled data never hurt. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(1):175–188, 2015. 4

Y. Li, J. Kwok, and Z. Zhou. Semi-supervised learning using label mean. In *Proc. International Conference on Machine Learning*, pages 633–640. ACM, 2009. 4

Y. Li, X. Hou, C. Koch, J. Rehg, and A. Yuille. The secrets of salient object segmentation. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 280–287. IEEE, 2014. 104

F. Lin and W. Cohen. Adaptation of graph-based semi-supervised methods to largescale text data. In *The 9th Workshop on Mining and Learning with Graphs*, 2011. 8

T. Liu, J. Sun, N. Zheng, X. Tang, and H. Shum. Learning to detect a salient object. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1–8. IEEE, 2007. 114

W. Liu and S. Chang. Robust multi-class transductive learning with graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 381–388. IEEE, 2009. 7

W. Liu, J. He, and S. Chang. Large graph construction for scalable semi-supervised learning. In *Proc. International Conference on Machine Learning*, pages 679–686, Haifa, Israel, 2010. 4, 30, 31, 36

T. Ma and L. Jan. Graph transduction learning with connectivity constraints with application to multiple foreground cosegmentation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1955–1962. IEEE, 2013. 7

R. Margolin, A. Tal, and L. Zelnik-Manor. What makes a patch distinct? In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1139–1146. IEEE, 2013. 104, 114

R. Margolin, L. Zelnik-Manor, and A. Tal. How to evaluate foreground maps. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 248–255. IEEE, 2014. 114

S. Maybank. A probabilistic definition of salient regions for image matching. *Nuerocomputing*, 120(23):4–14, 2013. 104

S. Mei and X. Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2015. 75

F. Morbidi. The deformed consensus protocol. *Automatica*, 49(10):3049–3055, 2013. 16, 19

E. Nievas, O. Suarez, G. García, and R. Sukthankar. Violence detection in video using computer vision techniques. In *Computer Analysis of Images and Patterns*, pages 332–339. Springer, 2011. 43

G. Niu, B. Dai, C. Plessis, and M. Sugiyama. Transductive learning with multi-class volume approximation. In *Proc. International Conference on Machine Learning*, pages 1377–1385, 2014. 8

M. Orbach and K. Crammer. Graph-based transduction with confidence. In *ECML-PKDD*. 2012. 7

K. Patil, X. Zhu, Ł. Kopeć, and B. Love. Optimal teaching for limited-capacity human learners. In *Advances in Neural Information Processing Systems*, pages 2465–2473, 2014. 75

F. Perazzi, P. Krahenbuhl, Y. Pritch, and A. Hornung. Saliency filters: Contrast based filtering for salient region detection. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 733–740. IEEE, 2012. 104, 114

K. Petersen and M. Pedersen. The matrix cookbook. *Technical University of Denmark*. 28, 87

H. Qiu and E. Hancock. Clustering and embedding using commute times. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(11):1873–1890, 2007. 80

M. Quang, L. Bazzani, and V. Murino. A unifying framework for vector-valued manifold regularization and multi-view learning. In *Proc. International Conference on Machine Learning*, pages 100–108, 2013. 4, 30

Z. Ren, Y. Hu, L. Chia, and D. Rajan. Improved saliency detection based on superpixel clustering and saliency propagation. In *Proceedings of the International Conference on Multimedia*, pages 1099–1102. ACM, 2010. 105, 111

D. Rohde and D. Plaut. Language acquisition in the absence of explicit negative evidence: How important is starting small? *Cognition*, 72(1):67–109, 1999. 105

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. 41

H. Rue and L. Held. *Gaussian markov random fields: theory and applications*. Chapman & Hall, 2005. 55

J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004. 56

J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000. 6

A. Shinohara and S. Miyano. Teachability in computational learning. *New Generation Computing*, 8(4):337–347, 1991. 75

A. Singla, I. Bogunovic, G. Bartok, A. Karbasi, and A. Krause. Near-optimally teaching the crowd to classify. In *Proc. International Conference on Machine Learning*, pages 154–162, 2014. 75

K. Sinha and M. Belkin. Semi-supervised learning using sparse eigenfunction bases. In *Advances in Neural Information Processing Systems*, pages 1687–1695, 2009. 8

J. Solomon, R. Rustamov, L. Guibas, and A. Butscher. Wasserstein propagation for semi-supervised learning. In *Proc. International Conference on Machine Learning*, pages 306–314, 2014. 9

A. Subramanya and J. Bilmes. Semi-supervised learning with measure propagation. *The Journal of Machine Learning Research*, 12:3311–3370, 2011. 7

J. Supancic and D. Ramanan. Self-paced learning for long-term tracking. In *CVPR*, pages 2379–2386, 2013. 75

M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, pages 945–952, Vancouver, Canada, 2002a. 8

M. Szummer and T. Jaakkola. Information regularization with partially labeled data. In *Advances in Neural Information Processing Systems*, pages 1025–1032, 2002b. 7

K. Tang, V. Ramanathan, F. Li, and D. Koller. Shifting weights: Adapting object detectors from image to video. In *NIPS*, pages 638–646, 2012. 75

J. Taylor and W. Thompson. An introduction to error analysis: the study of uncertainties in physical measurements. *Measurement Science and Technology*, 9 (6):1015, 1998. 90

W. Tong and R. Jin. Semi-supervised learning by mixed label propagation. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 651, 2007. 9

V. Vapnik. Statistical learning theory, 1998. 4, 30

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society Conference on*. IEEE, 2001. 42, 65

B. Wang and Z. Tu. Affinity learning via self-diffusion for image segmentation and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2312–2319. IEEE, 2012. 9

B. Wang, Z. Tu, and J. Tsotsos. Dynamic label propagation for semi-supervised multi-class multi-label classification. In *IEEE International Conference on Computer Vision*, pages 425–432. IEEE, 2013. viii, 9, 10, 94

F. Wang and C. Zhang. Label propagation through linear neighborhoods. In *Proc. International Conference on Machine Learning*, pages 985–992, 2006. 9, 57

F. Wang and C. Zhang. Label propagation through linear neighborhoods. *Knowledge and Data Engineering, IEEE Transactions on*, 20(1):55–67, 2008. 9

F. Wang, T. Li, G. Wang, and C. Zhang. Semi-supervised classification using local and global regularization. In *The AAAI Conference on Artificial Intelligence (AAAI)*, pages 726–731, 2008a. viii, 7, 10, 30

H. Wang, H. Huang, and C. Ding. Image annotation using multi-label correlated Green's function. In *IEEE International Conference on Computer Vision*, pages 2029–2034. IEEE, 2009a. 7

J. Wang and Y. Xia. Fast graph construction using auction algorithm. *arXiv preprint arXiv:1210.4917*, 2012. 4

J. Wang, T. Jebara, and S. Chang. Graph transduction via alternating minimization. In *Proc. International Conference on Machine Learning*, pages 1144–1151, Helsinki, Finland, 2008b. 4, 9, 30, 53, 94, 96

J. Wang, F. Wang, and C. Zhang. Linear neighborhood propagation and its applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(9):1600–1615, 2009b. viii, 4, 9, 10, 11, 30, 31, 46, 53, 55, 57, 62, 94

Y. Wang and S. Chen. Safety-aware semi-supervised classification. *Neural Networks and Learning Systems, IEEE Transactions on*, 2013. 4, 30

Y. Wang, S. Chen, and Z. Zhou. New semi-supervised classification method based on modified cluster assumption. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(5):689–702, 2012. 4, 30

Y. Wei, F. Wen, W. Zhu, and J. Sun. Geodesic saliency using background priors. In *European Conference on Computer Vision (ECCV)*, pages 29–42. Springer, 2012. 107, 111, 114

K. Weinberger, F. Sha, and L. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proc. International Conference on Machine Learning*, pages 839–846, Banff, Canada, 2004. 76

Z. Wen and W. Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013. 81

M. Wu and B. Schölkopf. Transductive classification via local learning regularization. In *International Conference on Artificial Intelligence and Statistics*, pages 628–635, 2007. 4, 7, 30, 31

X. Wu, Z. Li, M. Anthony, J. Wright, and S. Chang. Learning with partially absorbing random walks. In *Advances in Neural Information Processing Systems*, pages 3077–3085, 2012. 8

S. Xiang, F. Nie, and C. Zhang. Semi-supervised classification via local spline regression. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32 (11):2039–2053, 2010. 7

Y. Xie, H. Lu, and M. Yang. Bayesian saliency via low and mid level cues. *Image Processing, IEEE Transactions on*, 22(5):1689–1698, 2013. 117

H. Xu and S. Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012. 23, 24, 26

Z. Xu, I. King, M. Lyu, and R. Jin. Discriminative semi-supervised feature selection via manifold regularization. *Neural Networks, IEEE Transactions on*, 21(7):1033–1047, 2010. 7

W. Yan, L. Xu, J. Shi, and J. Jia. Hierarchical saliency detection. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1155–1162. IEEE, 2013. 114, 115

C. Yang, L. Zhang, and H. Lu. Graph-regularized saliency detection with convex-hull-based center prior. *Signal Processing Letters, IEEE*, 20(7):637–640, 2013a. 106, 107, 114, 117

C. Yang, L. Zhang, H. Lu, X. Ruan, and M. Yang. Saliency detection via graph-based manifold ranking. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 3166–3173. IEEE, 2013b. xi, 104, 105, 107, 108, 111, 113, 114

J. Yang and M. Yang. Top-down visual saliency via joint CRF and dictionary learning. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 2296–2303. IEEE, 2012. 104

X. Yang, X. Bai, L. Jan, and Z. Tu. Improving shape retrieval by learning graph transduction. In *European Conference on Computer Vision*, pages 788–801. Springer, 2008. 8

A. Yu and K. Grauman. Fine-grained visual comparisons with local learning. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 192–199. IEEE, 2014. 99

C. Zhang and F. Wang. A multilevel approach for learning from labeled and unlabeled data on graphs. *Pattern Recognition*, 43(6):2301–2314, 2010. 4

D. Zhou and O. Bousquet. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pages 321–328, Vancouver, Canada, 2003. viii, x, 4, 8, 9, 10, 11, 18, 29, 31, 46, 53, 57, 62, 77, 94

D. Zhou and B. Schölkopf. Learning from labeled and unlabeled data using random walks. *Pattern Recognition*, pages 237–244, 2004. 4

D. Zhou, T. Hofmann, and B. Schölkopf. Semi-supervised learning on directed graphs. In *Advances in Neural Information Processing Systems*, pages 1633–1640, 2004a. 9

D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. *Advances in Neural Information Processing Systems*, 16:169–176, 2004b. 8, 9, 105

J. Zhu, Y. Qiu, R. Zhang, J. Huang, and W. Zhang. Top-down saliency detection via contextual pooling. *Journal of Signal Processing Systems*, 74(1):33–46, 2014a. 104

W. Zhu, S. Liang, Y. Wei, and J. Sun. Saliency optimization from robust background detection. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 2814–2821. IEEE, 2014b. 114

X. Zhu. Machine teaching for bayesian learners in the exponential family. In *Advances in Neural Information Processing Systems*, pages 1905–1913, 2013. 75

X. Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2015. 75

X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, CMU-CALD-02-107, 2002. viii, x, 8, 9, 10, 11, 12, 75, 76, 77, 83, 84, 94

X. Zhu and A. Goldberg. Introduction to semi-supervised learning. 2009. 4, 30, 50

X. Zhu and Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Proc. International Conference on Machine Learning*, pages 1052–1059, Bonn, Germany, 2005. 4, 6, 30, 62, 64

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proc. International Conference on Machine Learning*, pages 912–919, 2003a. 1, 4, 6, 18, 29, 31, 46, 57, 62, 88, 110

X. Zhu, J. Lafferty, and Z. Ghahramani. Semi-supervised learning: From Gaussian fields to Gaussian processes. Technical report, CMU-CS-03-175, 2003b. 78

X. Zhu, J. Kandola, and Z. Ghahramani. Nonparametric transforms of graph kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2005. 8, 46, 57

S. Zilles, S. Lange, R. Holte, and M. Zinkevich. Models of cooperative teaching and learning. *The Journal of Machine Learning Research*, 12:349–384, 2011. 75