Study on Semi-Supervised Classification Based on Laplacian Kernel Machines Using Quasi-Linear Kernel

Yanni REN

February 2022

Waseda University Doctoral Dissertation

Study on Semi-Supervised Classification Based on Laplacian Kernel Machines Using Quasi-Linear Kernel

Yanni REN

Graduate School of Information, Production and Systems
Waseda University

February 2022

Abstract

Classification is one fundamental research topic in machine learning, which aims to recognize objects and separate them into classes. A classification model formulates the separation boundary between different classes, and generally, a nonlinear separation boundary is needed. Learning of a nonlinear classification model equals modeling a nonlinear separation boundary. Traditionally, classification model learning has been studied in the supervised scheme where all the training data instances have accurate labels. However, labeled data is expensive in contrast to unlabeled data. Therefore, semi-supervised classification (SSC) has gained prominence, which leverages a large amount of unlabeled data in addition to a small amount of labeled data for training. Usually, intrinsic SSC methods are extensions of existing supervised methods to include unlabeled data in the objective function.

Laplacian kernel machines, namely, Laplacian Support Vector Machine (LapSVM) and Laplacian Regularized Least Square (LapRLS), are among the most result-promising semi-supervised classification methods. They are extensions of supervised kernel machines, Support Vector Machine (SVM), and Regularized Least Square (RLS) by adding a graph regularization in the objective function of model parameter optimization. Kernel defines a linearly separable high-dimensional feature space, and a linear model in the feature space corresponds to a nonlinear model in the input space. The use of graph leverages unlabeled data by approximating data manifold, where data instance as nodes are sparsely connected by edges. The kernel is used again as edge weighting. Note that the kernel is used twice in Laplacian kernel machines, and its quality directly influences the performance of the classification model. General nonlinear kernels, such as radial basis function (RBF) kernels, implicitly define a general feature space. It is a black-box model from a modeling perspective, and prior knowledge cannot be used even if given.

In this dissertation, we are motivated to apply a two-step modeling method to model the nonlinear separation boundary using a set of linear models. The model parameters are estimated in two steps. In the first step, the nonlinear parameters connecting or combining the linear models are estimated. Then the classification model is formulated as a regression form with a known regression vector and a parameter vector. The parameter vector contains all the linear parameters to be estimated in the second step. In the second step, the linear parameters of all the linear models are estimated globally. The classification model can be further recast to a kernel form as an intermediate model.

The kernel is defined as the inner product of the know regression vectors, namely, quasi-linear kernel. As a result, the quasi-linear kernel is composed in an interpretable way, and it contains prior knowledge.

Although the quasi-linear kernel has been studied in many tasks, exploiting it by leveraging a small amount of labeled data and a large amount of unlabeled data remains challenging. Therefore, we focus on its study in this dissertation to achieve accurate performance. We propose a series of semi-supervised classification algorithms based on Laplacian kernel machines through the construction of an intermediate model named quasi-linear kernel.

The dissertation contains the following five chapters as follows:

Chapter 1 first introduces the concepts mentioned above, such as nonlinear classification, semi-supervised classification, and Laplacian kernel machines. Then, we discuss the insufficiency of general kernels from the modeling perspective and introduce the two-step modeling method and the quasi-linear kernel. At last, we list challenges under the semi-supervised context, on which we will give corresponding solutions in the following chapters.

Chapter 2 proposes a Laplacian SVM based semi-supervised classifier using multi-local linear model. The semi-supervised classifier is constructed in two steps. In the first step, by applying a pseudo-labeling approach, the input space is divided into multiple local linearly separable partitions along the potential separation boundary. A multi-local linear model is then built by interpolating multiple local linear models assigned to the partitions. In the second step, the multi-local linear model is formulated as a linear regression form with a new regression vector containing the information of potential separation boundary. Then all the linear parameters are optimized globally by a LapSVM algorithm using a quasi-linear kernel function defined as the inner product of the new regression vectors. Furthermore, the quasi-linear kernel function and the pseudo labels are used to construct a label-guided graph. As a result, the potential separation boundary is detected, and its information is incorporated into a LapSVM in kernel and graph levels. Numerical experiments exhibit the effectiveness of the proposed method by showing better performance against general kernel LapSVM with a “win/tie/lose=7/1/0” on 8 real-world datasets under 10% labeled data.

Chapter 3 proposes a semi-supervised classifier based on piecewise linear model using gated linear network. The semi-supervised classifier is constructed in two steps. In the first step, we design a label-guided autoencoder-based semi-supervised gating mechanism to generate binary sequences. By using a gated linear network, the binary sequences realize partitioning of a piecewise linear model indirectly. In the second step, the piecewise linear model is formulated as a linear regression form, and the linear parameters are then optimized globally by a LapRLS algorithm using a quasi-linear kernel function comprising the binary sequences. Moreover, the quasi-linear kernel function is used as a better similarity function for the graph construction. As a result, we estimate data manifold from both labeled and unlabeled data, and the data manifold is incorporated into both the kernel and the graph in LapRLS. The experimental results validate the effectiveness of the proposed method by showing a “win/tie/lose = 7/0/0” on 7 University of Cambridge Irvine (UCI) data sets compared to other SSC methods when 10% data is labeled.

Chapter 4 applies the proposed semi-supervised classifier based on piecewise linear model to parasite images, including a semi-supervised feature extractor based on deep CNN using contrastive learning. First, for the deep CNN feature extractor, we introduce real-world images with similar and clear semantic information to enhance the structure at the representation level. In addition, we introduce variant appearance transformations to eliminate the texture at the representation level. Second, a gated linear network is adopted as the classifier to realize a piecewise linear separation boundary. The linear parameters are optimized globally by a LapSVM algorithm using a quasi-linear kernel function composed of the representations and the binary sequences generated from the learned feature extractor. In summary, the proposed semi-supervised method tackles the structure and texture challenges and achieves accurate parasite classification. The proposed method shows better performance than state-of-the-art SSC methods when only 1% of microscopic images are labeled. It reaches an accuracy of 95.10% in a generalized testing set.

Chapter 5 concludes the dissertation and provides future works. To conclude, this dissertation proposes a series of semi-supervised classification algorithms based on Laplacian kernel machines (Laplacian SVM, Laplacian RLS) through the construction of an intermediate model named quasi-linear kernel. In this way, we effectively leverage a small amount of labeled and a large amount of unlabeled data for training to achieve accurate performance on the testing set. Numerical simulation results on a wide range of benchmarks and real-world data sets demonstrate the effectiveness of the proposed semi-supervised classification algorithms.

Preface

The general theme of this dissertation is to develop a set of semi-supervised classification algorithms based on Laplacian kernel machines using quasi-linear kernel to achieve accurate classification performance. This dissertation is organized in five chapters. Most of the materials have been published in following listed journal papers and conference papers.

The materials in Chapter 2 are related to a journal paper

- [J3] Y. Ren, H. Zhu, Y. Tian and J. Hu, “A Laplacian SVM based Semi-Supervised Classification Using Multi-Local Linear Model”, *IEEJ Transactions on Electrical and Electronic Engineering*, Vol.16, No.3, pp.455-463, March, 2021.

which formulates the major content of this chapter.

The material in Chapter 3 are related to

- [P3] Y. Ren, W. Li and J. Hu, “A Semi-Supervised Classification Using Gated Linear Model”, in *Proceedings of 2019 IEEE International Joint Conference on Neural Networks (IJCNN’2019) (Budapest)*, July, 2019. (7 pages)

which has been extended into a journal paper and formulated the major content of this chapter

- [J5] Y. Ren, W. Li and J. Hu, “A Semi-Supervised Classifier Based on Piecewise Linear Regression Model Using Gated Linear Network”, *IEEJ Transactions on Electrical and Electronic Engineering*, Vol.15, No.7, pp.1048-1056, July, 2020.

The materials in Chapter 4 are related to

- [P1] Y. Ren, H. Deng, H. Jiang, H. Zhu and J. Hu, “A Semi-Supervised Classification Method of Apicomplexan Parasites and Host Cell Using Contrastive Learning Strategy”, in *Proceedings of 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC’2021) (online)*, October, 2021. (6 pages) which has been extended into a journal paper and formulated the major content of this chapter

- [J1] Y. Ren, H. Jiang, H. Zhu, Y. Tian and J. Hu, “A Semi-Supervised Classification Method of Parasites Using Contrastive Learning”, *IEEJ Trans. on Electrical and Electronic Engineering*, Vol.17, No.3, March, 2022. (9 pages)

Acknowledgements

Time flies, the past few years in Waseda University have been full of rewarding. There are many who helped me along the way on this journey. I want to take a moment to thank them.

Foremost, I would like to express my deep and sincere gratitude to my research supervisor Jinglu Hu for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. He went above and beyond to help me reach my goal.

Besides, I would like to thank the rest of my dissertation committee. Professor Fujimura Shigeru and Professor Yoshie Osamu served as wise committee members for their guidance, encouragement, and insightful comments.

My sincere thanks also go to Dr. Weite Li for influential guidance in the beginning of my research. I would also like to acknowledge all of the other students no matter graduated or not, too numerous to name all of them.

Last but not the least, I would like to thank my parents: your love and understanding helped me through the dark times. Without you believing in me, I never would have made it. It is time to celebrate; you earned this degree right along with me.

Kitakyushu, Japan

Yanni REN

September 18th, 2021

Contents

List of Figures

List of Tables

Abbreviations

SSC	Semi-Supervised Classification
SVM	Support Vector Machine
RLS	Regularized Least Square
LapSVM	Laplacian Support Vector Machine
LapRLS	Laplacian Regularized Least Square
NN	Neural Network
DNN	Deep Neural Network
k-NN	k-Nearest Neighbors
MLP	Multilayer Perceptron
ReLU	Rectified Linear Unit
RBF	Radial Basis Function
PCG	Preconditioned Conjugate Gradient
S-K algorithm	Schlesinger-Kozinec algorithm
CH	Convex Hull
UCI	University of Cambridge Irvine
HM	Harmonic Mixtures
GCLI	Graph Construction based on Labeled Instances
TE	Temporal Embedding
MT	Mean Teacher
VAT	Virtual Adversarial Training
EMA	Exponential Moving Average
ROC	Receiver Operating Characteristic
AUC	Area Under the Curve

Symbols

X	input
Y	output
f	model
$\mathcal{P}_X$	marginal distribution
$\mathcal{P}_{XY}$	joint distribution
$\mathcal{X}_L = \{(x_l, y_l)\}_{l=1}^{L}$	labeled data in training set
$X_U = \{x_u\}_{u=L+1}^{N}$	unlabeled data in training set
$\phi(\cdot)$	mapping function
α	Lagrange multiplier
$k(x_i, x_j)$	kernel function
W	graph matrix
E	graph topology
$\mathbf{L}$	graph Laplacian
Θ	linear parameter of a hyperplane
Ω_j	linear parameters in partitions
b, b_j	bias parameter
ℓ	objective function (loss function)
$g_j(x)$	gate control signals
μ_j, δ_j	center and radius of j-th partition
$a(\cdot)$	activation function

Chapter 1

Introduction

1.1 Classification

In the field of machine learning, for every task, the learning subject is a model [1]. Given an input x, a model provides an output y; the input and output vary from different tasks. And a model $f : X \to Y$ contains model structure and model parameters.

There are two examples, first for the simplest binary classification task in 2-dimensional space. The input x is a point on a plane, and the output $y = sign[f(x)]$ is to which class the point belongs. The model structure is a linear separation boundary $f(x;\Theta,b) = \Theta^T x + b$ in a linearly separable case, and the model parameters are linear parameters Θ, b. Second, for complex tasks, a deep neural network (DNN) can be used. The input could be anything, such as an image, a recording, or a video. The output could be what is it in the given image, the emotion in the given recording, and any abnormality in the given video. The model structure could be very complex, and there are a vast number of model parameters.

Learning for a specific task, we design specific model structure and optimize model parameters by specific strategies. We *train* a model by using a set of data instances called as *training set*. The training goal is to find an f that maximum or minimum the predefined *objective function* on the training set. To evaluate the performance of the trained (learned) model, we *test* it on *testing set*, which is never used in the training phase.

There is an important rule, the more model parameters in a model, the more training data is needed to optimize these model parameters.

A classification model f predicts the input x to which class it belongs, and the output y is limited in the given classes.

$$y = sign[f(x)] \tag{1.1}$$

A classification model formulates the separation boundary to separate data instances from different classes. Learning of classification model equals modeling the separation boundary.

1.1.1 Linear classification

The linear model takes the form of

$$f(x;\Theta,b) = \Theta^T x + b \tag{1.2}$$

once the parameters Θ, b have been learned. In the case of binary classification, if $f(x) > 0$, the data instance x can be labeled to positive class. If $f(x) \leq 0$, the data instance x can be labeled to negative class.

1.1.2 Nonlinear classification

Generally, a linear model cannot separate most data sets, and a nonlinear model is needed. From the modeling perspective, learning of a nonlinear classification model equals modeling a nonlinear separation boundary.

There are two typical nonlinear models with different model structures [2].

- **Kernel Machines**

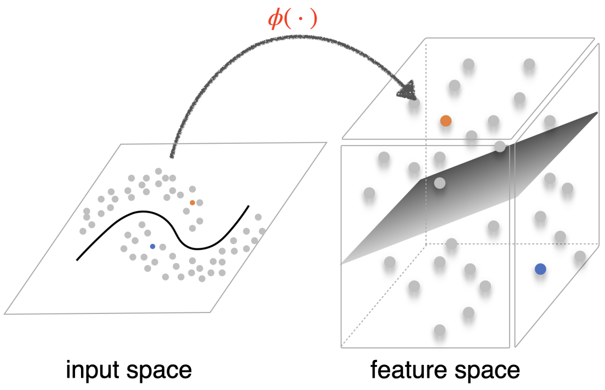

FIGURE 1.1: A linear model in the feature space corresponds to a nonlinear separation boundary in the input space.

As illustrated in Fig.1.1, suppose the data instances are mapped from the input space to a high-dimensional linearly separable feature space. We can realize the nonlinear classification in the input space by a linear model in the feature space. A kernel function $k(\cdot)$ is the similarity of two data instances in the feature space.

$$f(x;\alpha,b) = \alpha^T \mathbf{k}(x) + b \tag{1.3}$$

- **Neural Networks**

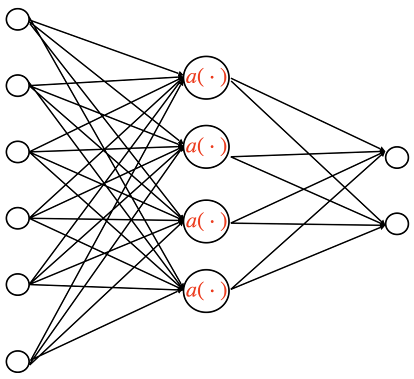

FIGURE 1.2: A neural network with a nonlinear hidden layer.

As illustrated in Fig.1.2, we double the linear transformation with a nonlinear activation function $a(\cdot)$ in the middle step so that we end up with a nonlinear model

$$f(x;\Theta_1,\Theta_2,b_1,b_2) = \Theta_2^T[a(\Theta_1^T x + b_1)] + b_2 \tag{1.4}$$

1.2 Semi-Supervised Classification

Traditionally, the learning of classification model has been studied in supervised schemes where all the data instances in the training set have accurate labels. Labeled data, however, is often difficult or time-consuming to collect.

The lack of the labeled data motivates the study of methods trained with limited supervision, such as semi-supervised classification [3], weakly supervised learning [4, 5], unsupervised domain adaptation [6, 7] and transfer learning [8, 9].

My work focuses on semi-supervised classification (SSC) involving unlabeled data in the training phase. SSC is typically with a small amount of labeled data and a large amount of unlabeled data in the training set since unlabeled data is much easier to collect. Fig.1.3 is a toy example to illustrate the difference between supervised learning and semi-supervised classification. The separation boundary has changed significantly after introducing the unlabeled data.

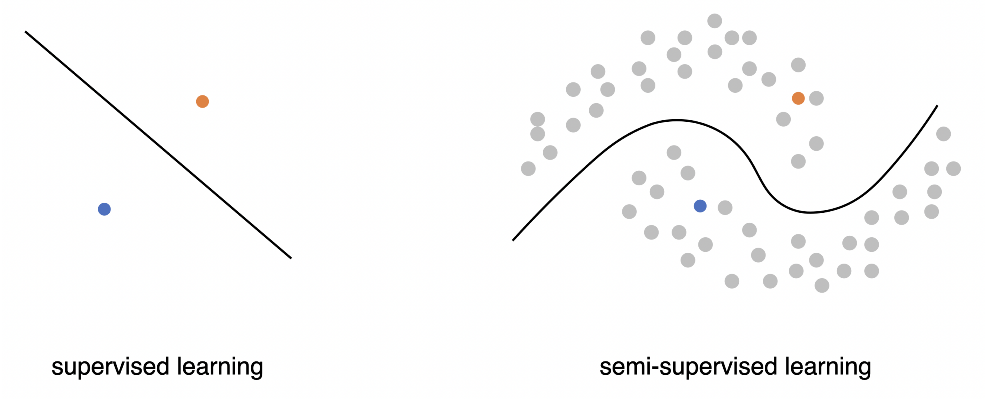

FIGURE 1.3: Binary classification in 2-dimensional space. Blue and orange points stand for labeled data of two classes, respectively, and gray points stand for unlabeled data.

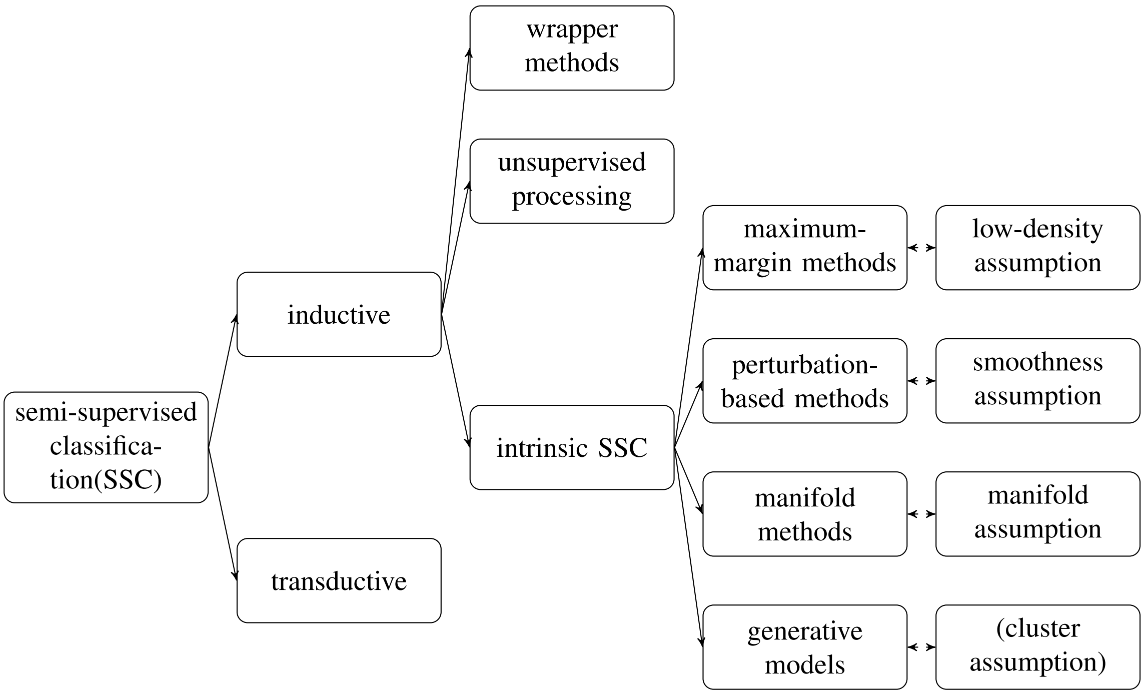

FIGURE 1.4: Semi-supervised classification taxonomy.

1.2.1 Prerequisite

In the case of a semi-supervised classification problem, labeled data (x, y) is sampled from joint distribution $\mathcal{P}_{XY}$ while unlabeled data $x \in X$ is sampled from marginal distribution $\mathcal{P}_X$. The training set $\mathcal{X}_N = \mathcal{X}_L \cup X_U$ contains N data instances, where L labeled data $\mathcal{X}_L = \{(x_l, y_l)\}_{l=1}^{L}$ and U unlabeled data $X_U = \{x_u\}_{u=L+1}^{N}$, and $X_N = X_L \cup X_U$.

There is a very important prerequisite, the distribution of labeled data $\mathcal{P}_{XY}$ and unlabeled data $\mathcal{P}_X$ has to be the same.

1.2.2 Semi-supervised classification methods

The taxonomy of semi-supervised classification is visualized in Fig.1.4. The first distinction lies between inductive and transductive methods. The former yields a classification model which can predict the label of previously unseen testing data. The latter does not yield such a model but instead only provides predictions of unlabeled training data [10–17].

The simplest inductive approach is to first train a classification model on labeled data, and use the predictions of unlabeled data as additional labeled data. Then re-train on both the additional labeled data and the existing labeled data. Such methods are known as wrapper methods [18–26]. Secondly, unsupervised preprocessing methods, which extract useful features in an unsupervised manner [27–35]. And then can be used with any supervised classification model.

Definitely, intrinsic SSC should be the focus, which directly incorporates unlabeled data into the objective function or optimization procedure of the learning, and does not rely on any supervised base learners or unsupervised intermediate steps. Usually, the intrinsic SSC methods are extensions of existing supervised methods to include unlabeled data in objective function.

Intrinsic SSC can be roughly divided into different categories based on different assumptions; low-density assumption [36–42], smoothness assumption, manifold assumption, and cluster assumption [43–49] is a generalization of the above three. In the recent decade, most state-of-the-art methods are based on smoothness assumption and manifold assumption. So is my doctoral work.

- **Smoothness Assumption**

  Smoothness assumption is the very basic assumption that assumes if two data instances that are close by in the input space, their labels should be the same.

- **Manifold Assumption**

  Manifold assumption assumes

  1. high-dimensional data instances lie (roughly) on low-dimensional manifolds;
  2. the data instances lying on the same manifold have the same label.

Usually, smoothness assumption and manifold assumption are not used independently. I use shallow and deep structures to categorize.

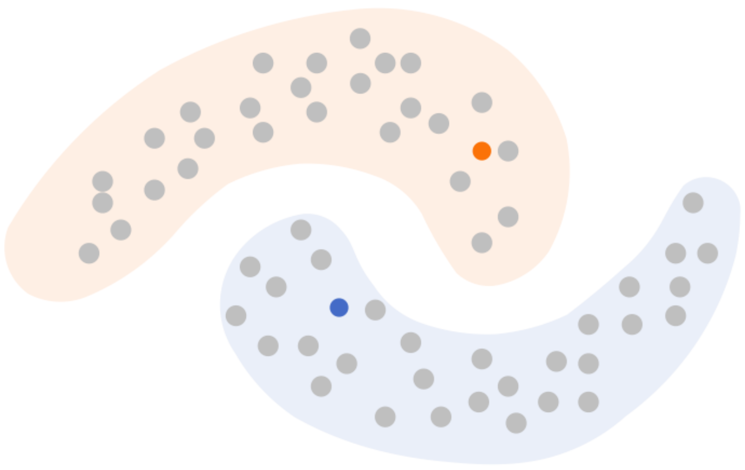

FIGURE 1.5: There are two manifolds in the toy example.

- **Shallow Structured Models**

  Shallow structured models take *features* as inputs. Usually, add a manifold regularization to the objective function of a kernel machine or others. The manifold is approximated by a graph, so it can be called graph regularization.

  Belkin et al. [50] formulated a general framework for regularizing inductive learners based on manifolds. This general framework leads to semi-supervised extensions of kernel machines, such as Laplacian Regularized Least Squares (LapRLS) and Laplacian support vector machines (LapSVM) [51]. Zhu et al. [52] proposed to incorporate a manifold regularization term in a generative model. Sindhwani et al. [53] extend manifold regularization to the co-regularization framework. Qi et al. [54] suggested to extend twin SVMs, which optimize two SVM-like objective functions to yield two non-parallel decision boundaries (one for each class), to include the LapSVM regularization term.

- **Deep Structured Models**

  Deep structured models target *structured data* like images and contain both feature extraction and downstream classification. Usually, add an unsupervised loss term to the training of a DNN. The manifold is estimated by the DNN.

  Weston et al. [55] incorporated a manifold regularization term into deep neural networks. Bachman et al. [56] proposed a general framework for perturbing the neural network model itself. Rasmus et al. [57] proposed ladder network to explicitly perturb the input data. Miyato et al. [58] proposed a regularization procedure that takes the perturbation direction into account.

Deep structured models focus on feature extraction, while shallow structured models focusing on classification when given features. Both feature extraction and classification are essential for a task. My doctoral research mainly revolves around classification.

1.3 Related Works

1.3.1 Laplacian kernel machines

Proposed by Belkin et al. [50, 51, 59], Laplacian kernel machines are among the most result promising semi-supervised classification algorithms, which belongs to an intrinsic SSC method using smoothness and manifold assumptions in an inductive manner.

Laplacian kernel machines, namely, Laplacian Support Vector Machine (LapSVM) and Laplacian Regularized Least Square (LapRLS), are the extensions of supervised kernel machines, Support Vector Machine (SVM) and Regularized Least Square (RLS), by solely adding a manifold regularization which is estimated by a graph Laplacian associated with all the training data. Laplacian kernel machines realize nonlinear classification by leveraging unlabeled data in addition to labeled data.

Kernel composition and graph construction are the model structure design in Laplacian kernel machines.

- **Kernel** realizes nonlinear classification by defining a feature space.

  According to Mercer's theorem, any continuous symmetric positive semi-definite function can be used as a kernel function in Eq.(1.3) since such a function corresponds to an implicit mapping function. General nonlinear kernels such radial base function (RBF) kernel and polynomial kernel

$$
\begin{aligned}
k_{RBF}(x_i, x_j) &= \exp(-\gamma \|x_i - x_j\|_2^2) \\
k_{poly}(x_i, x_j) &= (x_i^T x_j + c)^d
\end{aligned}
\tag{1.5}
$$

implicitly map data from the input space to the general feature space. Note that the kernel stands for similarity of data instances in the feature space, and larger value stands for higher similarity.

- **Graph** leverages unlabeled data by approximating data manifold.

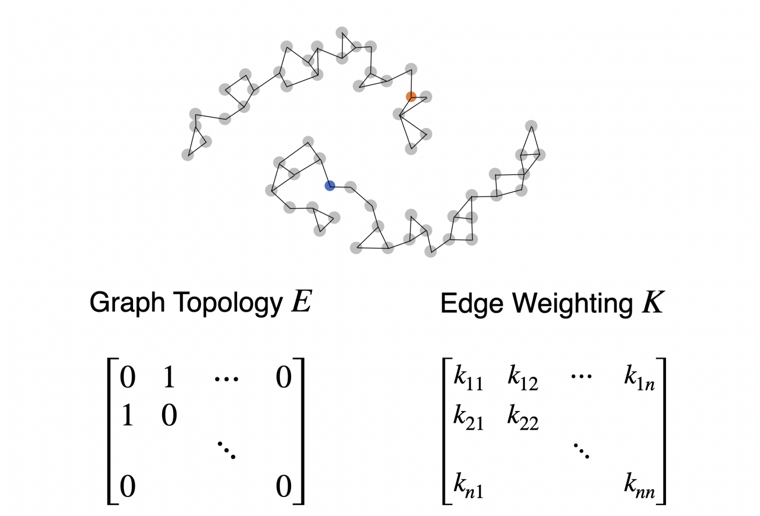
Graph Topology E
Edge Weighting K
$\begin{bmatrix} 0 & 1 & \cdots & 0 \\ 1 & 0 & & \\ & & \ddots & \\ 0 & & & 0 \end{bmatrix}$
$\begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1n} \\ k_{21} & k_{22} & & \\ & & \ddots & \\ k_{n1} & & & k_{nn} \end{bmatrix}$

FIGURE 1.6: Graph to approximate the data manifold.

The graph matrix W can be decomposed to a graph topology E and an edge weighting K.

$$W = E * K \tag{1.6}$$

Taking each data instance in the training set as a node, as illustrated in Fig1.6, the nodes are sparsely connected by edge based on some similarity measure. E is the graph topology matrix, if there is an edge between the nodes x_i and x_j , $E_{ij} = 1$; otherwise, $E_{ij} = 0$, and there are no self-loops, $E_{ii} = 0 \forall i = 1, \cdots, N$. The edge weighting K evaluates the weight of each edge, a kernel is used.

Note that the kernel is used again in the graph as edge weighting.

1.3.2 LapSVM and LapRLS

Given the model structure, LapSVM and LapRLS are good choices to optimize the model parameters α and b in Eq.(1.3).

LapSVM and LapRLS extend SVM and RLS respectively by adding a graph regularization. Their objective functions vary in the supervised loss term, and they have different optimization procedures. A large number of experimental results show a slight difference between them with the same kernel and the same graph. It is difficult to distinguish their pros and cons from an algorithm perspective.

- **SVM and RLS**

  The objective functions of supervised kernel machines take the same form:

  ℓ = supervised loss + kernel regularizer.

  Supervised loss for labeled data, the distances between the prediction and the true label. Kernel regularization, the complexity of the classification model. The only difference lies in the supervised loss.

  SVM

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{L} \max(1 - y_i f(x_i), 0) + \gamma_A \|f\|_A^2 \tag{1.7}$$

  RLS

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{L} (y_i - f(x_i))^2 + \gamma_A \|f\|_A^2 \tag{1.8}$$

  where γ_A controls the complexity of the function f in the ambient space.

- **Graph Regularization**

The manifold/graph regularization penalizes differences in the behavior of a classifier under slight changes along the manifold.

$$\begin{aligned}&\frac{1}{2}\|f\|_I^2\\ =&\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}[f(x_i)-f(x_j)]^2\\ =&F^T\mathbf{L}F\end{aligned} \tag{1.9}$$

where $\mathbf{L} = D - W$, and D is the degree matrix formulated as $d_i = \sum_{j=1}^{N} w_{ij}$, $D = \mathrm{diag}(d_1,\cdots,d_i,\cdots,d_N)$. $\mathbf{L}$ is called graph Laplacian.

LapSVM and LapRLS are the extensions of SVM and RLS respectively. There is an additional manifold regularization which is estimated by graph Laplacian as unsupervised loss.

ℓ = supervised loss + kernel regularizer + unsupervised loss (graph regularizer)

For notation simplicity, we ignore every normalization coefficient of each term here, γ_A controls the complexity of the function f in the ambient space and γ_I controls the complexity of the function f in the intrinsic geometry of $\mathcal{P}_X$.

- **LapSVM** taking classification model as $f(x) = \alpha^T\mathbf{k}(x) + b$.

$$\begin{aligned}&\min_{f\in\mathcal{H}_k}\sum_{i=1}^{L}\max(1-y_if(x_i),0)+\gamma_A\|f\|_A^2+\frac{\gamma_I}{2}\|f\|_I^2\\ =&\min_{f\in\mathcal{H}_k}\sum_{i=1}^{L}\max(1-y_i(\alpha^T\mathbf{k}(x_i)+b),0)+\gamma_A\alpha^TK\alpha+\gamma_I(\alpha^TK+1^Tb)\mathbf{L}(K\alpha+1b)\end{aligned} \tag{1.10}$$

- **LapRLS** taking classification model as $f(x) = \alpha^T\mathbf{k}(x)$.

$$\begin{aligned}&\min_{f\in\mathcal{H}_k}\sum_{i=1}^{L}(y_i-f(x_i))^2+\gamma_A\|f\|_A^2+\frac{\gamma_I}{2}\|f\|_I^2\\ =&\min_{f\in\mathcal{H}_k}\sum_{i=1}^{L}(y_i-\alpha^T\mathbf{k}(x_i))^2+\gamma_A\alpha^TK\alpha+\gamma_I\alpha^TK\mathbf{L}K\alpha\end{aligned} \tag{1.11}$$

1.3.3 Two-step modeling

The kernel is used twice in the Laplacian kernel machines, both for nonlinear classification and manifold approximation. Its quality has a direct influence on the performance of the classification model.

A nonlinear separation boundary in the input space is realized by a linear separation boundary in the feature space to realize. So the quality of the feature space is critical to the final performance of the classification model.

General nonlinear kernels such as RBF kernel and polynomial kernel in Eq.(1.5) implicitly map the data from the input space to the general feature space. Taking the RBF kernel as an example, the corresponding feature space as

$$\phi(x) = e^{-\gamma x^2} \cdot [1, x\sqrt{\frac{2\gamma}{1!}}, \sqrt{\frac{(2\gamma)^2}{2!}}x^2, \sqrt{\frac{(2\gamma)^3}{3!}}x^3, \cdots]^T \tag{1.12}$$

where the data x is mapped to an infinite dimensional space.

The general kernels are simple to apply and effective sometimes. However, we do not know if a general feature space is suitable for a specific task. In other words, data distribution, including labeled and unlabeled data in the semi-supervised context, cannot be used to design the feature space. From a modeling perspective, general kernels are black-box models with less interpretability, and prior knowledge cannot be used even if given.

Therefore, in this dissertation, we are motivated to model the separation boundary in an interpretable way by using a set of linear models to approximate the nonlinear separation boundary.

The formulation of the classification model is

$$f(x) = \sum_{j=1}^{M}(\Omega_j^T x + b_j) \cdot g_j(x) + b \tag{1.13}$$

where $\Omega_j^T x + b_j \quad j = 1, \cdots, M$ are a set of linear models, $g_j(x) \quad j = 1, \cdots, M$ are nonlinear functions that connect or combine these linear models. All the model parameters are estimated in two steps.

In the first step, nonlinear parameters $g_j(x) \quad j = 1, \cdots, M$ are estimated.

By introducing a regression vector $\phi(x)$ and a parameter vector Θ

$$\begin{aligned} \phi(x) &= [\mathbf{g}^T(x) \otimes [1, x^T]]^T \\ \Theta &= [b_1, \Omega_1^T, \cdots, b_M, \Omega_M^T]^T \end{aligned} \tag{1.14}$$

where the symbol $\otimes$ denotes Kronecker product and $\mathbf{g}(x) = [g_1(x), \cdots, g_M(x)]^T$. The classification model Eq.(1.13) is formulated into a linear regression form

$$f(x) = \Theta^T \phi(x) + b \tag{1.15}$$

where $\phi(x)$ is a known regression vector and Θ is a parameter vector containing all the linear parameters to estimate in the second step.

In the second step, all the linear parameters $\Omega_j, b_j, b \qquad j = 1, \cdots, M$ are estimated globally.

1.3.4 Quasi-linear kernel

Since $\phi(\cdot)$ defined in Eq.(1.14) is equivalent to a mapping function that maps the input to a finite high dimensional spanned feature space. Eq.(1.15) can be further recast as a kernel form

$$f(x) = \alpha^T \mathbf{k}(x) + b \tag{1.16}$$

where $\mathbf{k}(x) = [k(x, x_1), \cdots, k(x, x_L), k(x, x_{L+1}), \cdots, k(x, x_N)]^T$, and α is an N dimensional coefficient. The kernel is defined by inner product of the regression vectors as similarity between instances in the spanned feature space

$$\begin{aligned} k(x_i, x_j) &= \phi^T(x_i)\phi(x_j) \\ &= (1 + x_i^T x_j)\mathbf{g}^T(x_i)\mathbf{g}(x_j) \end{aligned} \tag{1.17}$$

which is called quasi-linear kernel.

As a result, different from general kernels, the quasi-linear kernel as Eq.(1.17) is composed in an interpretable way, and it contains the prior knowledge **g**. From the modeling perspective, the quasi-linear kernel model as Eq.(1.16) is the intermediate model of the two-step modeling method.

There are a series of researches about the two-step modeling method and the quasi-linear kernel. It was firstly used by Zhou et al. [60]. Li et al. proposed a supervised algorithm [61] and an unsupervised algorithm [62] to compose the quasi-linear kernel. Liang et al. used the two-step modeling method to tackle an imbalanced problem [63]. Zhu et al. used the two-step modeling method for a task with missing data [64]. As a task closer to ours, Zhou et al. proposed to train with labeled and unlabeled data using the two-step modeling method in a *transductive* manner, where the low-density and manifold assumptions were used [65].

In this dissertation, our task is to classify in a semi-supervised context, in which a small amount of labeled data and a large amount of unlabeled data are used for training.

From the aspect of two-step modeling method, we use both labeled data and unlabeled data to estimate nonlinear parameters in the first step and linear parameters in the second step. And evaluate the performance of the learned classification model on the testing set, which is never used in the training phase.

From the aspect of quasi-linear kernel, we compose the quasi-linear kernel in a semi-supervised manner. And with the quasi-linear kernel, we optimize the model parameters in a semi-supervised manner by using Laplacian kernel machines.

1.4 Goals of the Dissertation

This dissertation considers that human annotations are expensive and lack of label is a common and fatal problem in real-world classification tasks. We aim to design several semi-supervised classification algorithms to leverage unlabeled data, which is much easier to collect, in addition to labeled data for training.

The kernel is used twice in Laplacian kernel machines, both for the feature mapping and the data manifold approximation in the graph. The quality of the kernel has a direct influence on the performance of the classification model. General nonlinear kernels are black-box models with less interpretability, and prior information cannot be used even if given.

In this dissertation, we are motivated to apply a two-step modeling method to model the nonlinear separation boundary with a set of linear models, where a quasi-linear kernel model is formulated as an intermediate model. As a result, the quasi-linear kernel is composed in an interpretable way, and it contains prior knowledge.

Therefore, this dissertation proposes a series of Laplacian kernel machine based algorithms that support semi-supervised classification through the construction of an intermediate model called quasi-linear kernel.

1.4.1 Multi-local linear model

The whole training data can be divided into multiple local partitions, where each partition is almost linearly separable. The center and radius of partitions are information of potential separation boundary. A multi-local linear model interpolating the linear models in each partition can be formulated to take advantage of the information of potential separation boundary.

We aim to build a multi-local linear model in a semi-supervised context to approximate the nonlinear separation boundary, and we propose a LapSVM based semi-supervised classifier that uses the quasi-linear kernel as intermediate model (Chapter 2).

1.4.2 Piecewise linear model

Learning data manifold is one way to leverage unlabeled data to help classification. Laplacian kernel machines approximate the data manifold by a graph. And the data manifold can also be estimated by neural networks. In order to combine these two methods to learn and utilize the data manifold effectively to help classification, we introduce a gated linear network to realize a piecewise linear model.

We aim to build a piecewise linear model in a semi-supervised context to approximate the nonlinear separation boundary, and we propose a LapRLS based semi-supervised classifier that uses the quasi-linear kernel as intermediate model (Chapter 3).

1.4.3 Parasite image classification

For a classification task of multiple parasites in the microscopic image, previous work assumed all the data is labeled [66]. However, the lack of labels is a common and fatal challenge in practical clinical scenarios. We aim to classify the parasites with a small amount of labeled data and a large amount of unlabeled data.

We aim to apply the proposed semi-supervised classifier based on piecewise linear model to parasite images. We propose a semi-supervised classification algorithm containing a high-performance LapSVM classifier and a DNN feature extractor using contrastive learning (Chapter 4).

1.5 Challenges

In this dissertation, we propose a series of semi-supervised classification algorithms based on Laplacian kernel machines, which apply the two-step modeling method and construct an intermediate model named quasi-linear kernel.

The foremost challenge is the usage of labeled and unlabeled data. How to leverage the unlabeled data to make a positive influence on the learning process? And how to fully exploit the limited valuable labeled data? From a modeling perspective, challenges lie in both steps. How to connect or combine the linear models to approximate a nonlinear separation boundary? And how to optimize the linear models? From the aspect of Laplacian kernel machines, how to leverage prior knowledge and label information to build the kernel and the graph?

1.5.1 Usage of unlabeled & labeled data

For the multi-local linear model, detecting the potential separation boundary relied on labeled data [61]. However, in a semi-supervised context, only a small amount of labeled data is available, and we have a large amount of unlabeled data. It is challenging to leverage unlabeled data for detection.

For the piecewise linear model, an auto-encoder can be trained to realize partitioning indirectly for a piecewise linear model, in which the data manifold is estimated in an unsupervised manner [62]. However, in a semi-supervised context, we also have some valuable labeled data except for unlabeled data. How to take advantage of valuable labeled data to guide the manifold estimation is challenging.

1.5.2 Challenges of parasite images

For the parasite image classification, considering human annotations are very expensive in medical imaging, we assume only 1% parasite images are labeled. It is challenging to classify the parasite images with only 1% labeled data.

Furthermore, there are two challenges in microscopic parasite image classification. On the one hand, the semantic objects, namely salient structures, are fuzzier and more complex than real-world images, leading to microscopic images are not as distinguishable as macroscopic images. On the other hand, the insignificant textures, like image background staining, lightness, or contrast level, vary much in samples from different clinical scenarios. At the same time, the available training data is of a similar pattern in each category, which may lead to poor generalization in real-world applications.

1.6 Dissertation Outlines and Main Contributions

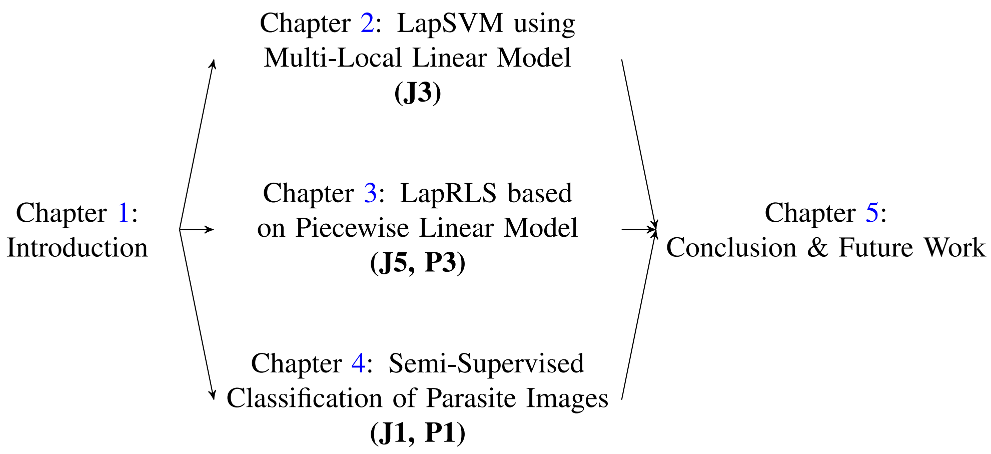
Chapter 1: Introduction
Chapter 2: LapSVM using Multi-Local Linear Model
(J3)
Chapter 3: LapRLS based on Piecewise Linear Model
(J5, P3)
Chapter 4: Semi-Supervised Classification of Parasite Images
(J1, P1)
Chapter 5: Conclusion & Future Work

FIGURE 1.7: Outline of this dissertation.

This dissertation shows the cumulative works over my doctoral period through five chapters.

Chapter 1 lays the fundamental background that connects the whole dissertation and shows an outline. In Chapter 2, we propose a Laplacian SVM based semi-supervised classifier using multi-local linear model. Chapter 3 proposes a semi-supervised classifier based on piecewise linear model using gated linear network. Chapter 4 proposes a semi-supervised classification algorithm of parasites, which contains a semi-supervised feature extractor trained by contrastive learning and a semi-supervised classifier optimized by LapSVM. Finally, in Chapter 5 we summarize our findings and applications and discuss the future study. The outline of this dissertation is depicted in Fig.1.7, in which the indexes represent the published paper in Publication List.

Chapter 2 proposes a Laplacian SVM based semi-supervised classifier using multi-local linear model. The semi-supervised classifier is constructed in two steps. In the first step, by applying a pseudo-labeling approach, the input space is divided into multiple local linearly separable partitions along the potential separation boundary. A multi-local linear model is then built by interpolating multiple local linear models assigned to the partitions. In the second step, the multi-local linear model is formulated as a linear regression form with a new regression vector containing the information of potential separation boundary. Then all the linear parameters are optimized globally by a LapSVM algorithm using a quasi-linear kernel function defined as the inner product of the new regression vectors. Furthermore, the quasi-linear kernel function and the pseudo labels are used to construct a label-guided graph.

The main contributions related to this chapter are shown as follows:

- This dissertation builds a multi-local linear model under a semi-supervised context.
- A pseudo-labeling approach is applied to leverage unlabeled data to detect the potential separation boundary;
- The linear parameters in each local partition are optimized globally by labeled and unlabeled data using LapSVM;
- The information of potential separation boundary is incorporated into the kernel function of LapSVM using an interpretable model, leading to higher theoretical interpretability and data adaptability.
- For the graph of LapSVM, the limited labels are fully used simply and effectively to fit the classification task better.

Chapter 3 proposes a semi-supervised classifier based on piecewise linear model using gated linear network. The semi-supervised classifier is constructed in two steps. In the first step, we design a label-guided autoencoder-based semi-supervised gating mechanism to generate binary sequences. By using a gated linear network, the binary sequences realize partitioning of a piecewise linear model indirectly. In the second step, the piecewise linear model is formulated as a linear regression form, and the linear parameters are then optimized globally by a LapRLS algorithm using a quasi-linear kernel function comprising the binary sequences. Moreover, the quasi-linear kernel function is used as a better similarity function for the graph construction.

The main contributions related to this chapter are shown as follows:

- This dissertation realizes a piecewise linear model to approximate the nonlinear separation boundary using the gated linear network in a semi-supervised context.

- A label-guided autoencoder is designed to leverage labeled data to guide the data manifold estimation.
- The estimated data manifold is incorporated into both the kernel and the graph in LapRLS;
- Data manifold approximation by graph and data manifold estimation by neural network are realized both;

Chapter 4 proposes a semi-supervised classification algorithm of microscopic parasite images, which contains a semi-supervised feature extractor trained by contrastive learning and a semi-supervised classifier optimized by LapSVM. First, for the deep CNN feature extractor, we introduce real-world images with similar and clear semantic information to enhance the structure at the representation level. In addition, we introduce variant appearance transformations to eliminate the texture at the representation level. Second, a gated linear network is adopted as the classifier to realize a piecewise linear separation boundary. The linear parameters are optimized globally by a LapSVM algorithm using a qausi-linear kernel function composed of the representations and the binary sequences generated from the learned feature extractor.

The main contributions related to this chapter are shown as follows:

- The proposed method shows excellent performance when only 1% data is labeled. Affordable and accurate parasite classification is achieved;
- Contrastive learning is applied on labeled data by connecting two domains to enhance the structure at the representation level;
- Contrastive learning is applied on unlabeled data by encouraging consistency of the same input under different appearance transformations to eliminate the texture at the representation level;
- A new dataset of the real-world images is built by photoing, which best matches the microscopic parasite images of interest.

Chapter 5 summarizes and concludes the dissertation and provides several suggestions for future researches. In conclusion, this dissertation proposes a series of semi-supervised classification algorithms based on Laplacian kernel machines through the construction of an intermediate model named quasi-linear kernel. Compared to Laplacian kernel machines with general nonlinear kernel, this dissertation detects prior knowledge from labeled and unlabeled data and incorporates the prior into the kernel and graph. Compared to previous studies about quasi-linear kernel in a supervised or unsupervised manner, this dissertation considers learning under limited supervision and models the nonlinear separation boundary by leveraging a small amount of labeled data and a large amount of unlabeled data. As a result, accurate performance is achieved under a semi-supervised context.

Chapter 2

LapSVM Using Multi-Local Linear Model

2.1 Background

[1]Traditionally, classifier learning has been studied in the supervised schemes where all the data points are labeled. Labeled data however is often difficult or time consuming to prepare in contrast to unlabeled data. Therefore, semi-supervised classification (SSC) has gained prominence, which aims to understand how introducing unlabeled data in the process may change the learning behavior, and design algorithms that take advantage of such an introduction. SSC is typically with a small amount of labeled data and a great deal of unlabeled data, so it is of immense practical interest in a wide range of applications, such as biology, image processing, natural language processing where labeled data is expensive while abundant unlabeled data is available [10, 67, 68].

An important prerequisite of SSC is that the underlying marginal distribution over the input space contains information about the posterior distribution. Different ways of

[1]This chapter is mainly extended from the Journal paper: Y. Ren, H. Zhu, Y. Tian and J. Hu, "A Laplacian SVM based Semi-Supervised Classification using Multi-Local Linear Model", *IEEJ Transactions on Electrical and Electronic Engineering*, Vol.16, No.3, pp.455-463, March, 2021.

interaction of marginal distribution and posterior distribution lead to different assumptions, such as smoothness assumption, low-density assumption and manifold assumption [3].

The manifold assumption states that the high-dimensional data points lie (roughly) on low-dimensional manifolds, and the data points lying on the same low-dimensional manifold have the same label. Neighborhood graph is one way to model the manifold smoothness. In the graph based SSC, each node of the graph corresponds to a labeled or unlabeled data point, and the edge weights encode the similarity between each pair of data points. Smoothness on manifold is encouraged by data points connected by large weights are given similar labels.

Graph based SSC algorithms, such as the graph mincuts [69], the Gaussian fields and harmonic functions [70] and the local and global consistency [71], are transductive in nature, Laplacian support vector machine (LapSVM) algorithm [51, 59] has been proposed as a typical inductive algorithm. It extends the support vector machine, a supervised kernel machine with marginal maximum, by incorporating a manifold regularizer which is estimated using the graph Laplacian associated to all the training data.

From the aspect of kernel machine, a kernel function implicitly maps the input to a feature space. However, most commonly used nonlinear kernels, like RBF (radial basis function) kernel, are black-box models with less interpretability. The choice of kernel is quite data dependent, RBF kernel may fail in some specific cases since it does not use any information from the data. From the aspect of graph based SSC algorithm, the graph construction is the key. The commonly used graph construction methods, like kNN graph, are unsupervised in nature, the labels available at our disposal are ignored.

In this chapter, following the manifold assumption, we assume that the data is distributed on low-dimensional manifolds, and the separation boundary is roughly along the data manifolds. For the nonlinear separable binary classification, the whole training data can be divided into multiple local partitions, where each partition is almost linear separable. Under this premise, the centers and the widths of the partitions can be seen as the information of potential separation boundary. A multi-local linear model interpolating the linear models in each partition is formulated to approximate the nonlinear separation boundary, where the interpolation function is determined by the centers and the widths (Section 2.2). After the centers and the widths being estimated by a pseudo-labeling method (Section 2.3), the multi-local linear model is formulated into a linear regression form, where the new input vector in the spanned space contains the centers and the widths, while the linear parameters in all the partitions are gathered in the linear vector. Then the linear parameters can be estimated globally by leveraging a powerful tool LapSVM, whose kernel is defined as the inner product of input in the spanned space, namely, a quasi-linear kernel. Furthermore, both the quasi-linear kernel and the pseudo labels are used to construct a label guided graph (Section 2.4). In this way, the detected information of potential separation boundary is ingeniously incorporated into LapSVM in both kernel level and graph level.

The contribution of this chapter has two folds:

- For the kernel, the information of potential separation boundary is detected, and then incorporated into the kernel function by using an interpretable model, which leads to higher theoretical interpretability and data adaptability.
- For the graph, the limited labels are fully used in a simple and effective way to better fit the classification task.

The rest of this chapter is organized as follows. Section 2.2 formulates a unified framework for the multi-local linear model which interpolates multiple local linear models in the linear separable partitions, the parameters are going to be estimated in two steps. In Section 2.3, the nonlinear parameters, the centers and the widths of linear separable partitions, are estimated as the first step. Section 2.4 shows the estimation of linear parameters using LapSVM as the second step. Section 2.5 presents the simulation results of a series of experiments to validate the effectiveness of our proposed method. Finally, the conclusions are drawn in Section 2.6.

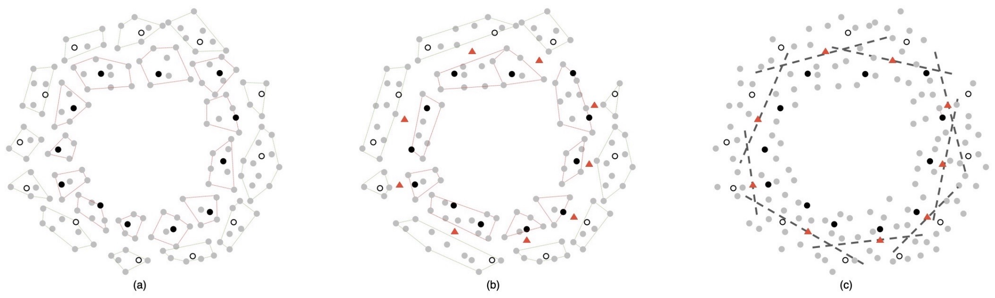

FIGURE 2.1: Framework of the proposed method.

2.2 Multi-Local Linear Model

In the case of a semi-supervised binary classification problem, there is a probability distribution $\mathcal{P}$ on $X \times \mathbb{R}^d$ and a prediction function $f(x)$ is learned from data sampled according to $\mathcal{P}$, labeled data (x, y) is sampled from the joint distribution $\mathcal{P}_{XY}$ while unlabeled data $x \in X$ is sampled from the marginal distribution $\mathcal{P}_X$. The training set $\mathcal{S} = \mathcal{L} \cup \mathcal{U}$ contains N data points, where L labeled data $\mathcal{L} = \{(x_l, y_l)\}_{l=1}^{L}$ with $y_l \in \{1, -1\}$ and U unlabeled data $\mathcal{U} = \{x_u\}_{u=L+1}^{N}$, and $X_N = X_L \cup X_U$.

A multi-local linear model [60] consisting of multiple local linear models with interpolations is considered to approximate the nonlinear separation boundary:

$$f(x) = \sum_{j=1}^{M} (\Omega_j^T x + b_j) \cdot g_j(x) + b \tag{2.1}$$

where $\Omega_j^T x + b_j (j = 1, \cdots, M)$ is a set of local linear models, b is a bias parameter and $g_j(x)(j = 1, \cdots, M)$ are the interpolation functions, described by

$$g_j(x) = \frac{\tilde{g}_j(x)}{\sum_{j=1}^{M} \tilde{g}_j(x)}, \quad \tilde{g}_j(x) = e^{-\frac{(x-\mu_j)^2}{\lambda \sigma_j^2}} \tag{2.2}$$

where μ_j and σ_j are the centers and the widths of the partitions, and each partition is assumed to be linearly separable, and λ is a scale parameter. The nonlinear classifier is given by

$$y_p = \text{sign}(f(x)) \tag{2.3}$$

The key problem is how to estimate the parameter set $[\Omega_j, b_j, b, \mu_j, \sigma_j \quad j = 1, \cdots, M]$ using both labeled and unlabeled data.

We develop a two-step algorithm. In the first step, the nonlinear parameters, the centers and the widths $\mu_j, \sigma_j \quad j = 1, \cdots, M$, are estimated by pseudo-labeling, partitioning and overlap removing. In the second step, the known variables and unknown linear parameters can be safely separated by introducing $\phi(x)$ and Θ:

$$\begin{aligned}\phi(x) &= [g_1(x), x^T g_1(x), \cdots, g_M(x), x^T g_M(x)]^T \\ &= [\mathbf{g}^T(x) \otimes [1, x^T]]^T\end{aligned} \tag{2.4}$$

$$\Theta = [b_1, \Omega_1^T, \cdots, b_M, \Omega_M^T]^T \tag{2.5}$$

where the symbol $\otimes$ denotes Kronecker product and $\mathbf{g}(x) = [g_1(x), \cdots, g_M(x)]^T$ contains the information of potential separation boundary. Eq.(2.1) is expressed as a linear regression form

$$f(x) = \phi^T(x)\Theta + b \tag{2.6}$$

It is equivalent to the fact that $\phi(x)$ defined as Eq.(2.4) maps the input to a finite high dimensional spanned feature space. Then Eq.(2.6) is recasted as a kernel machine

$$f(x) = \mathbf{k}^T(x)\alpha + b \tag{2.7}$$

where $\mathbf{k}(x)$ is the similarity vector of N training data and x in the spanned space, $\mathbf{k}(x) = \Phi_N\phi(x) = [k(x,x_1), \cdots, k(x,x_N)]^T$, $k(x_i,x_j) = \phi^T(x_i)\phi(x_j)$, $\Phi_N = [\phi(x_1), \cdots, \phi(x_L), \cdots, \phi(x_N)]^T$, and α is an N dimensional coefficient, $\Theta = \Phi_N^T\alpha$. The linear parameters Ω_j, b_j of each partition and the bias b are estimated implicitly by applying LapSVM. A visual representation of the proposed method is provided in Fig.2.1.

2.3 Estimation of Nonlinear Parameter

2.3.1 Pseudo-labeling

Considering the cases where there are overlaps between clusters and the data is not easily separated, we use a seeded and constrained clustering method based on k-mean method [72] to assign pseudo labels to the unlabeled data.

The labeled data is clustered firstly for seeding if there are a large number of labeled data points. In this chapter, we focus on the scenario that there are a small number of labeled data points as a typical SSC problem. When with less labeled data, each labeled data point is as a seed, and the labeled data points are as the initial centers $\{c_1, \cdots, c_L\}$ of the clusters $\{C_1, \cdots, C_L\}$.

The object is to minimize J:

$$J = \sum_{i=1}^{L} \sum_{x \in C_i} ||x - c_i||^2 \tag{2.8}$$

When doing cluster assignment, the labeled data points as seeds always remain unchanged in the clusters, for each unlabeled data point $x_{uj} \quad j = 1, \cdots, U$, it is assigned to the cluster C_i by $\arg\min_i ||x_{uj} - c_i||^2$. Then update the centers $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$. The process of assigning all the unlabeled data to L clusters is repeated until the centers converge.

Following the cluster assumption, it is a pseudo-labeling step, every unlabeled data point ends up with a pseudo label as illustrated in Fig.2.1(a). The cluster with positive labeled data is as positive cluster X^+ while the cluster with negative labeled data is as negative cluster X^-. In the context of binary classification with balanced labels, we end up with L clusters, where one half are positive clusters and the other half are negative clusters.

2.3.2 Partitioning

Given the pseudo labels which roughly detect the potential separation boundary, the partitioning can be performed along the potential separation boundary, and each partition is almost linearly separable.

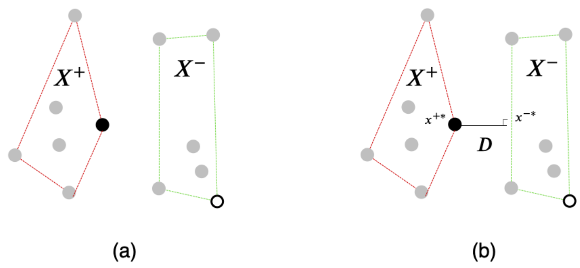

FIGURE 2.2: Pair and the shortest distance.

Step 1: pairing

For each cluster, its nearest cluster with opposite label is chosen to form a pair. We end up with M_0 different pairs, where $\lceil \frac{L}{2} \rceil \leq M_0 < L$, and each pair consists of a positive cluster and a negative cluster. Fig.2.2(a) shows an example of a pair $P = \{X^+, X^-\}$, each cluster is represented by a convex hull. Defined as [73] ,the convex hull of a finite point set is the set of all convex combinations of its points.

Step 2: merging

- Suppose $P_t = \{X_t^+, X_t^-\}$ and $P_k = \{X_k^+, X_k^-\}$ are merged, $P = P_t \cup P_k$

$$\begin{aligned} CH(X^+) &= CH(X_t^+ \cup X_k^+) \\ CH(X^-) &= CH(X_t^- \cup X_k^-) \end{aligned} \tag{2.9}$$

- As illustrated in Fig.2.2(b), by solving

$$\begin{aligned} &\min ||x^+ - x^-|| \\ &\text{s.t.} \quad x^+ \in CH(X^+), x^- \in CH(X^-) \end{aligned} \tag{2.10}$$

SK algorithm [74] always converges to a nearest point pair (x^{+*}, x^{-*}) between $CH(X^{+})$ and $CH(X^{-})$. Given a linear model

$$\begin{gathered} w^T x + b \\ w = x^{+*} - x^{-*}, \quad b = \frac{\|x^{-*}\|^2 - \|x^{+*}\|^2}{2} \end{gathered} \tag{2.11}$$

D is defined as the shortest distance between $CH(X^{+})$ and $CH(X^{-})$. If $\exists x^{+} \in X^{+} \quad wx^{+} + b \leq 0$ or $\exists x^{-} \in X^{-} \quad wx^{-} + b \geq 0$, P is not linearly separable and $D = -1$. Otherwise, P is linearly separable and $D = \|x^{+*} - x^{-*}\|$.

- If P is not linearly separable, it is as one local linear partition $\mathbf{P}$. When P is linearly separable, if it merges with its 3 nearest pairs P_{NN} respectively and all the merged ones are not linearly separable, P is also as one local linear partition $\mathbf{P}$.

Randomly choose a pair P_t and estimate whether it is a local linear partition. If it is a local linear partition, randomly choose another pair to do the estimation. Otherwise, do the merge process, choose P_k from the 3 nearest pairs of P_t by $\arg\max_k(D(P_t \cup P_k))$, $P_t = P_t \cup P_k$. Then do the estimation of P_t again. The algorithm ends when all the M_0 pairs have been estimated or merged. Finally, we end up with M local linear partitions, where $M \leq M_0$. Note that the number of partitions M in the proposed method is determined automatically, instead of setting a fixed value in advance.

2.3.3 Estimating the centers and the widths

The data near the potential separation boundary can be seen as overlapping. We remove the overlapping to get centers and widths with higher confidence.

The data near the potential boundary is deleted to make sure $D \geq \epsilon$ for all the partitions. ϵ is empirically set to be $D_{\max}$ which is the maximum D among all the partitions in our simulation.

For each partition, if $\mathbf{P}_i$ is not linearly separable ($D_i < 0$), delete the data point x by $\arg\min_{\vec{u}} proj_{\vec{v}}(\vec{u})$ iteratively until $D_i > 0$, where $proj_{\vec{v}}(\vec{u}) = |\frac{\vec{u}\cdot\vec{v}}{\|\vec{v}\|^2}\vec{v}|$, $\vec{u} = x - \frac{1}{2}(\bar{x}^+ + \bar{x}^-)$ and $\vec{v} = \bar{x}^+ - \bar{x}^-$, $\bar{x}^+$ and $\bar{x}^-$ are the mean value of the data in the cluster X^+ and X^-. When $\mathbf{P}_i$ is linearly separable ($D_i > 0$), delete the data point x by $\arg\min_{\vec{u}} proj_{\vec{v}}(\vec{u})$ iteratively until $D_i \geq \epsilon$, where $\vec{u} = x - \frac{1}{2}(x^{+*} + x^{-*})$ and $\vec{v} = x^{+*} - x^{-*}$.

Finally, the center of $\mathbf{P}_i$ is given by

$$\mu_i = \frac{1}{2}(x^{+*} + x^{-*}) \quad i = 1, \cdots, M \tag{2.12}$$

as illustrated in Fig.2.1(b), and the width σ_i is the radius defined as the largest distance between the center μ_i and the data in the local partition.

2.4 Estimation of Linear Parameter by LapSVM

2.4.1 Kernel trick

Given the centers and the widths, by introducing $\phi(x)$ and Θ as Eq.(2.4)Eq.(2.5), the multi-local linear model is expressed as a linear regression form as Eq.(2.6), then it is recasted as a kernel machine as Eq.(2.7). $k(x_i, x_j)$ is a data-dependent kernel called quasi-linear kernel [60]

$$\begin{aligned} k(x_i, x_j) &= \phi^T(x_i)\phi(x_j) \\ &= (1 + x_i^T x_j)\mathbf{g}^T(x_i)\mathbf{g}(x_j) \end{aligned} \tag{2.13}$$

The estimation of Θ is converted to the estimation of α to take advantage of the kernel trick, in this way, there is no need to explicitly calculate the input in the spanned feature space $\phi(x)$, instead, the kernel is given by input x and interpolation function $\mathbf{g}(x)$. As a result, the information of separation boundary is incorporated into the kernel.

2.4.2 Objective function

In the context of SSC, Laplacian support vector machine (LapSVM) formulated by

$$\min_{f\in\mathcal{H}_k} \sum_{i=1}^{L} \max(1-y_i f(x_i),0) + \frac{\gamma_A}{2}\|f\|_A^2 + \frac{\gamma_I}{2}\|f\|_I^2 \tag{2.14}$$

is considered to be a good choice to solve Eq.(2.7). γ_A controls the complexity of the function in the ambient space and γ_I controls the complexity of the function in the intrinsic geometry of $\mathcal{P}_X$. By using the graph Laplacian regularizer $F^T LF$, where $F = K\alpha + 1b$, we have

$$\begin{aligned}&\min_{\alpha,b}\frac{1}{2}\sum_{i=1}^{L}\max(1-y_i(\mathbf{k}^T(x_i)\alpha+b),0)^2\\ &+\frac{\gamma_A}{2}\alpha^T K\alpha+\frac{\gamma_I}{2}(\alpha^T K+1^T b)\mathbf{L}(K\alpha+1b)\end{aligned} \tag{2.15}$$

where $K = \Phi_N\Phi_N^T = [\mathbf{k}(x_1),\cdots,\mathbf{k}(x_N)]^T$, $\mathbf{L}$ is the graph Laplacian built from X_N, and 1 is the vector whose all entries equal to 1.

2.4.3 Training in the primal

The LapSVM described by Eq.(2.15) can be trained both in the primal and in the dual. For small scale datasets, it can be trained in the dual, referred to Ref. [65] for the detail algorithm, while for large scale or high dimensional datasets, it can be trained in primal with lower computational complexity and shorter training time. We briefly describe the algorithm of training in the primal.

The problem of Eq.(2.15) is piecewise quadratic, and gradient descent is a natural choice for an efficient minimization, indicating the vector $z = \begin{bmatrix} b \\ \alpha \end{bmatrix}$:

$$z^t = z^{t-1} - s\nabla \tag{2.16}$$

The gradient ∇ and the Hessian H of Eq.(2.15) with respect to z can be calculated as:

$$\begin{aligned}\nabla &= \begin{bmatrix} \nabla_b \\ \nabla_\alpha \end{bmatrix} \\ &= \begin{bmatrix} 1^T I_{\mathcal{E}}(K\alpha + 1b) - 1^T I_{\mathcal{E}} y + \gamma_I 1^T \mathbf{L}(K\alpha + 1b) \\ K I_{\mathcal{E}}(K\alpha + 1b) - K I_{\mathcal{E}} y + \gamma_A K\alpha + \gamma_I K\mathbf{L}(K\alpha + 1b) \end{bmatrix}\end{aligned} \tag{2.17}$$

$$\begin{aligned}H &= \begin{bmatrix} \nabla_b^2 & \nabla_\alpha(\nabla_b) \\ \nabla_b(\nabla_\alpha) & \nabla_\alpha^2 \end{bmatrix} \\ &= \begin{bmatrix} 1^T I_{\mathcal{E}} 1 + \gamma_I 1^T \mathbf{L} 1 & 1^T I_{\mathcal{E}} K + \gamma_I 1^T \mathbf{L} K \\ K I_{\mathcal{E}} 1 + \gamma_I K \mathbf{L} 1 & K I_{\mathcal{E}} K + \gamma_A K + \gamma_I K \mathbf{L} K \end{bmatrix}\end{aligned} \tag{2.18}$$

where the error vector $\mathcal{E}$ is the subset of labeled data with the points that generate a hinge loss value greater than zero, the matrix $I_{\mathcal{E}} \in \mathbb{R}^{N,N}$ is a diagonal matrix where the only entries different from 0 (equal to 1) along the main diagonal are in positions corresponding to points that belong to $\mathcal{E}$, $y \in \mathbb{R}^N$ is an augmented version of label vector where y_i for labeled data while 0 for unlabeled data.

Combining Eq.(2.17) and Eq.(2.18) one can write $\nabla = Hz - c$, and the vector z for which $\nabla = 0$ can be computed by solving the system $Hz = c$

$$\begin{aligned}&\begin{bmatrix} 1^T I_{\mathcal{E}} 1 + \gamma_I 1^T \mathbf{L} 1 & 1^T I_{\mathcal{E}} K + \gamma_I 1^T \mathbf{L} K \\ K I_{\mathcal{E}} 1 + \gamma_I K \mathbf{L} 1 & K I_{\mathcal{E}} K + \gamma_A K + \gamma_I K \mathbf{L} K \end{bmatrix} z = \\ &\begin{bmatrix} 1^T I_{\mathcal{E}} y \\ K I_{\mathcal{E}} y \end{bmatrix}\end{aligned} \tag{2.19}$$

Preconditioned Conjugate Gradient (PCG) is used for efficient optimization by avoiding matrix inversions. Given a preconditioner P, the algorithm indirectly solves the system $Hz = c$ by solving $\hat{H}z = \hat{c}$, where $\hat{H} = P^{-1}H$ and $\hat{c} = P^{-1}c$. In particular, we can factorize

Eq.(2.19) as

$$\begin{bmatrix} 1 & 0^T \\ 0 & K \end{bmatrix} \begin{bmatrix} 1^T I_{\mathcal{E}} 1 + \gamma_I 1^T \mathbf{L} 1 & 1^T I_{\mathcal{E}} K + \gamma_I 1^T \mathbf{L} K \\ I_{\mathcal{E}} 1 + \gamma_I \mathbf{L} 1 & I_{\mathcal{E}} K + \gamma_A I + \gamma_I \mathbf{L} K \end{bmatrix} z = \begin{bmatrix} 1 & 0^T \\ 0 & K \end{bmatrix} \begin{bmatrix} 1^T I_{\mathcal{E}} y \\ I_{\mathcal{E}} y \end{bmatrix} \tag{2.20}$$

and select a preconditioner $P = \begin{bmatrix} 1 & 0^T \\ 0 & K \end{bmatrix}$.

In this way:

$$\begin{aligned} \hat{\nabla} &= \hat{H} z - \hat{c} \\ &= \begin{bmatrix} 1^T I_{\mathcal{E}} 1 + \gamma_I 1^T \mathbf{L} 1 & 1^T I_{\mathcal{E}} K + \gamma_I 1^T \mathbf{L} K \\ I_{\mathcal{E}} 1 + \gamma_I \mathbf{L} 1 & I_{\mathcal{E}} K + \gamma_A I + \gamma_I \mathbf{L} K \end{bmatrix} z \\ &- \begin{bmatrix} 1^T I_{\mathcal{E}} y \\ I_{\mathcal{E}} y \end{bmatrix} \end{aligned} \tag{2.21}$$

$$\nabla = Hz - c = P\hat{H}z - P\hat{c} = P\hat{\nabla} \tag{2.22}$$

More details refer to Ref. [59].

2.4.4 Graph construction

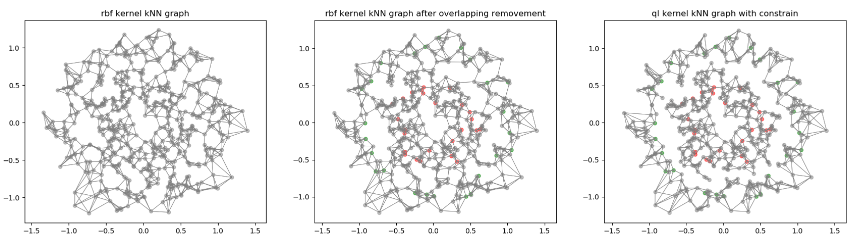

FIGURE 2.3: Graph topology of different methods.

Graph Laplacian is a matrix representation of a graph. Given an undirected graph with N nodes, its graph Laplacian is defined as:

$$\mathbf{L} = D - W \tag{2.23}$$

where D is the degree matrix formulated as $d_i = \sum_{j=1}^{N} w_{ij}$, $D = \text{diag}(d_1, \cdots, d_i, \cdots, d_N)$, and the adjacency matrix W can be decomposed into graph topology E and edge weighting k

$$W(x_i, x_j) = E_{ij}k(x_i, x_j) \tag{2.24}$$

If there is an edge between the nodes x_i and x_j, $E_{ij} = 1$, else, $E_{ij} = 0$, and there are no self-loops, $E_{ii} = 0, \forall i = 1, \cdots, N$.

1) A label-guided graph

Most of the commonly used graph construction methods like kNN graph and ϵ graph are unsupervised in nature, a certain number of labels available at our disposal are ignored.

In contrast with the unsupervised methods, there are several schemes using the available labels to optimize the graph structure to better fit the classification task.

To name a few, IDML [75] applies Mahalanobis based distance learning in the Gaussian edge weighting. The metric learning and label estimation with Gaussian Random Fields (RGF) objective are performed alternatively, until the method converges. A currently unlabeled instance is considered a new labeled training instance for next round of metric learning if its label distribution has low entropy. Under the using of a densely sampled manifold, a larger neighborhood (say k'NN) [76] in the ambient space can be searched to find the optimal k neighborhood on the manifold. Therefore, all the subgraphs of k'NN graph are as the hypothesis space.An algorithm to optimize the smoothness functional with respect to the neighborhood graph in the proposed hypothesis space is provided. GBILI [77] employs information conveyed by mutual k nearest neighbors and labeled data, and all data seek to connect to the closest label data. SSLRR [78] integrate the given label information into the state-of-the-art self-representation methods, such as the LRR graph, by restricting the representation coefficients between labeled points from different classes to be zero. Intuitively, this information helps us prevent the structure sparsity of the coefficients from being destroyed in challenging real world scenarios, i.e., small signal-to-noise ratio, dependent subspaces and/or nonlinear manifolds.

Instead of building a new model, we propose a simple method by making full use of the detected information of potential separation boundary to build a label guided graph. The circles data is used to illustrate the effectiveness of the proposed method intuitively in Fig.2.3.

Ahead of all, the graph construction is benefit from the unlabeled overlapping removement in the first step of the proposed method, as illustrated in Fig.2.3(b). Since some points connecting different data classes, which can be called bridge points, are deleted.

- edge weighting k:

  The quasi-linear kernel is used as edge weighting, since the information of potential separation boundary has been ingeniously incorporated into the quasi-linear kernel.

  The quasi-linear kernel leads to the fact that distance calculation is insensitive in the direction which is perpendicular to the local linear separation boundary. Therefore, the use of quasi-linear kernel as similarity measurement is a special case of data adaptive feature selection.

- graph topology E:

  We firstly build a kNN graph topology by using the quasi-linear kernel as similarity measurement. In addition, the edge between two data points is cut off if they have different pseudo labels.

  In this way, the graph topology is built by fusing the information from both the original space and spanned feature space in a label guided way. The intuitive effectiveness of proposed graph construction method is illustrated in Fig.2.3(c).

2) In formulation

Define a set $\Lambda = supp_h\{k_i\}$ containing elements with maximum h values in $k_i = [k_{i1}, \cdots, k_{ij}, \cdots, k_{iN}]^T$, where $K_{NN} = [k_1, \cdots, k_i, \cdots, k_N]^T$:

$$\underset{1 \le i,j \le N}{E_{ij}} = \begin{cases} 1, & i \neq j \text{ and } j \in \Lambda = supp_h\{k_i\} \text{ and } C_i = C_j \\ 0, & otherwise \end{cases} \tag{2.25}$$

where C_i is the pseudo label of x_i. If there is disconnected data after cutting off, we build an edge between it and its nearest neighbor with the same pseudo label.

Final E is symmetric since the graph is assumed to be undirected:

$$\underset{1 \le i,j \le N}{E_{ij}} = \begin{cases} 1, & E_{ij} + E_{ji} > 0 \\ 0, & otherwise \end{cases} \tag{2.26}$$

So the adjacency matrix W is given by:

$$\begin{aligned} W(x_i, x_j) &= E_{ij} k(x_i, x_j) \\ &= E_{ij}[(1 + x_i^T x_j)\mathbf{g}^T(x_i)\mathbf{g}(x_j)] \end{aligned} \tag{2.27}$$

Moreover, a normalized Laplacian [79] $\tilde{\mathbf{L}} = D^{-\frac{1}{2}} \mathbf{L} D^{-\frac{1}{2}}$ is used.

2.5 Numerical Experiments

2.5.1 Experiment setting

In this section, the proposed method is evaluated on a range of small and medium scale data sets including four semi-supervised benchmark data sets [67] and four UCI data sets [80], with vary size and dimensionality.

Data sets

Table.2.1 briefly summarizes the characteristics of the data sets. The split of data sets

TABLE 2.1: Data sets

dataset	dim	totoal	positive	negative
g241c	241	1500	750	750
digit1	241	1500	766	734
usps	241	1500	1200	300
text	11960	1500	750	750
ionosphere	34	351	225	126
hill valley noise	100	1212	614	598
splice	240	3190	1535	1655
madelon	500	2600	1300	1300

for training and testing is retained if they have a predefined split; otherwise, the data is randomly divide into a 80% training set and a 20% testing set. For each data set, some instances are randomly selected to be served as labeled data, and the remaining instances are served as the unlabeled data. Both the dividing and the selecting use stratified sampling to guarantee a similar input and label distribution.

TABLE 2.2: parameter settings

parameter	description	value
λ	scale parameter of proposed model	{ 0.01, 0.1, 1, 10, 100 }
γ	width of RBF (Gaussian) kernel	{ 0.01, 0.1, 1, 10, 100 }
γ_A	complexity of the function in ambient space	{0.01, 0.1, 1, 10, 100}
γ_I	complexity of the intrinsic geometry of $\mathcal{P}_X$	{0.01, 0.1, 1, 10, 100}
M	number of mixture components in HM	{50, 100, 150, 200, 250}

Evaluation metrics

As for the evaluation metrics, we use *Accuracy* and *F-score*, defined by

$$\begin{aligned} F\text{-}score &= \frac{2 \times Precision \times Recall}{Precision + Recall} \\ Accuracy &= \frac{TP + TN}{TP + FP + TN + FN} \end{aligned} \tag{2.28}$$

TABLE 2.3: Test results (%) of supervised and semi-supervised algorithms with 10% labeled data

	RLS		SVM		LapRLS		LapSVM		HM		proposed	
with 10% labeled data	F	A	F	A	F	A	F	A	F	A	F	A
g241c	80.00	79.67	80.62	79.00	80.26	80.00	79.61	79.33	78.46	78.11	80.63	80.33
digit1	92.91	93.33	93.47	93.67	94.29	94.67	93.66	94.00	92.99	92.99	93.67	93.92
usps	78.99	91.67	73.58	90.67	58.14	88.00	76.69	89.67	73.55	86.35	81.75	91.67
text	67.63	70.00	67.59	69.00	68.94	69.67	68.94	69.67	69.99	69.30	70.48	71.00
ionosphere	78.71	76.87	78.63	76.19	81.17	76.06	81.69	76.11	80.11	74.65	82.47	76.26
hill valley noise	88.16	88.92	88.57	88.12	88.95	88.91	89.67	89.93	91.22	91.44	91.42	91.42
splice	85.19	86.21	85.63	84.64	91.39	90.91	91.34	90.75	88.87	89.21	91.56	91.22
madelon	53.01	54.50	53.54	54.77	53.33	54.33	53.42	54.80	53.40	54.61	53.77	55.00
LapRLS win/tie/loss aginst others	4/3/1		5/1/2		-		-		-		-	
LapSVM win/tie/loss aginst others	5/1/2		6/2/0		4/1/3		-		-		-	
HM win/tie/loss aginst others	4/2/2		4/1/3		3/1/4		1/1/6		-		-	
proposed win/tie/loss aginst others	8/0/0		8/0/0		7/0/1		7/1/0		8/0/0		-	

TABLE 2.4: Test results (%) with various proportions of labeled data

		SVM					LapSVM					proposed					win/tie/loss		
		2%	5%	10%	20%	30%	2%	5%	10%	20%	30%	2%	5%	10%	20%	30%	LapSVM &SVM	proposed &SVM	proposed &LapSVM
g241c	F	67.09	74.67	80.62	83.39	83.97	66.89	73.33	79.61	84.31	84.87	67.09	71.29	80.63	84.41	84.59	2/2/1	4/1/0	2/3/0
	A	65.33	68.33	79.00	83.67	83.33	66.67	65.33	79.33	84.00	84.67	66.87	71.00	80.33	84.00	84.67			
digit1	F	70.48	88.06	93.47	94.81	95.14	85.82	88.73	93.66	96.19	96.17	87.05	88.00	93.67	96.31	96.41	5/0/0	5/0/0	1/4/0
	A	77.67	89.33	93.67	95.00	95.33	86.67	89.33	94.00	96.33	96.33	88.00	89.76	93.92	95.67	95.67			
usps	F	42.55	43.59	73.58	70.00	82.64	51.43	55.56	76.69	71.95	88.00	51.95	56.38	81.75	71.19	85.50	5/0/0	5/0/0	3/0/2
	A	64.00	85.33	90.67	90.00	93.00	71.67	76.00	89.67	84.67	95.00	75.33	72.67	91.67	88.67	86.33			
text	F	61.28	65.97	67.63	69.45	73.40	61.79	65.49	68.94	72.30	73.54	61.84	69.48	70.48	72.96	75.45	3/2/0	5/0/0	2/3/0
	A	61.67	67.33	70.00	71.67	73.67	61.67	67.33	69.67	72.67	74.33	61.33	68.67	71.00	72.33	72.67			
ionosphere	F	78.57	78.63	78.63	81.80	84.00	66.67	78.26	81.69	84.71	85.22	78.63	81.63	82.47	84.44	86.32	3/0/2	5/0/0	4/0/1
	A	66.20	76.02	76.19	79.92	80.61	50.70	76.83	76.11	81.69	81.46	64.79	76.65	76.26	80.28	81.69			
hill valley noise	F	67.10	69.70	88.57	91.92	97.61	64.79	71.99	89.67	91.07	97.75	69.20	71.60	91.42	91.48	97.89	3/2/0	4/1/0	3/2/0
	A	66.67	67.11	88.12	91.09	97.52	68.98	68.32	89.93	91.17	97.96	71.95	68.37	91.42	92.08	97.86			
splice	F	71.81	79.38	85.63	88.13	88.12	80.18	85.45	91.34	91.94	93.37	80.31	87.33	91.56	91.99	92.21	5/0/0	5/0/0	4/1/0
	A	75.39	75.08	84.64	88.56	88.72	79.15	84.95	90.75	91.38	92.01	80.25	86.99	91.22	91.75	92.01			
madelon	F	52.54	52.36	53.54	53.57	57.53	52.78	52.95	53.42	56.03	57.43	52.40	52.99	53.77	56.09	58.86	2/3/0	4/1/0	2/3/0
	A	52.83	53.50	54.77	55.00	56.38	52.67	54.00	54.80	56.17	56.67	53.67	53.90	55.00	55.33	56.67			
win/tie/loss against SVM							3/4/1	5/1/2	5/3/0	7/1/0	7/1/0	5/3/0	6/2/0	8/0/0	7/1/0	7/1/0			
proposed win/tie/loss against LapSVM												6/2/0	5/3/0	7/1/0	2/4/2	2/5/1			

where $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, and TP, FP, TN, FN are true positive, fault positive, true negative, and fault negative, respectively. For balanced data, F-score and Accuracy are combined for comparison while for imbalanced data like usps and ionosphere, we only focus on the overall metric F-score.

Baselines

RLS and SVM sever as supervised baselines, with both linear or RBF kernel. LapRLS and LapSVM sever as semi-supervised baselines, with both linear or RBF kernel. Following a clear and future-proof framework [3] of semi-supervised classification algorithms, the proposed method is a manifold regularization algorithm with a shallow structure [55], so the harmonic mixtures (HM) model [52] is as the counterpart, as it combines mixture model and graph which also realizes data manifold approximation in an inductive way.

The parameters listed in Table 2.2 are chosen by a 3-fold cross validation. The number of nearest neighbors in the graph construction is fixed to 5.

2.5.2 Comparison results

Comparison results among all the algorithms are listed in Table 2.3 with labeled data proportion fixed at 10% for space limitation.

As can be seen from Table 2.3 that the proposed method works quite well. Specifically, in terms of wins, the proposed method performs the best in comparison with the other three semi-supervised algorithms.

When comparing with two supervised baselines, LapRLS and LapSVM show performance reductions in one or two cases, while HM having even worse performance. In contrast, the proposed method is never inferior to these two supervised baselines.

HM has no advantage among the semi-supervised algorithms neither. LapRLS and LapSVM have similar performance, and the proposed method outperform these two in most cases.

2.5.3 Influence of the number of labeled data

We also evaluate the proposed method with various proportions of labeled data, the results are listed in Table 2.4 with supervised baseline SVM and semi-supervised baseline LapSVM due to space limitation.

When we have little labeled data, in the cases of 2%, 5%, in contrast to LapSVM that reduces performance on some cases, the proposed method never shows decreased performance. The advantage of the proposed method over SVM and LapSVM reaches the highest when the label data proportion is 10%. Moreover, we can see the performance of LapSVM and the proposed method are quite similar in the case of 20% and 30% cases, and both of they are found to be highly competitive to SVM.

2.5.4 Influence of imbalance

Although the balance data classes is often a mild assumption, it might still be violated in some cases, and usps and ionosphere are two imbalance data sets.

Table 2.4 shows that in the balanced data sets, the proposed method has better advantage over SVM, while LapSVM having more tie and even loss against SVM.

The proposed method only lost to LapSVM in these two imbalance data sets. Therefore, the proposed method can not have stable advantage over LapSVM in imbalanced cases, since the imbalance has not been considered in the partitioning, which is the direction of future work.

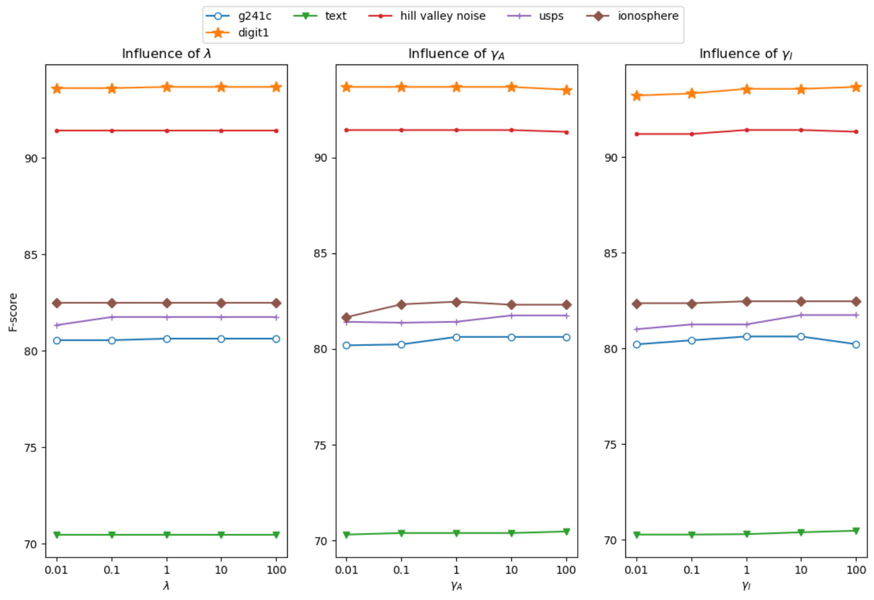

FIGURE 2.4: Parameter influence with 10% labeled data.

2.5.5 Influence of parameters

Figure.2.4 studies the influence of λ, γ_A and γ_I on 6 representative data sets (the results on other data sets are similar) with 10% labeled data by fixing other parameters as default values. It can be seen that, though the number of labeled instances is small, the performance of the proposed method is insensitive to the setting of the parameters. One possible reason is that, rather than interpolating multiple data points, the proposed methods interpolating multiple local linear models. This property makes the proposed method more attractive, especially when the number of labeled instances is too small to afford a reliable model selection.

2.5.6 Influence of kernel and graph

We further study the influence of the kernel and the graph respectively.

The left part of Table 2.5 lists the performance of LapSVM with three different kernels on 6 balanced data sets. Linear and RBF kernel are the most commonly used linear and non-linear kernel respectively. As we can see, RBF kernel fails dramatically in text, so the choice of kernel is very data-dependent. In the contrast, the quasi-linear kernel

TABLE 2.5: Influence of kernel and graph with 10% labeled data.

	Kernel				Graph		
	Linear	RBF	Quasi		Un	GCLI	Ours
g241c	79.57	79.61	80.11	g241c	80.11	80.66	80.63
digit1	93.50	93.67	93.67	digit1	93.67	93.07	93.67
text	68.94	0.00	70.44	text	70.44	70.44	70.48
hill valley noise	89.67	89.55	90.88	hill valley noise	90.88	91.11	91.42
splice	91.22	91.34	91.44	splice	91.44	91.47	91.56
madelon	53.20	53.42	53.63	madelon	53.63	53.63	53.77
w/t/l aginst linear kernel		4/0/2	6/0/0	w/t/l aginst un		3/2/1	5/1/0
w/t/l aginst rbf kernel			5/1/0	w/t/l aginst GCLI			5/0/1

never show such a significant reduction. The quasi-linear kernel is defined as the inner product of the input in a finite dimensional spanned feature space which is learned from the data set itself, and the flexibility is automatically adopt to data set.

The right part of Table 2.5 shows the performance of LapSVM with quasi-linear kernel, and three different graph construction methods. The baseline is the unsupervised graph construction method. The proposed method first build a *k*NN graph with quasi-linear kernel as similarity measurement and edge weighting, then cut off the edges between two data if they have different pseudo label, so it is a label guided graph construction method. Most of the label guided graph construction methods build a new complex model, we choose a relatively simple one GCLI [77] to do the comparison. We can see the proposed method perform stably, unlike GCLI, the proposed method never fails against the unsupervised graph construction method.

2.6 Summary

The proposed method firstly detect the information of potential separation boundary, then ingeniously incorporated it into LapSVM both in the kernel level and graph level. From the aspect of kernel machine, the incorporation of detected information of potential separation boundary makes a kernel function have higher theoretical interpretability and data adaptability. From the aspect of graph construction, the limited labels are fully used in an explicit and simple way. The simulation results shows the effectiveness of the proposed method.

Chapter 3

LapRLS Based on Piecewise Linear Model

3.1 Background

[1]The semi-supervised classification (SSC) is of immense practical interest in a wide range of applications, such as biology, image processing, natural language processing where the labeled data is expensive while abundant unlabeled data is available. Furthermore, much of human learning involves a small amount of direct instruction combined with large amounts of observations. Hence, the semi-supervised classification is a plausible model for human learning [10, 67, 68].

The SSC considers a classification problem of learning from both labeled and unlabeled data, typically with a small amount of labeled data and a great deal of unlabeled data, under the prerequisite that the distribution of labeled data and unlabeled data has to be the same. Thus, certain assumptions have to hold, such as smoothness assumption, cluster assumption and manifold assumption. The approaches such as change of representation [81], generative models [82] and low-density separation [83], and graph-based methods [70] roughly correspond to these underlying assumptions respectively.

[1]This chapter is mainly extended from the Journal paper: Y. Ren, W. Li and J. Hu, "A Semi-Supervised Classifier Based on Piecewise Linear Regression Model Using Gated Linear Network", *IEEJ Transactions on Electrical and Electronic Engineering*, Vol.15, No.7, pp.1048-1056, July, 2020.

A Laplacian regularized least squares (LapRLS) algorithm [51, 59], which assumes that if two instances are close in the intrinsic geometry of margin distribution then the conditional distributions are similar, has been proposed as a typical graph-based method based on the manifold assumption which assumes that the (high-dimensional) data lie (roughly) on a low dimensional manifold. A LapRLS extends the ridge regression, a standard supervised regularization algorithm, by incorporating additional information about the geometric structure of the margin distribution of data which is estimated using the graph Laplacian associated to all the training instances.

Another algorithmic way to extend a supervised learning method to an SSC method is solely adding a consistency regularization to the training of a neural network while leaving the training and model unchanged from what would be used in the fully-supervised setting [84]. A consistency regularization, which assumes realistic perturbations of instances should not significantly change the outputs, can be seen as a way of leveraging the unlabeled data to find a smooth data manifold, and it has produced some currently state-of-art SSC methods. Temporal Ensembling (TE) [85] uses an exponentially accumulated average of outputs as consistency target where the output is itself stochastic due to common regularization technique such as dropout and adding noise. Instead of relying on the built-in stochasticity of output, Virtual Adversarial Training (VAT) [58] directly approximates a tiny perturbation to add to input which would most significantly affect the output of prediction function.

In order to effectively utilize data manifold learned from both training instances and limited labels we have, in this chapter we propose a semi-supervised classifier based on a piecewise linear regression model. The idea is to introduce a gated linear network to realize the piecewise linear model approximating the nonlinear separation boundary, in which a semi-supervised gating mechanism is designed for realizing the partitioning. Fig.3.1 shows an overview of the proposed method, which consists of two steps. In the first step, a gating mechanism is pre-trained to generate a set of binary gating sequences to control the base linear models in the gated linear network. Each binary gating sequence corresponds to one partition. Instead of using an unsupervised manner [62], the gating mechanism is trained as a semi-supervised neural network which can capture

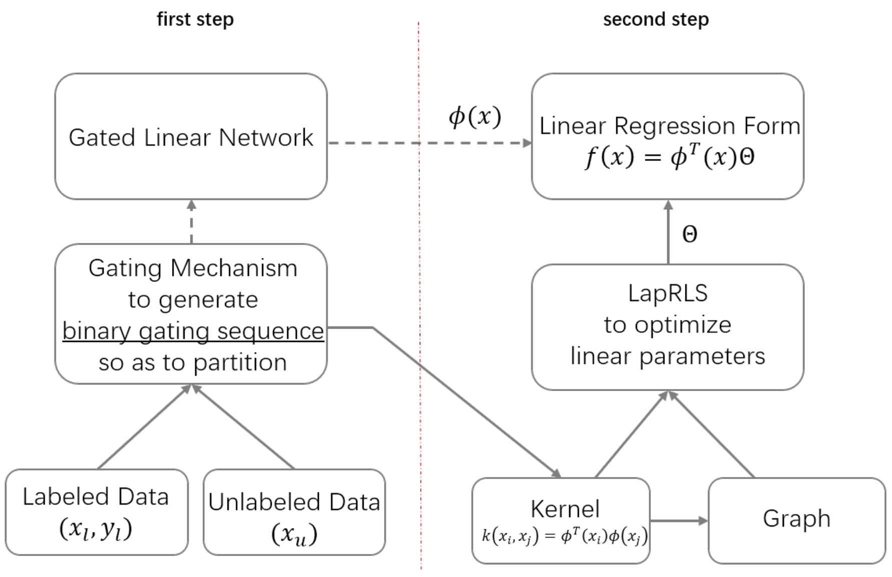
first step
second step
Gated Linear Network
$\phi(x)$
Linear Regression Form
$f(x) = \phi^T(x)\Theta$
Θ
Gating Mechanism to generate binary gating sequence so as to partition
LapRLS to optimize linear parameters
Labeled Data (x_l, y_l)
Unlabeled Data (x_u)
Kernel $k(x_i, x_j) = \phi^T(x_i)\phi(x_j)$
Graph

FIGURE 3.1: The overview of the proposed method.

the data manifold from both labeled and unlabeled data, by using a k-sparse aggressive sparsity strategy. In the second step, the gated linear network is first transformed into a linear regression form and the linear parameters are then optimized globally by a Laplacian regularized least squares (LapRLS) algorithm using a kernel function composed with the gating sequences obtained in the first step [86]. Moreover, we use the kernel function as a similarity measurement to construct the graph in LapRLS. In this way, we capture data manifold from both the instances and the labels we have, and the captured data manifold is ingeniously incorporated both into the kernel and the graph Laplacian in the LapRLS. Numerical experiments on various real-world datasets exhibit the effectiveness of the proposed method.

The rest of this chapter is organized as follows. Section 3.2 formulates a unified framework for a gated linear network based piecewise linear regression model. In Section 3.3, we design a semi-supervised gating mechanism. Section 3.4 shows the optimization process using LapRLS. Section 3.5 presents the simulation results of a series of experiments to validate the effectiveness of our proposed method. Finally, the conclusions are drawn in Section 3.6.

3.2 Piecewise Linear Model

In the case of a semi-supervised binary classification problem, there is a probability distribution P on $X \times \mathbb{R}^n$ and a prediction function is learned from data sampled according to P, labeled pairs (x, y) are sampled from the joint distribution $\mathcal{P}_{XY}$ of P while unlabeled instances $x \in X$ are sampled from the marginal distribution $\mathcal{P}_X$ of P. Suppose we have a training set containing L labeled pairs $\{(x_l, y_l)\}_{l=1}^{L}$ with $y_l \in \{1, -1\}$ and $N - L$ unlabeled instances $\{x_u\}_{u=L+1}^{N}$. X_N represents all of labeled and unlabeled instances in the training dataset.

Consider the case where applying a multilayer perceptron (MLP) with rectified linear unit (ReLU) to approximate the nonlinear separation boundary. Since $\mathrm{ReLU}(z) = \max\{0, z\}$ is a compound activation function of linear unit weighted by a step function $\mathrm{ReLU}(z) = z \cdot S(z)$, the rectified MLP can be separated into a gated linear network and a gating mechanism. When considering only simple cases where the rectified neural network is a two-layer MLP, the gated linear network can be expressed by

$$f(x) = \sum_{j=1}^{M} (\Omega_j^T x + b_j) \cdot g_j(x) + b \tag{3.1}$$

where $g_j(x)$ $(j = 1, ..., M)$ is a step function of gate control signals generated by a pre-trained MLP, and the linear parameters Ω_j, b_j, b may be further optimized using LapRLS algorithm, which may be considered as a fine-tuning of the pre-trained MLP.

By given the gating sequence $\mathbf{g}(x) = [g_1(x), \cdots, g_M(x)]^T$ and introducing two vectors $\phi(x)$ and Θ, defined by

$$\begin{aligned} \phi(x) &= [1, g_1(x), x^T g_1(x), \cdots, g_M(x), x^T g_M(x)]^T \\ &= [1, \mathbf{g}^T(x) \otimes [1, x^T]]^T \end{aligned} \tag{3.2}$$

$$\Theta = [b, b_1, \Omega_1^T, \cdots, b_M, \Omega_M^T]^T \tag{3.3}$$

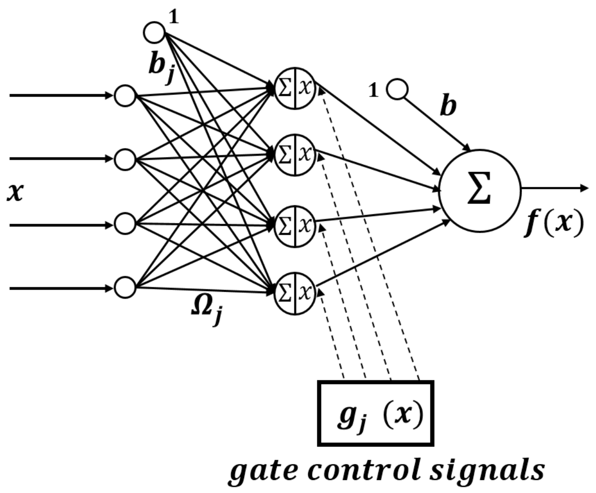

FIGURE 3.2: The architecture of gated linear network realizing piecewise linear regression model.

where the symbol $\otimes$ denotes Kronecker product, the gated linear network described by Eq.(3.1) can then be further expressed by a linear regression form as

$$f(x) = \phi^T(x)\Theta. \tag{3.4}$$

where $\phi(x)$ is the known regression vector while Θ is the linear parameters to be further optimized.

On the other hand, we can see from Fig.3.2 that the gated linear network realizes a piecewise linear model, where $\Omega_j^T x + b_j (j = 1, \cdots, M)$ are a set of linear base models, b is the bias parameter, and $g_j(x)$ is the gate control signal determining whether the j-th linear base model plays a role; $g_j(x) = 1$ indicates that the j-th linear base model does play a role while $g_j(x) = 0$ meaning it does not. Different gating sequences, which mean different combinations of linear base models, correspond to different linear submodels, thus different partitions. In this way, we are able to pre-train a neural network to realize a sophisticated partitioning and build a set of linear submodels for the piecewise linear model by generating a set of binary gating sequences using the pre-trained neural network.

It is well known that a neural network can be trained to capture data manifold [87], and the captured data manifold can guide the generation of gate control signals in our piecewise linear regression model [62][86]. In Section 3.3, we design a semi-supervised

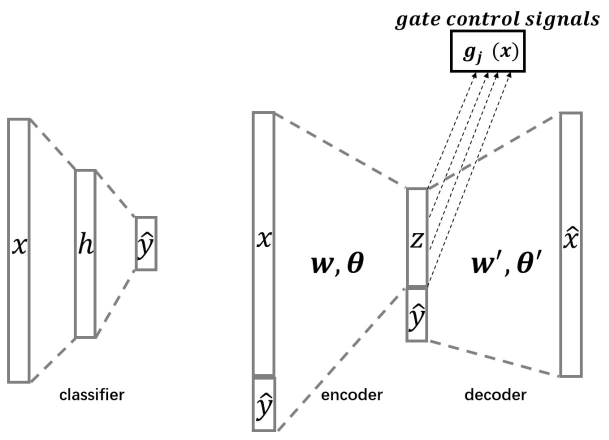

FIGURE 3.3: A labeled autoencoder as the gating mechanism, consisting of a classifier, an encoder and a decoder.

gating mechanism based on the neural network, which can capture data manifold from both labeled and unlabeled data, to generate gate control signals. Then in Section 3.4, we optimize the linear parameters Θ by applying a LapRLS algorithm.

3.3 Semi-Supervised Gating Mechanism

By using a gated linear network, as shown in Fig.3.2, the estimation of break-points in the piecewise linear regression model is replaced by generating binary gating sequences to control the linear base models. We will design a semi-supervised gating mechanism based on a label guided k-sparse autoencoder [88] in order to leverage the label information.

3.3.1 Structure of the gating mechanism

Fig.3.3 shows a gating mechanism based on a labeled autoencoder, consisting of a classifier, an encoder and a decoder. Inspired by Ref. [49], we design the semi-supervised gating mechanism by adding a classifier and introducing label guidance into an autoencoder. The classifier is described by

$$\hat{y} = \pi_\phi(x) \tag{3.5}$$

where $\pi_\phi(\cdot)$ is represented as a feedforward neural network and $\hat{y}$ is a probability vector after a softmax activation function, and the encoder by

$$z = a(\mathbf{w}^T(x,y) + \theta), \tag{3.6}$$

while the decoder by

$$\hat{x} = \mathbf{w}'^T(z,y) + \theta'. \tag{3.7}$$

Since the encoder after training will generate the gate control signals [62], following the idea of k-sparse strategy, by defining a set $\Gamma = supp_k\{\mathbf{w}^T(x,y) + \theta\}$ containing hidden units with top-$k\%$ activation values, the activation function $a(\cdot)$ can be seen as a compound activation function of linear unit weighted by a step function:

$$\begin{aligned} a(w_j^T x + \theta_j) &= (w_j^T x + \theta_j) \cdot S(w_j^T x + \theta_j) \\ S(w_j^T x + \theta_j) &= \begin{cases} 1, & j \in \Gamma = supp_k\{\mathbf{w}^T x + \theta\} \\ 0, & j \notin \Gamma = supp_k\{\mathbf{w}^T x + \theta\} \end{cases} \end{aligned} \tag{3.8}$$

where there are M neurons in the hidden representation z and the sparsity level k is the only hyper-parameter to tune, and the data manifold is captured by this k-sparse strategy.

After training, the generation of gate signals is guided by the captured data manifold:

$$g_j(x) = \begin{cases} 1, & j \in \Gamma = supp_k\{\hat{\mathbf{w}}^T(x,\hat{y}) + \hat{\theta}\} \\ 0, & j \notin \Gamma = supp_k\{\hat{\mathbf{w}}^T(x,\hat{y}) + \hat{\theta}\} \end{cases} \tag{3.9}$$

where $\hat{\mathbf{w}}$ and $\hat{\theta}$ are the estimates of $\mathbf{w}$ and θ.

3.3.2 Training of the gating mechanism

In the training process, firstly we will get probability vectors $\hat{y_l}$, $\hat{y_u}$ for labeled and unlabeled instances respectively through the classifier. For labeled instances x_l, the true labels y_l are concatenated with the instances as the input of encoder, which are also concatenated with the representation as the input of decoder. For unlabeled instances x_u, we concatenate all the possible labels with the instance as the input of encoder, and also concatenate the same labels with the representation as the input of decoder to get all possible loss to form a loss vector L_u by calculating the mean square error between the input and the reconstruction $||\hat{x_u} - x_u||_2^2$. In the context of binary classification, here are two possible labels so that we can get two loss values in the loss vector L_u.

The final loss function consists of the losses of both labeled instances $\mathcal{L}(x_l, y_l)$ and unlabeled instances $\mathcal{U}(x_u)$

$$Loss = \mathcal{L}(x_l, y_l) + \beta \cdot \mathcal{U}(x_u) \tag{3.10}$$

where β is a coefficient, and

$$\mathcal{L}(x_l, y_l) = \alpha H(y_l, \hat{y}_l) + ||\hat{x}_l - x_l||_2^2 \tag{3.11}$$

where α is a coefficient, and $H(y_l, \hat{y}_l) = -\sum y_l \log \hat{y}_l$ is the cross entropy between y_l and $\hat{y}_l$, and

$$\mathcal{U}(x_u) = H(\hat{y}_u) + \hat{y}_u \cdot L_u \tag{3.12}$$

where $H(\hat{y}_u) = -\sum \hat{y}_u \log \hat{y}_u$ is the information entropy of $\hat{y}_u$, and $\hat{y}_u \cdot L_u$ stands for weighted loss. In our simulation, the coefficients α and β are set to $\alpha = 0.1N$ and $\beta = 1$, where N is the number of training samples.

3.3.3 Gate control signals

The gate control signals are generated as Eq.(3.9), and note that the data manifold, which guides the generation of gate control signals, is learned from both training instances and limited labels. Different gating sequences $\mathbf{g}(x)$ stand for different combinations of $p = \lfloor M * k\% \rfloor$ linear base models, thus different linear submodels which correspond to different partitions. The number of linear submodels in the piecewise linear regression model may be up to C_M^p, and it is easy to use Pascal's Triangle to show that C_M^p achieves its largest value when $k = 50$.

3.4 Laplacian Regularized Least Squares

3.4.1 Ridge regression with manifold regularization

Given the regression vector $\phi(x)$, we optimize Θ in Eq.(3.4) by applying a ridge regression with manifold regularization (LapRLS)[89]. As a result, we have an optimization problem described by

$$\begin{aligned}\min_{\Theta} \ell = \min_{\Theta} \frac{1}{2} \sum_{k=1}^{N} [y_k - \phi^T(x_k)\Theta]^2 \\ + \frac{\gamma_A}{2}\Theta^T\Theta + \frac{\gamma_I}{2} F^T \mathbf{L} F\end{aligned} \tag{3.13}$$

where $F = \Phi_N \Theta$ and

$$\Phi_N = [\phi(x_1), \cdots, \phi(x_L), \phi(x_{L+1}), \cdots, \phi(x_N)]^T.$$

So we have

$$\begin{aligned}\min_{\Theta} \ell = \min_{\Theta} \frac{1}{2} \|(\hat{Y} - J\Phi_N\Theta)\|^2 \\ + \frac{\gamma_A}{2}\Theta^T\Theta + \frac{\gamma_I}{2}\Theta^T\Phi_N^T\mathbf{L}\Phi_N\Theta\end{aligned} \tag{3.14}$$

where J is an $N \times N$ diagonal matrix given by $J = \text{diag}(1, \cdots, 1, 0, \cdots, 0)$ with the first l diagonal entries as 1 and the rest 0, and $\hat{Y} = [y_1, \cdots, y_L, 0, \cdots, 0]^T$ is an augmented version of Y where y_i for labeled instances while 0 for unlabeled instances. For notation simplicity, we ignore every normalization coefficient of each term here, γ_A controls the complexity of the function in the ambient space and γ_I controls the complexity of the function in the intrinsic geometry of $\mathcal{P}_X$, $\mathbf{L}$ is the graph Laplacian built from X_N. The derivative of the objective function vanishes at the minimizer:

$$\begin{aligned}\frac{\partial \ell}{\partial \Theta} = 0 \\ (\hat{Y} - J\Phi_N\Theta)^T(-J\Phi_N) + \gamma_A\Theta^T + \gamma_I\Theta^T\Phi_N^T\mathbf{L}\Phi_N = 0\end{aligned} \tag{3.15}$$

If Φ_N has more rows than columns and is of full column rank, we can have following closed form solution for the over-determined problem:

$$\Theta^* = (\Phi_N^T J \Phi_N + \gamma_A I + \gamma_I \Phi_N^T \mathbf{L} \Phi_N)^{-1} \Phi_N^T \hat{Y} \tag{3.16}$$

while if Φ_N having more columns than rows and is also of full row rank, we can have following closed form solution for the under-determined problem:

$$\Theta^* = \Phi_N^T (J \Phi_N \Phi_N^T + \gamma_A I + \gamma_I \mathbf{L} \Phi_N \Phi_N^T)^{-1} \hat{Y} \tag{3.17}$$

3.4.2 A kernel formulation

Eq.(3.16) and Eq.(3.17) can naturally be related to a kernel formulation since both of them contain inner product of regression vectors Φ_N. By combining Eq.(3.17) with the original model in Eq.(3.4) as an example:

$$\begin{aligned} f(x) &= \phi^T(x)\Theta^* \\ &= \phi^T(x)\Phi_N^T (J \Phi_N \Phi_N^T + \gamma_A I + \gamma_I \mathbf{L} \Phi_N \Phi_N^T)^{-1} \hat{Y} \end{aligned} \tag{3.18}$$

Under the circumstance of our piecewise linear regression model, the kernel is defined as the inner product of regression vectors according to Eq.(3.2)

$$\begin{aligned} k(x_i, x_j) &= \phi^T(x_i)\phi(x_j) \\ &= 1 + (1 + x_i^T x_j)\mathbf{g}^T(x_i)\mathbf{g}(x_j) \end{aligned} \tag{3.19}$$

which is called a quasi-linear kernel[62, 90]. So $K_{NN} = \Phi_N \Phi_N^T = [K_N(x_1), \cdots, K_N(x_N)]^T$ where $K_N(x) = \Phi_N \phi(x) = [k(x, x_1), \cdots, k(x, x_N)]^T$ and we have

$$f(x) = K_N^T(x)(J K_{NN} + \gamma_A I + \gamma_I \mathbf{L} K_{NN})^{-1} \hat{Y} \tag{3.20}$$

Eq.(3.19) shows that by using the gating mechanism we can get a mapping function $\phi(x)$ defined by Eq.(3.2), which maps an input to a finite dimensional spanned feature space. Then in the feature space, we can discuss the optimization of a linear regression model.

Therefore, it is natural to have a direct comparison among a linear LapRLS, a RBF LapRLS, and our piecewise linear regression model at a kernel level. As a reminder, here are the linear kernel in Eq.(3.21) and the RBF kernel in Eq.(3.22).

$$k(x_i, x_j) = x_i^T x_j \tag{3.21}$$

$$k(x_i, x_j) = \exp(-\gamma \|x_i^T - x_j\|_2^2) \tag{3.22}$$

3.4.3 Graph construction

Furthermore, the quasi-linear kernel is applied to graph construction in the LapRLS. Graph Laplacian is a matrix representation of a graph. Given an undirected graph with N nodes, its graph Laplacian is defined as:

$$\mathbf{L} = D - W \tag{3.23}$$

where W is adjacency matrix defined by

$$W(x_i, x_j) = B_{ij} k(x_i, x_j) \tag{3.24}$$

and D is degree matrix formulated as

$$\begin{aligned} d_i &= \sum_{j=1}^{N} w_{ij} \\ D &= \mathrm{diag}(d_1, \cdots, d_i, \cdots, d_N) \end{aligned} \tag{3.25}$$

Graph construction can be decomposed into graph sparsification and edge weighting. B is the edge sparsification matrix, if there is an edge between the nodes x_i and x_j, $B_{ij} = 1$, else, $B_{ij} = 0$, and there are no self-loops, $B_{ii} = 0, \forall i = 1, \cdots, N$. In a traditional method, we construct B using a standard neighbor algorithm like k-nearest neighbor (k-NN) where undirected weights between a node and its k-nearest neighbors are greedily added, and a Gaussian kernel is often used to evaluate the weight of each edge. So both B and k are based on an Euclidian distance in the input space.

However, the distance between instances under an Euclidean distance becomes ambiguous in higher input dimension [91]. On one hand, the quasi-linear kernel is an inner product of the input in the spanned space. On the other hand, the data manifold has been ingeniously incorporated into the quasi-linear kernel. So we proposed a method to construct the graph using the quasi-linear kernel K_{NN} as a similarity measurement.

We first find the closest h instances to each instance based on the values in the quasi-linear kernel matrix, the greater the value, the closer the two instances. Define a set $\Lambda = supp_h\{k_i\}$ containing elements with maximum h values in $k_i = [k_{i1}, \cdots, k_{ij}, \cdots, k_{iN}]^T$, where $K_{NN} = [k_1, \cdots, k_i, \cdots, k_N]^T$:

$$\underset{1\le i,j\le N}{B_{ij}} = \begin{cases} 1, & j \neq i \; and \; j \in \Lambda = supp_h\{k_i\} \\ 0, & otherwise \end{cases} \tag{3.26}$$

Then we need to make B symmetric since the graph is assumed to be undirected:

$$\underset{1\le i,j\le N}{B_{ij}} = \begin{cases} 1, & B_{ij} + B_{ji} > 0 \\ 0, & B_{ij} + B_{ji} \le 0 \end{cases} \tag{3.27}$$

So the adjacency matrix W_{NN} is given by:

$$\begin{aligned} W_{NN}(x_i, x_j) &= B_{ij} k(x_i, x_j) \\ &= B_{ij}[1 + (1 + x_i^T x_j)\mathbf{g}^T(x_i)\mathbf{g}(x_j)] \end{aligned} \tag{3.28}$$

where B_{ij} is constructed as Eq.(3.26), Eq.(3.27) and $k(x_i, x_j)$ is the quasi-linear kernel. Moreover, a normalized Laplacian

$$\tilde{\mathbf{L}} = D^{-\frac{1}{2}} \mathbf{L} D^{-\frac{1}{2}} \tag{3.29}$$

can be used interchangeably also. Using $\tilde{\mathbf{L}}$ instead of $\mathbf{L}$ provides certain theoretical guarantees[79] and seems to perform as well or better in practical tasks.

Naturally we get a normalized graph Laplacian $\tilde{\mathbf{L}}_{\mathbf{NN}}$ based on the quasi-linear kernel K_{NN} according to Eq.(3.28), Eq.(3.25) Eq.(3.23) and Eq.(3.29).

At last, the piecewise linear regression classifier takes the form of

$$y = \mathrm{sign}[f(x)] \tag{3.30}$$

where

$$f(x) = K_N^T(x)(JK_{NN} + \gamma_A I + \gamma_I \tilde{\mathbf{L}}_{\mathbf{NN}} K_{NN})^{-1}\hat{Y} \tag{3.31}$$

according to Eq.(3.18).

To summarize, we incorporate the data manifold captured in the semi-supervised gating mechanism into both kernel and graph Laplacian in the LapRLS. On one hand, we avoid calculating the Euclidean distance in the input space. Instead, we measure the similarity between instances in the spanned space via the inner product, see Eq.(3.2). On the other hand, we construct the graph in a semi-supervised way, and it is expected to work better than the traditional k-NN algorithm which is unsupervised. As a result, the semi-supervised nature of the entire model is preserved both at a kernel level and a graph level.

3.5 Numerical Experiments

In this section, a set of real-world datasets from UCI machine learning repository are used to show the effectiveness of our proposed method. Table 3.1 briefly describes the datasets, referred to Ref. [80] for more details.

3.5.1 Evaluation metrics

As for the evaluation metrics, we use *Accuracy* and the overall metric *F-score*, defined as (2.28).

TABLE 3.1: Dataset description.

datasets	features	positive	negative	train	test
Splice	60	1535	1655	2233	957
Digit1	241	766	734	1200	300
g241c	241	750	750	1200	300
g241n	241	748	752	1200	300
Madelon	500	1300	1300	2000	600
Gisette	5000	3500	3500	6000	1000
text	11960	750	750	1200	300
ESR_{1-5}	178	2300	2300	3220	1380

3.5.2 Experimental settings

We keep the split of datasets for training and testing if they have a predefined split, otherwise, we integrate all the data we have and randomly divide it into a 70% training set and a 30% testing set using stratified sampling to guarantee a similar input and label distribution. All the features of continuous data are standardized into a zero mean and unit variance while they are further normalized into a vector with a unit $l2$-norm if fed into a kernel calculation. We manually remove some labels in the training set to make it a semi-supervised classification dataset.

TE, VAT and labeled autoencoder (lae) are all based on neural networks which have many hyper-parameters to be determined in advance. To have a fair comparison, TE, VAT and the classifier of lae share all the hyper-parameters like network structure, learning rate and training epoch in our experiments as showed in Tabel 3.2. When training TE and VAT on MNIST, SVHN and CIFAR-10, a “WRN-28-2”, i.e. ResNet with depth 28 and width 2 [92], is used. All the neural networks are implemented based on PyTorch [93]. All the hyper-parameters of LapRLS are listed in Table 3.3, where the nearest neighbors in the graph construction is fixed to be 10 for all the methods.

When compared with different semi-supervised (SS) classifiers as discussed in Subsection 3.5.3, the hyper-parameters k, γ_A and γ_I in our proposed method are chosen by a 3-fold cross validation. The training of all the SS classifiers are repeated for 10 rounds due to the random initialization of networks weights. When evaluating our proposed algorithm from the aspects of kernel machine and graph construction as discussed in Subsections 3.5.4 and 3.5.5, we obtain a fixed gate mechanism, which generates the gate control signals to compose the quasi-linear kernel, by using cross validation method.

TABLE 3.2: Parameter setting of TE, VAT and lae.

data	NN structure (TE,VAT, classifier of lae)	$dim(z)$ in lae	learning rate	epoch
Splice	$dim(\text{input})\times50\times2$	1000	2e-4	100
Digital				
g241c		500		
g241n				
Madelon				
Gisette	$dim(\text{input})\times500\times50\times2$	1000		150
text	$dim(\text{input})\times1000\times100\times2$		3e-4	
ESR	$dim(\text{input})\times50\times2$	50		100

TABLE 3.3: Parameter setting of LapRLS.

parameter	description	value
k	the sparsity level of gating mechanism	$\{5, 10, 30\}$
γ	width of RBF (Gaussian) kernel	$\{0.01, 0.1, 1, 10, 100\}$
γ_A	complexity of the function in ambient space	$\{0.01, 0.1, 1, 10, 100\}$
γ_I	complexity of the intrinsic geometry of $\mathcal{P}_X$	$\{0.01, 0.1, 1, 10, 100\}$

3.5.3 Compared with different classifiers

Table 3.4 lists the performance of five models under the datasets with 5%, 10%, 15%, 20% and 25% labeled data.

We use the ridge regression (RLS) as a supervised baseline, and the LapRLS as a semi-supervised baseline. Obviously, the labeled autoencoder (lae) is very unqualified as a semi-supervised classifier, since it is not even as good as the supervised baseline in many cases. We can observe that our proposed method outperform other methods on most of the datasets, even though the labeled autoencoder, as the first step of our method, plays a poor role at classification. As a gating mechanism it is only used to extract rough nonlinear information, and the linear parameters of piecewise linear regression model are optimized in the second step via LapRLS. This two-step strategy in our proposed model makes sure the stability of the entire model even when the gating mechanism

TABLE 3.4: Test results (%) of different SSC models on binary datasets with different label rate.

		5%		10%		15%		20%		25%	
	counterparts	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
splice	RLS	85.05	84.80	89.43	89.19	90.49	90.28	91.83	91.69	91.83	91.69
	LapRLS	85.24	84.80	91.39	90.91	92.42	92.16	92.87	92.63	93.13	92.95
	proposed	**87.61±0.99**	**87.24±1.03**	**91.70±0.62**	**91.49±0.65**	**93.07±0.46**	**92.79±0.47**	**93.09±0.39**	**92.79±0.38**	**93.39±0.54**	**93.10±0.53**
	lae	83.15±1.86	82.71±1.86	87.26±2.04	87.08±2.01	88.48±1.38	88.26±1.40	88.92±0.94	88.54±0.92	89.80±0.81	89.53±0.84
	TE	81.66±0.85	81.16±1.10	86.60±1.17	86.19±1.26	87.14±0.97	86.60±0.72	88.87±0.80	88.45±0.84	91.28±0.48	90.86±0.50
	VAT	86.31±1.11	85.66±1.59	88.86±1.74	88.34±1.55	90.19±1.65	89.76±1.26	91.05±1.32	90.64±1.29	91.93±0.79	91.50±0.84
digit1	RLS	89.82	90.33	91.99	92.33	92.96	93.33	92.36	92.67	94.08	94.33
	LapRLS	90.11	91.00	93.47	93.67	94.77	95.00	95.59	95.67	96.58	96.67
	proposed	**92.64±0.73**	**93.07±0.66**	**95.23±0.34**	**95.30±0.31**	**95.61±0.55**	**95.70±0.43**	**96.21±0.11**	**96.30±0.10**	**96.93±0.41**	**97.00±0.39**
	lae	89.51±1.76	90.0±1.64	91.39±1.62	91.53±1.61	89.21±1.63	89.83±1.49	91.88±1.64	92.03±1.62	92.28±1.46	92.33±1.45
	TE	91.59±1.03	92.13±0.95	92.05±0.60	92.30±0.59	95.60±1.07	95.83±0.99	95.29±0.80	95.56±0.75	93.58±0.64	93.93±0.57
	VAT	89.33±2.27	91.47±1.43	94.20±1.69	94.40±1.80	95.55±1.87	95.63±1.20	95.66±1.22	95.97±1.18	96.89±1.13	96.40±1.02
g241c	RLS	69.93	71.33	75.77	76.33	78.00	78.41	80.70	81.67	79.73	80.33
	LapRLS	71.08	72.33	80.26	80.00	**81.48**	**81.67**	81.10	81.67	**83.44**	**83.33**
	proposed	**72.52±0.77**	**73.23±0.63**	**80.68±0.94**	**80.33±0.91**	80.59±0.79	80.93±0.79	**81.40±0.57**	**82.13±0.54**	83.25±1.03	83.13±1.10
	lae	65.43±4.25	65.97±4.27	69.74±3.56	69.93±3.59	69.10±3.03	70.40±2.82	74.93±3.00	75.50±3.04	76.42±1.91	76.23±1.86
	TE	69.38±2.25	70.17±1.99	79.46±1.59	79.53±1.54	80.30±1.60	80.00±1.73	80.82±1.23	80.83±1.22	83.34±1.05	83.10±1.07
	VAT	70.72±1.74	71.10±1.63	80.60±1.53	80.67±1.30	79.00±0.89	78.73±0.83	80.79±1.52	80.60±1.97	82.70±2.20	80.83±2.04
g241n	RLS	65.37	52.67	75.33	75.33	78.62	79.33	84.93	85.33	84.54	85.00
	LapRLS	66.23	65.00	81.90	81.00	83.22	83.33	82.56	84.67	87.59	88.00
	proposed	68.33±1.11	67.60±1.75	**82.85±0.63**	**82.33±0.62**	**83.85±1.30**	**84.33±1.33**	**85.06±0.64**	**84.73±0.67**	**87.67±0.49**	**88.03±0.48**
	lae	60.71±4.54	60.00±4.06	66.67±2.86	66.60±2.52	71.69±2.63	72.13±2.80	73.41±2.74	73.27±2.29	75.15±2.89	75.47±3.01
	TE	**69.54±1.93**	**68.87±2.23**	76.08±1.05	76.83±1.21	78.75±1.05	78.13±1.00	80.90±0.97	80.93±0.81	83.06±1.05	82.90±1.20
	VAT	66.24±2.02	63.50±3.37	81.92±3.41	82.13±3.35	82.18±1.67	80.00±1.68	84.43±1.36	83.97±1.02	86.28±1.72	86.03±1.67
madelon	RLS	49.98	57.00	50.69	51.50	53.29	52.67	55.08	55.50	53.32	54.17
	LapRLS	52.92	54.33	53.83	54.83	53.74	54.67	55.80	56.17	55.78	55.33
	proposed	53.66±0.65	54.52±0.56	**54.38±0.68**	**55.45±0.64**	**55.18±0.64**	**56.00±0.51**	**56.53±0.39**	**57.02±0.37**	**58.06±0.48**	**57.93±0.47**
	lae	52.03±2.71	53.28±2.11	52.11±2.71	52.70±2.40	51.77±2.45	52.45±2.30	53.86±1.17	53.23±1.14	53.75±1.77	53.63±1.92
	TE	53.96±3.61	52.35±0.96	51.87±1.16	52.70±0.85	54.07±1.10	53.50±1.28	52.72±1.64	52.82±1.40	55.03±0.86	53.72±1.32
	VAT	**54.56±3.80**	**51.78±1.66**	53.27±2.51	51.27±1.23	53.84±1.71	51.35±1.57	52.55±1.92	52.50±1.42	54.26±1.41	52.92±1.20
gisette	RLS	93.63	93.50	94.79	94.80	95.09	95.00	95.24	95.20	96.00	96.00
	LapRLS	93.66	93.60	93.61	93.50	94.91	94.90	94.61	94.60	94.91	94.90
	proposed	93.98±0.28	93.88±0.29	**95.37±0.22**	**95.37±0.23**	95.90±0.12	95.90±0.12	96.03±0.20	95.99±0.20	96.65±0.31	96.62±0.31
	lae	91.45±0.77	91.25±0.85	93.05±0.77	93.00±0.78	94.18±0.67	94.15±0.69	94.37±0.66	94.52±0.65	94.86±0.41	94.81±0.41
	TE	94.06±0.57	94.01±0.61	95.18±0.30	95.20±0.31	**96.72±0.43**	**96.72±0.44**	**96.69±0.58**	**96.70±0.58**	96.72±0.24	96.70±0.24
	VAT	**94.17±1.10**	**94.47±0.86**	94.99±0.55	95.06±0.68	95.42±0.28	95.41±0.28	95.83±1.34	95.66±1.57	**96.72±0.14**	**96.20±0.14**
text	RLS	65.13	64.67	67.63	70.00	66.90	69.33	70.05	70.33	73.40	73.67
	LapRLS	66.67	66.33	69.03	68.00	70.75	71.33	70.92	72.67	73.54	74.33
	proposed	**67.50±0.78**	**66.30±0.48**	**69.84±0.89**	**69.50±1.47**	**71.67±0.66**	**71.87±0.48**	**71.33±1.48**	**72.67±2.30**	**74.11±0.43**	**74.57±0.45**
	lae	58.49±6.49	57.63±8.38	64.31±3.67	64.17±4.89	62.19±2.31	62.23±2.07	64.59±4.81	64.07±6.54	66.57±2.97	64.63±4.24
	TE	60.66±3.50	63.47±1.80	56.78±2.94	65.53±1.65	64.86±2.46	66.47±2.18	66.61±2.68	68.03±2.39	68.43±1.57	70.63±1.56
	VAT	61.84±3.34	64.57±1.97	63.56±2.84	65.83±2.06	66.81±2.34	67.00±2.82	67.91±2.10	70.77±3.06	70.12±2.21	72.73±1.58

TABLE 3.5: Performance on large-sized datasets (test errors (%)).

	MNIST		SVHN	CIFAR-10
	N=100	N=1000	N=1000	N=4000
TE	4.33±0.13	4.12±0.16	8.23±0.33	17.64±0.50
VAT	2.12±0.11	2.03±0.19	6.91±0.37	15.72±0.34
proposed	2.06±0.03	2.00±0.07	6.67±0.11	15.47±0.12

fluctuates greatly. Therefore, we believe the proposed model is a competitive model for a semi-supervised binary classification, and it compensates the shortcomings of the neural network to some extent by eliminating the parameter tuning and improving stability, especially for small and medium-sized datasets.

We should note a special case that LapRLS works worse than RLS in gisette, and the proposed method doesn't have advantage in gisette, maybe the graph Laplacian approximation is not adoptive to gissete.

The competitors, TE and VAT, are the deep learning methods. As reference, Table 3.5 shows the performance of the proposed algorithm, the TE and VAT algorithms on MNIST, SVHN and CIFAR-10 datasets. In the context of deep learning, the proposed algorithm uses the deep neural network pre-trained by the TE or VAT algorithm as gating mechanism to generate gate control signals and compose a deep quasi-linear kernel [90]. The proposed algorithm can be seen as a fine-tuning of the deep neural network models trained by the TE or VAT algorithms. From Table 3.5, we can see that the performance has been improved by the proposed algorithm on all datasets. The better performance on the pre-training step, the smaller improvement is obtained by the proposed algorithm.

However, it should be noted that the proposed method is time consuming in this case, it is recommendable to use TE or VAT for large scale datasets.

TABLE 3.6: Test results (%) of LapRLS with different kernels.

LapRLS			5%		10%		15%		20%		25%	
data	Graph	kernel	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
splice	kNN+ Gaussian	linear	85.24	84.80	91.16	90.91	91.44	91.22	91.86	91.69	92.31	92.16
		RBF	83.74	83.39	91.39	90.91	92.42	92.16	92.87	92.63	93.13	92.95
		ae-based	86.89	86.52	91.44	91.22	92.56	92.32	93.31	93.10	**93.74**	**93.57**
		lae-based	**88.05**	**87.62**	**91.88**	**91.69**	**93.07**	**92.79**	**93.51**	**93.26**	93.17	92.94
digit1		linear	88.73	89.33	93.33	93.33	94.12	94.33	95.30	95.33	95.56	95.67
		RBF	90.11	91.00	93.47	93.67	94.77	95.00	95.59	95.67	96.58	96.67
		ae-based	91.99	92.33	93.66	94.00	95.14	95.33	95.70	95.70	96.66	96.73
		lae-based	**93.37**	**93.67**	**95.24**	**95.33**	**96.19**	**96.33**	**96.58**	**96.67**	**96.91**	**97.00**
g241c		linear	70.63	72.00	79.47	79.33	81.48	81.67	81.10	81.67	82.31	82.67
		RBF	71.08	72.33	80.26	80.00	79.86	80.33	80.84	81.67	83.44	83.33
		ae-based	71.55	72.67	80.39	80.00	80.26	80.50	81.40	82.33	83.06	83.00
		lae-based	**73.22**	**73.67**	**80.92**	**80.67**	**81.88**	**82.00**	**82.35**	**83.00**	**84.39**	**84.33**
g241n		linear	66.23	65.00	81.01	80.00	79.73	79.67	85.26	84.67	83.28	83.67
		RBF	54.81	59.33	81.90	81.00	83.22	83.33	83.05	83.67	87.59	88.00
		ae-based	66.89	66.67	82.19	81.25	83.33	82.67	85.34	85.00	87.50	87.30
		lae-based	**68.63**	**68.00**	**82.91**	**82.00**	**84.04**	**83.67**	**85.81**	**85.33**	**87.66**	**87.33**
madelon		linear	52.92	54.33	53.01	54.50	53.74	54.67	55.37	55.67	55.78	55.33
		RBF	52.56	52.17	53.83	54.83	53.65	54.50	55.80	56.17	54.30	54.00
		ae-based	53.06	54.00	53.83	54.83	**54.95**	**56.00**	56.24	56.17	57.14	57.00
		lae-based	**53.42**	**54.67**	**54.20**	**55.50**	54.51	55.50	**56.91**	**56.83**	**57.76**	**57.83**
gisette		linear	93.00	92.80	93.61	93.50	94.40	94.30	94.22	94.10	94.50	94.40
		RBF	93.66	93.60	93.32	93.30	94.91	94.90	94.61	94.60	94.91	94.90
		ae-based	93.99	93.90	95.02	95.00	95.50	95.50	95.21	95.13	96.04	96.00
		lae-based	**94.19**	**94.10**	**95.59**	**95.60**	**95.91**	**95.90**	**96.33**	**96.30**	**96.44**	**96.40**
text		linear	66.67	66.33	69.03	68.00	70.75	71.33	70.92	72.67	73.54	74.33
		RBF	65.99	66.33	64.73	67.67	69.53	71.67	69.77	74.00	73.54	74.33
		ae-based	67.52	66.33	68.40	67.67	70.95	71.33	71.13	72.67	73.72	74.33
		lae-based	**68.39**	**67.33**	**69.62**	**70.33**	**71.23**	**72.00**	**71.58**	**73.00**	**74.40**	**75.00**

TABLE 3.7: Test results (%) on ESR.

LapRLS			5%		10%		15%		20%		25%	
data	Graph	kernel	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
1vs2	kNN+ Gaussian	linear	51.78	54.46	52.41	59.13	56.19	61.52	56.30	61.20	55.84	60.97
		RBF	74.82	76.96	82.22	81.85	83.87	83.70	83.21	82.93	83.82	83.59
		ae-based	81.70	83.36	87.93	88.66	90.06	90.10	89.81	89.97	90.20	90.33
	proposed		**87.30**	**87.50**	**90.02**	**90.37**	**91.99**	**92.12**	**90.64**	**90.92**	**92.86**	**92.84**
1vs3	kNN+ Gaussian	linear	51.65	53.80	53.64	59.78	50.38	50.76	57.46	62.83	54.73	60.98
		RBF	74.35	74.35	82.49	81.63	82.07	81.20	83.32	82.50	82.90	82.07
		ae-based	84.79	85.59	90.49	90.49	89.33	89.82	91.37	91.37	91.03	91.22
	proposed		**88.38**	**88.32**	**92.44**	**92.48**	**92.45**	**92.49**	**93.61**	**93.65**	**92.57**	**92.88**
1vs4	kNN+ Gaussian	linear	49.89	53.70	49.15	54.57	54.44	58.15	54.98	58.70	53.93	58.59
		RBF	82.75	82.83	87.51	87.07	89.01	88.80	89.29	89.02	89.50	89.24
		ae-based	91.14	91.33	95.18	95.33	95.46	95.60	95.32	95.49	95.69	95.80
	proposed		**92.93**	**93.32**	**95.76**	**95.78**	**96.76**	**96.82**	**96.05**	**96.16**	**96.51**	**96.58**
1vs5	kNN+ Gaussian	linear	53.49	57.28	52.79	59.57	55.82	61.63	58.18	63.59	56.35	62.61
		RBF	79.42	79.89	84.16	82.93	85.77	85.22	86.11	85.87	86.40	85.76
		ae-based	94.68	94.77	95.50	95.67	96.31	96.36	97.14	97.16	96.77	96.79
	proposed		**97.62**	**97.62**	**97.15**	**97.22**	**97.83**	**97.83**	**97.41**	**97.45**	**97.74**	**97.76**

3.5.4 Compared with different kernels

All these four kernel machines in Table 3.6 are different from each other in the kernel level while sharing the graph Laplacian which is constructed using k-NN matching and Gaussian weights. Linear and RBF are commonly used kernels, autoencoder (ae-based) stands for an unsupervised gating mechanism to compose the kernel while labeled autoencoder (lae-based) standing for a semi-supervised one to compose the kernel as describe in Section 3.3.

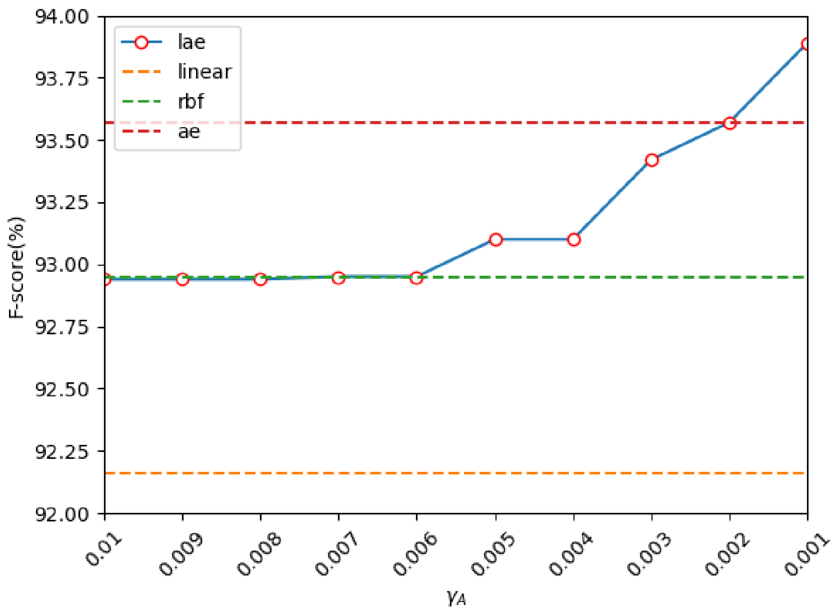

FIGURE 3.4: Analysis the effect of γ_A on splice 25% while other hyper-parameters are fixed, linear, RBF and ae are used as references.

Firstly, it is obvious that our proposed method achieves better performance among kernel machines in all situations. For datasets with low dimensional features, our kernel works better with a tiny number of labels, and as the number of labels increases, our advantage decreases while for high dimensional datasets, our kernel outperforms steadily. Secondly, we can observe that the label guidance introduced in the gating mechanism actually improves the performance almost in all the situation under all datasets. It verifies that the data manifold captured from both labeled and unlabeled data in a semi-supervised way is better than the one captured from only unlabeled data in an unsupervised manner. When the label guidance leads to an overfitting problem like splice

25%, we notice the $\gamma_A = 0.01$ reaches the minimum of the search range. We tend to assume it is a low noise dataset and if the complexity of Θ can be improved in the second step of our method, the performance would be better. So we fixed all the other hyper-parameters just analysis the effect of γ_A. The results showed in Fig.3.4 verify our assumption, if we can extend the search range of γ_A, the overfitting problem can be solved in the second step in our method.

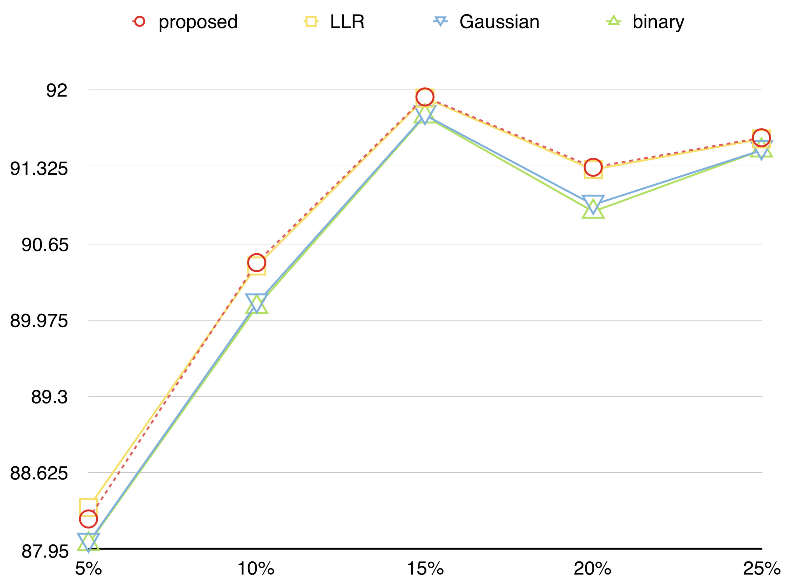

FIGURE 3.5: F-score of different graph constructions in LapRLS on ESR dataset (Class 1 vs 2 as an example)

3.5.5 Compared with different graphs

The ESR dataset has five classes, each of which contains 2300 instances, and only instances in Class 1 have epileptic seizure while others not, we focus on Class 1 vs other classes. We found that the proposed method is particularly prominent on the ESR dataset as shown in Table.3.7, so we further explore the effectiveness of the graph construction in our proposed method on ESR as an example.

We use binary weight, Gaussian weight and locally linear reconstruction (LLR) [94] as comparative methods in edge weighting where the k-NN is used for graph sparsification. On the other hand, in our proposed method both of the graph sparsification and edge weighting are based on the quasi-linear kernel as described in Subsection 4.3.

Fig.3.5 shows the effect of different methods of graph construction in LapRLS on the classification performance under the same quasi-linear kernel. We found that the performance of binary and Gaussian almost overlap, and both LLR and our method perform better. LLR performs better because of its local manifold approximation, and it's reasonable to believe the success of our method is due to the data manifold captured in the gating mechanism. Although the two can have a similar effect, LLR is more time-consuming.

3.6 Summary

In this chapter, we have proposed a semi-supervised classifier based on a piecewise linear regression model which aims to fully utilize the data manifold learned from both training instances and limited labels. We introduce a gated linear network to realize the piecewise linear model which has two step. In the first step, we pre-train a semi-supervised gating mechanism to transform the break-point estimation of a piecewise linear model into the generation of gate control signals, which is guided by the data manifold learned from both labeled and unlabeled data, so as to partition. In the second step, the piecewise linear model is transformed into a linear regression form where the known regression vector is composed with the gate control signals. We optimize the linear parameters globally by a LapRLS algorithm where the kernel is defined as inner product of the regression vectors. Moreover, we use the kernel function as a similarity measurement to construct the graph in LapRLS. In this way, the data manifold is ingeniously incorporated both into the kernel and the graph Laplacian in the LapRLS.

Chapter 4

Semi-Supervised Classification of Parasite Images

4.1 Background

[1]According to the World Health Organization report, infectious and parasitic diseases are the second leading cause of death worldwide [95]. Plasmodium, Babesia, and Toxoplasma are amongst the most prevalent and morbidity-causing parasites in humans and animals worldwide [96]. Plasmodium is the causative agent of malaria, which impacts over 200 million individuals and kills over 300,000 children annually [97]. Babesiosis caused by Babesia is a disease with many clinical features that are similar to those of malaria [98]. Toxoplasma is estimated to infect one-third of the world's population, life-threatening in the pregnancy and the immunocompromised [99]. Measuring parasite infection by direct microscopy observation remains relatively widespread as a point-of-care diagnostic in clinical and epidemiological settings [100]. Affordable and accurate diagnostic testing on microscopic images could reduce the risk of illness and death significantly.

[1]This chapter is mainly extended from the Journal paper:Y. Ren, H. Jiang, H. Zhu, Y. Tian and J. Hu, "A Semi-Supervised Classification Method of Parasites using Contrastive Learning", *IEEJ Trans. on Electrical and Electronic Engineering*, Vol.17, No.3, March, 2022. (9 pages)

Recently, deep learning techniques have become a popular choice in both computer vision and the medical imaging community [101]. For the recognition of the parasites and the Erythrocyte, which is often overlapped with the parasites, deep learning has obtained impressive results. A deep learning model using hand-craft features in morphology to classify healthy and abnormal Erythrocytes [102]. Well-known Convolutional Neural Networks (CNNs), including Inception v3 [103], LeNet, AlexNet, and GoogLeNet [104], are used to identify Plasmodium parasites. A solution for multiple parasites and Erythrocytes is given based on Toxoplasma, Babesia, Plasmodium, and Erythrocytes are variant in morphology under microscopy [66].

However, the outlined methods have been designed in a prerequisite that all the training data is labeled. In practical clinical scenarios, human annotations are expensive and time-consuming to obtain due to the shortage of specialists in diagnostic imaging despite the increasing spread of imaging equipment [105]. The lack of the labeled data motivates the study of methods trained with limited supervision, such as semi-supervised classification [3], weakly supervised learning [4], and unsupervised domain adaptation [6]. This chapter focus on semi-supervised classification (SSC), which formulates the separation boundary by leveraging a large amount of unlabeled data in addition to a small amount of labeled data. Furthermore, there are two challenges in microscopic parasite image classification. On the one hand, the semantic objects, namely salient structures, are fuzzier and more complex than real-world images, leading to microscopic images are not as distinguishable as macroscopic images. On the other hand, the insignificant textures, like image background staining, lightness, or contrast level, vary much in samples from different clinical scenarios. At the same time, the available training data [106] is of a similar pattern in each category, which may lead to poor generalization in real-world applications.

In this chapter, we propose a semi-supervised classification method for three parasites and Erythrocytes microscopic images. It contains a feature extractor trained by contrastive learning and a classifier optimized by Laplacian Support Vector Machine

(LapSVM). Contrastive learning aims to map similar data close together and map dissimilar data further away in the embedding space. By defining the similarity and dissimilarity distribution, the desired invariant/covariant properties of the learned representation are specified [107]. LapSVM extends the support vector machine, a supervised kernel machine with marginal maximum, by incorporating a manifold regularizer estimated using the graph Laplacian associated with all the training data [51].

For the feature extractor, we aim to learn structure-enhanced and texture-invariant representations using contrastive learning [108]. It is difficult to classify the microscopic images directly due to the fuzzy structure, especially when the labeled data is minimal. Therefore, we introduce the macroscopic images with similar and clear semantic information, which are much easier to classify, and we connect microscopic images and macroscopic images by similarity to enhance the structure at the representation level. In addition, we introduce variant appearance transformations to eliminate the insignificant texture at the representation level. For the classifier, we realize a piecewise linear model approximating the nonlinear separation boundary using the gated linear network, in which the gate control signals are for partitioning [109]. Given the representations and the gate control signals generated from the learned feature extractor, the gated linear network is recast into a linear regression form. The linear parameters are optimized by LapSVM using a kernel function composed of the representations and the gate control signals. In summary, the proposed semi-supervised method tackles the structure and texture challenges and achieves affordable and accurate parasite classification.

This chapter is organized as follows. Section 4.2 states the data and gives an overview of the proposed method. Section 4.3 describes the data pre-processing and the feature extractor trained using contrastive learning. Section 4.4 shows the classifier optimized using LapSVM. Section 4.5 presents the simulation results of a series of experiments to validate the effectiveness of our proposed method. Finally, the conclusions are drawn in Section 4.6.

4.2 Methodology

4.2.1 Materials

Microscopic images including 5758 Plasmodium, 5741 Toxoplasma, 5878 Babesia, and 6981 Erythrocytes are used as Micro data [106]. Following the knowledge from parasitologists that Plasmodium is ring-shaped, Toxoplasma is banana-shaped, Babesia is double-pears-shaped, and Erythrocyte is apple-shaped [66]. We photo 500 macroscopic images each of ring, banana, double-pears, and apple as Macro data [110], which best match the Micro objects of interest. Macro data shares the same label with its corresponding Micro data.

The split of Micro data for the training and testing sets is retained, so the testing set contains 1000 Micro data for each category. To evaluate the generalization ability, we transform the testing set to a random color distortion version (randomly changes the hue, lightness, and saturation of each test image).

We suppose that only 50 Micro data in each category are labeled, about 1% of the whole data set, in a practical clinical scenario. The training set contains 2000 Macro data $\mathcal{X}_S = \{(x_s, y_s)\}_{s=1}^{S}$ which is fully labeled with $y_s \in \{0, 1, 2, 3\}$ as source data, 200 labeled Micro data $\mathcal{X}_T = \{(x_t, y_t)\}_{t=1}^{T}$ with $y_t \in \{0, 1, 2, 3\}$ as target data, and 20158 unlabeled Micro data $X_U = \{x_u\}_{u=1}^{U}$. Samples from the training set are illustrated in Fig.4.1.

4.2.2 Framework

As illustrated in Fig. 4.2(1)(2), for the feature extractor, firstly, Macro data is transformed to its adaptive version with the style of corresponding Micro data, as the data pre-processing. Then a deep CNN G is trained as the feature extractor, with a combination of supervised loss only for the labeled data and unsupervised loss for both the labeled and the unlabeled data. In addition to representation z, the trained G can generate binary gate control signals $g_j(z)$ [90].

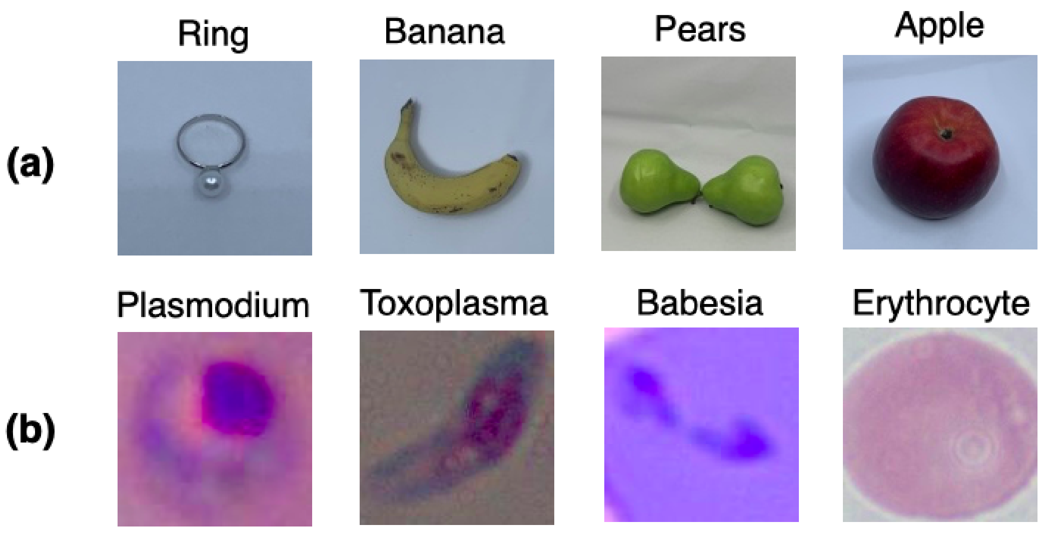

FIGURE 4.1: Samples from the training set.

As illustrated in Fig. 4.2(3), given the representations z and gate control signals $g_j(z)$, a gated linear network is adopt as the classifier to realize a piecewise linear separation boundary [109]

$$f(z) = \sum_{j=1}^{M} (\Omega_j^T z + b_j) \cdot g_j(z) + b \tag{4.1}$$

where $\Omega_j^T z + b_j (j = 1, \cdots, M)$ are a set of linear base models. Gate control signal $g_j(z)$ determines whether the j-th linear base model plays a role; $g_j(z) = 1$ indicates that the j-th linear base model does play a role while $g_j(z) = 0$ meaning it does not. Different gating sequences $\mathbf{g}(z) = [g_1(z), \cdots, g_M(z)]$, which mean different combinations of linear base models, correspond to different linear submodels, thus different partitions. The linear parameters Ω_j, b_j of linear base models and the bias b are estimated implicitly by applying LapSVM.

4.3 Deep CNN Feature Extractor

4.3.1 Data pre-processing

Directly using fully labeled Macro data and weakly labeled Micro data together in the training phase may lead to bad performance due to the different data distributions,

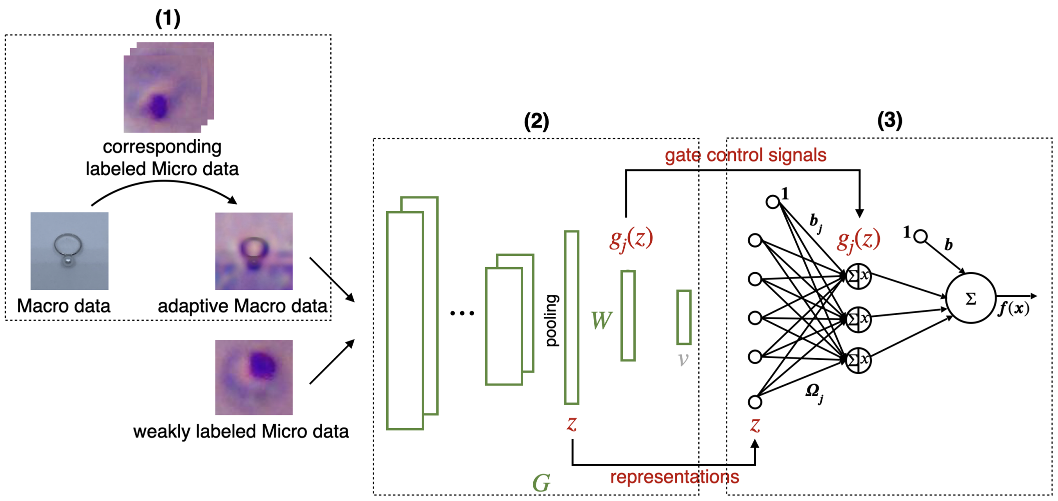

FIGURE 4.2: Framework of proposed method.

known as “domain shift” [111]. In this subsection, we adapt Macro data to appear as if drawn from the Micro domain to realize the visual alignment, as shown in Fig. 4.2(1).

We take rendering the ring image with Plasmodium style as an example. Style features of overall labeled Plasmodium data should be separated as Plasmodium style. Content features of the object in the ring image should be extracted and recombined with Plasmodium style to produce an adaptive ring image.

Firstly a pre-trained deep CNN with fixed parameters is used to extract and store features. For every ring image $(x_s, 0)$, the content features are from itself, and the style features are from all the labeled Plasmodium images $\mathcal{X}_t = \{(x_t, 0)\}_{t=1}^{p}$. The content features on CNN layer l could be stored in a matrix $F^l \in \mathbb{R}^{N_l \times H_l \times W_l}$, where N_l is the number of distinct filters, H_l and W_l denotes the height and width of the feature map respectively. The style features on CNN layer l could be represented by Gram matrix $G^l \in \mathbb{R}^{N_l \times N_l}$, where $G^{l,ij} = F^{l,i} \odot F^{l,j}$ is the inner product between the vectorized feature maps i and j in layer l [112].

Secondly, the overall loss function

$$\ell_{pre} = J_{content}(x_a, x_s) + \lambda_s \cdot J_{style}(x_a, \mathcal{X}_t) \tag{4.2}$$

is minimized to update the image x_a iteratively until it simultaneously matches the content features of x_s and the style features of $\mathcal{X}_t$, where x_a can be initiated as a white noise

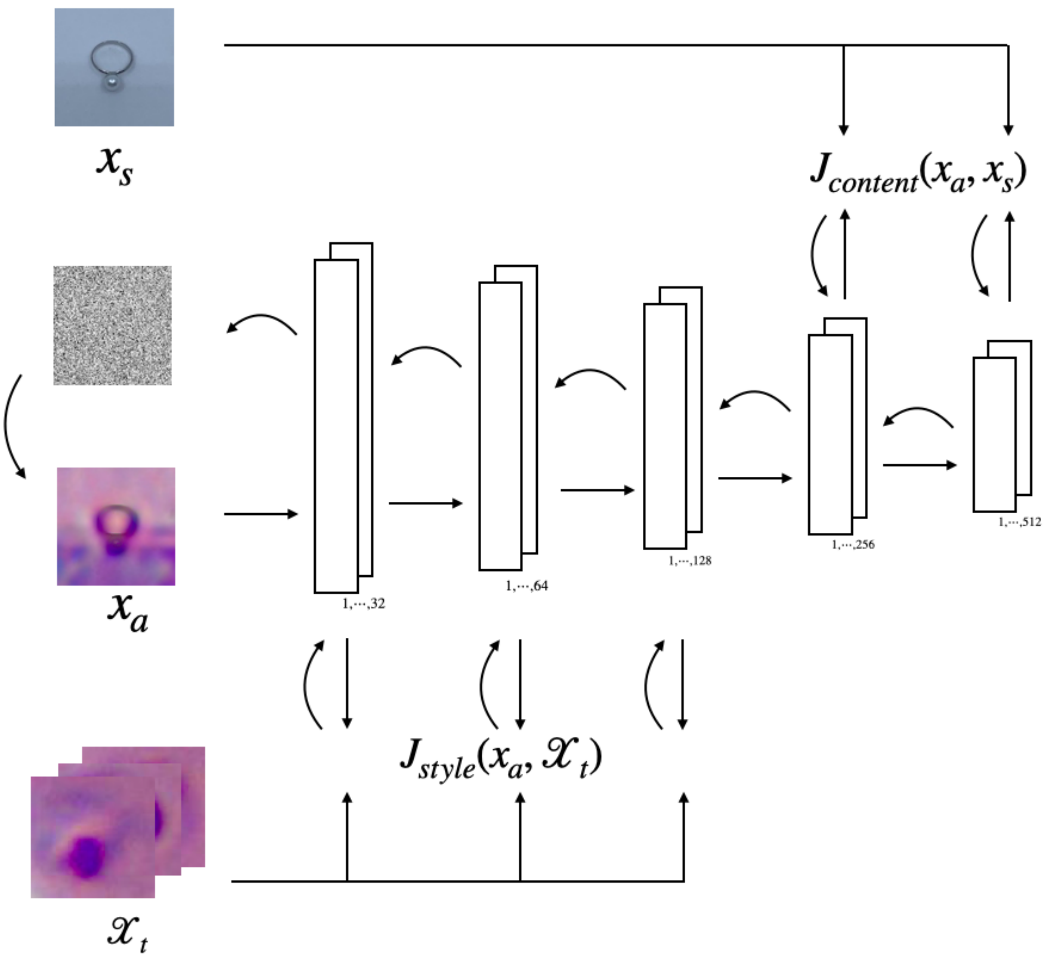

FIGURE 4.3: Rendering the Macro image with Micro style using a pre-trained deep CNN.

image andÂ λ_s is a coefficient.

Based on the fact that deeper convolutional layers respond to higher semantics [113], $J_{content}$ and J_{style} are defined as

$$\begin{aligned} J_{content}(x_a, x_s) &= \frac{1}{2}\sum_{l \in L_h} w^l \cdot (F_a^l - F_s^l)^2 \\ J_{style}(x_a, \mathcal{X}_t) &= \frac{1}{2}\sum_{l \in L_o} w^l \cdot (G_a^l - \bar{G}_t^l)^2 \end{aligned} \tag{4.3}$$

where L_h are the higher CNN layers which capture the high-level content in terms of objects, L_o are the lower CNN layers, w^l is the weight of layer l. Specifically, we use $\bar{G}_t^l = \frac{\sum_t G_t^l}{p}$ as the average style features in the Micro domain instead of using the style of a single Micro data.

With labeled Micro data, all the Macro data $\mathcal{X}_S = \{(x_s, y_s)_{s=1}^S\}$ is transferred to its adaptive version $\mathcal{X}_A = \{(x_a, y_s)_{s=1}^S\}$ with the corresponding Micro style.

4.3.2 Trained by contrastive learning

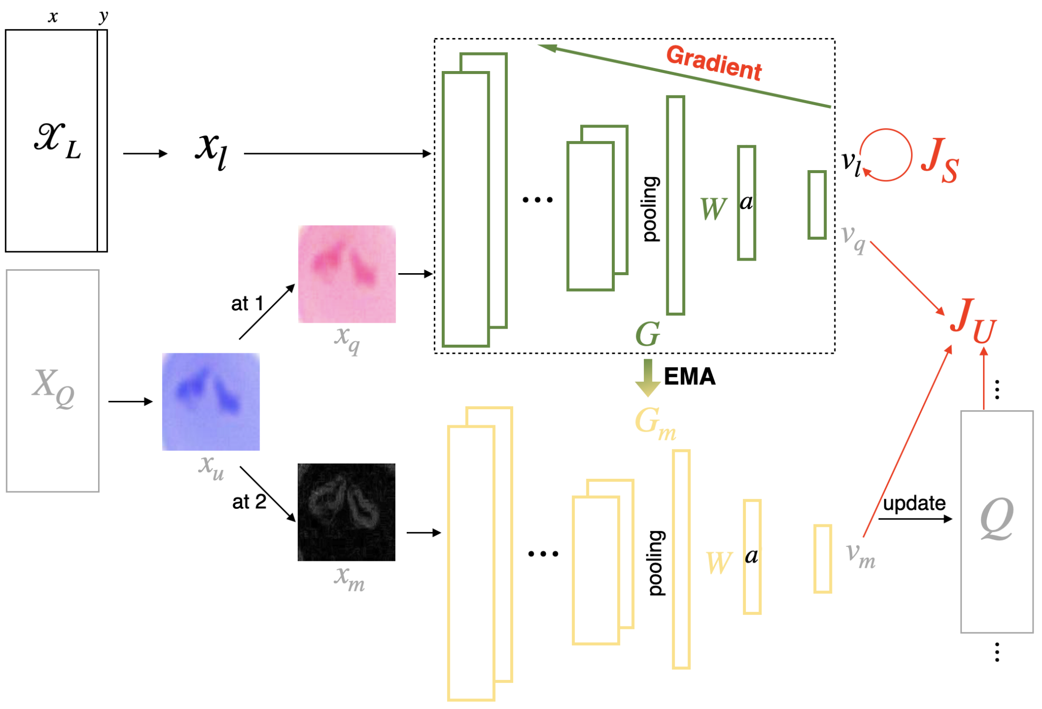

FIGURE 4.4: Train the feature extractor G using contrastive learning.

As illustrated in Fig. 4.2(2), the feature extractor denoted as $G(x;\Theta_g)$ can be conceptually decomposed to a CNN encoder $e(x;\theta_e) : \mathcal{X} \to \mathcal{Z}$ and a multilayer perceptron (MLP) signal generator $h(z;\theta_h) : \mathcal{Z} \to \mathcal{V}$, where z is the representation of the input x, and $g_j(z)$ are the gate control signals. The losses are calculated using the metric embeddings $\mathcal{V}$. All the labeled data is denoted as $\mathcal{X}_L = \mathcal{X}_T \cup \mathcal{X}_A$, and all the data except labels is denoted as $X = X_T \cup X_A \cup X_U$.

Fig.4.4 illustrates how to train the feature extractor. The update of Θ_g is by back-propagation on a combination of supervised loss J_S associated with $\mathcal{X}_L$ and unsupervised loss J_U associated with X

$$\ell_e = J_S + \lambda \cdot J_U \tag{4.4}$$

In addition to the online network G, a offline momentum network $G_m(x;\Theta_m)$ is used as a memory trick [114]. Θ_m is an exponential moving average (EMA) of Θ_g

$$\Theta_m \leftarrow \alpha \cdot \Theta_m + (1-\alpha)\cdot \Theta_g \tag{4.5}$$

In particular, to add a manual control on the flexibility of the piecewise linear model[62], following the idea of k-sparse strategy [88], we define the activation function of the hidden layer in the signal generator as

$$a(w_j^T z) = \begin{cases} w_j^T z, & j \in \Gamma = supp_k\{W^T z\} \\ 0, & j \notin \Gamma = supp_k\{W^T z\} \end{cases} \tag{4.6}$$

where the set $\Gamma = supp_k\{W^T z\}$ contains hidden units with top-$k\%$ activation values. After training, the generation of gate control signals is guided by the captured data manifold

$$g_j(z) = \begin{cases} 1, & j \in \Gamma = supp_k\{\hat{W}^T z\} \\ 0, & j \notin \Gamma = supp_k\{\hat{W}^T z\} \end{cases} \tag{4.7}$$

InfoNCE [115] is adopted as the form of contrastive loss function due to its efficiency and simplicity, with a well-grounded motivation from information theory. Considering similarity matching as a form of dictionary look-up, given query v, the InfoNCE is the negative log-likelihood

$$-\log p_v \tag{4.8}$$

where the likelihood is

$$p_v = \frac{\exp(v \cdot v^+/\sigma)}{\exp(v \cdot v^+/\sigma) + \sum_i \exp(v \cdot v_i^-/\sigma)} \tag{4.9}$$

and σ is a temperature parameter that controls the concentration level of the distribution [116]. The positive keys v^+ are sampled from similarity distribution, and the negative keys v_i^- are sampled from dissimilarity distribution.

Structure-enhanced and texture-invariant representations are learned for the downstream classification task by defining the positive and negative keys for all the training data as follows.

Structure enhancement in J_S

We introduce the Macro data and emphasize the connections between the labeled Micro data and its corresponding Macro data by contrasting.

Every query embedding v_t from Micro data, its positive keys are sampled from its corresponding adaptive Macro set. Its negative keys are the embeddings of all the other unrelated adaptive Macro data. And it is the same way for adaptive Macro data.

The supervised loss for $\mathcal{X}_L$ is

$$J_S = -\sum_{t\in\Omega_T} \log p_t - \sum_{a\in\Omega_A} \log p_a \tag{4.10}$$

where

$$\begin{aligned} p_t &= \sum_{k^+\in\Omega_a} \frac{\exp(v_t \cdot v_{k^+}/\sigma)}{\exp(v_t \cdot v_{k^+}/\sigma) + \sum_{k^-\in\mathbf{C}_{\Omega_A}^{\Omega_a}} \exp(v_t \cdot v_{k^-}/\sigma)} \\ p_a &= \sum_{k^+\in\Omega_t} \frac{\exp(v_a \cdot v_{k^+}/\sigma)}{\exp(v_a \cdot v_{k^+}/\sigma) + \sum_{k^-\in\mathbf{C}_{\Omega_T}^{\Omega_t}} \exp(v_a \cdot v_{k^-}/\sigma)} \end{aligned} \tag{4.11}$$

t is the index of Micro data, and Ω_a are all the indexes of its corresponding adaptive Macro data. a is the index of adaptive Macro data, and Ω_t are all the indexes of its corresponding Micro data. Ω_T are all the indexes of Micro data, and Ω_A are all the indexes of adaptive Macro data.

Texture elimination in J_U

We encourage a consistent representation of the same input under different appearance transformations.

Two kinds of appearance transformations are introduced to simulate variant textures, color distortion (which randomly changes the hue, lightness, and saturation of an image) and flittering (Sobel, Scharr, Laplacian). Â For every x_u in X, it is randomly transformed to two different views x_q, x_m, and we define the positive pair $p^+(v_q, v_m)$. Naturally, all the others are the negative keys.

Since InfoNCE benefits from more negative keys, we apply an offline momentum network $G_m(x;\Theta_m)$ and maintain a dynamic dictionary Q on metric embeddings v_m as a queue [114]. Introducing Q decouples the dictionary size from the mini-batch size, which can be much larger than the mini-batch size. Θ_m is the EMA copy of Θ, which smooths the learning dynamics.

The unsupervised loss for X is

$$J_U = -\sum_u \log p_u \tag{4.12}$$

where

$$p_u = \frac{\exp(v_q \cdot v_m/\sigma)}{\exp(v_q \cdot v_m/\sigma) + \sum_i \exp(v_q \cdot Q_i/\sigma)} \tag{4.13}$$

$v_q = G(x_q;\Theta_g)$, $v_m = G_m(x_m;\Theta_m)$, and i is the index of the metric embeddings in the maintained queue under the current state.

4.4 LapSVM Classifier

As illustrated in Fig. 4.2(3), given the representations z and gate control signals $g_j(z)$, the known variables and unknown linear parameters of the gated linear network can be safely separated by introducing $\phi(z)$ and Θ:

$$\phi(z) = [\mathbf{g}^T(z) \otimes [1, z^T]]^T \tag{4.14}$$

$$\Theta = [b_1, \Omega_1^T, \cdots, b_M, \Omega_M^T]^T \tag{4.15}$$

where the symbol $\otimes$ denotes Kronecker product. Eq.(4.1) is recast into a linear regression form

$$f(z) = \phi^T(z)\Theta + b \tag{4.16}$$

It is equivalent to the fact that $\phi(z)$ defined as Eq.(4.14) maps the input to a finite high dimensional spanned feature space. Then Eq.(4.16) is expressed as a kernel machine

$$f(z) = \mathbf{k}^T(z)\alpha + b \tag{4.17}$$

where $\mathbf{k}(z)$ is the similarity vector of all the training data and z in the spanned space, $\mathbf{k}(z) = \Phi_N\phi(z) = [k(z,z_1),\cdots,k(z,z_N)]^T$, $k(z_i,z_j) = \phi^T(z_i)\phi(z_j)$, $\Phi_N = [\phi(z_1),\cdots,\phi(z_N)]^T$, and α is an N dimensional coefficient. $k(z_i,z_j)$ is called quasi-linear kernel[60]

$$\begin{aligned} k(z_i,z_j) &= \phi^T(z_i)\phi(z_j) \\ &= (1 + z_i^T z_j)\mathbf{g}^T(z_i)\mathbf{g}(z_j) \end{aligned} \tag{4.18}$$

Estimation of Θ is converted to estimation of α to take advantage of the kernel trick. In this way, there is no need to explicitly calculate the representation in the spanned feature space $\phi(z)$. Instead, the kernel is composed of the representation and gate control signals.

In the context of SSC, LapSVM formulated by

$$\min_{f\in\mathcal{H}_k} \sum_{i=1}^{L} \max(1 - y_i f(z_i), 0) + \frac{\gamma_A}{2}\|f\|_A^2 + \frac{\gamma_I}{2}\|f\|_I^2 \tag{4.19}$$

is considered to be a good choice to solve Eq.(4.17). γ_A controls the complexity of the function in the ambient space and γ_I controls the complexity of the function in the intrinsic geometry of $\mathcal{P}_X$. By using the graph Laplacian regularizer $F^T LF$, where $F = K\alpha + \mathbf{1}b$, we have

$$\begin{aligned} &\min_{\alpha,b} \frac{1}{2}\sum_{i=1}^{L} \max(1 - y_i(\mathbf{k}^T(z_i)\alpha + b), 0)^2 \\ &+ \frac{\gamma_A}{2}\alpha^T K\alpha + \frac{\gamma_I}{2}(\alpha^T K + \mathbf{1}^T b)\mathbf{L}(K\alpha + \mathbf{1}b) \end{aligned} \tag{4.20}$$

where $K = \Phi_N \Phi_N^T = [\mathbf{k}(z_1), \cdots, \mathbf{k}(z_N)]^T$, $\mathbf{L}$ is the graph Laplacian built from K [109], and $\mathbf{1}$ is the vector whose all entries equal to 1.

We train it in primal for lower computational complexity and shorter training time [117]. Multiple binary one-vs-others classifiers are trained

$$\begin{aligned} y &= \max[f(z)] \\ f(z) &= [f_1(z), \cdots, f_C(z)]^T \\ f_i(z) &= \mathbf{k}^T(z)\alpha_i + b_i \end{aligned} \tag{4.21}$$

and the final result is the maximum among all the classifiers' result.

4.5 Numerical Experiments

4.5.1 Evaluation metrics

We use *Accuracy* (AC), *F-score* (F1), and *Jaccard index* (JA) to evaluate the performance. *Accuracy* (AC) and *F-score* (F1) are defined as (2.28), *Jaccard index* (JA) is defined as

$$JA = \frac{TP}{TP + FP + FN} \tag{4.22}$$

where TP, FP, TN, FN are true positive, false positive, true negative, and false negative, respectively.

4.5.2 Technical details

For the data pre-processing, we adopt pre-trained DenseNet-121[118] as the CNN backbone. In particular, we only include the five convolutional layers in the set, i.e., $L = \{conv1, dense1c, dense2d, dense3f, dense4c\}$ to extract and store features, as the features of these layers generally have the highest capability in each scale, where $L_h = \{dense3f, dense4c\}$ and $L_o = \{conv1, dense1c, dense2d\}$. The weights w^l of layers are generally determined on the visual appearance of adaptive Macro data. When optimizing Eq.(4.2), we empirically set a small weight of $\lambda_s = 10^{-3}$, since our ultimate goal is to semantically classify the parasites, which requires preserving the semantic content precisely.

For our deep CNN feature extractor and other deep CNN methods, DenseNet-121 architecture is employed as the CNN backbone for a fair comparison. Its last layer (after global average pooling) has a fixed-dimensional output (1024-D). The following MLP leads to a 256-D hidden layer, then to a 128-D v in our feature extractor and a 4-D softmax layer in other classification methods. The sparsity of the activation function of the hidden layer is set $k = 20$. For the likelihood calculation, we set temperature $\sigma = 0.08$. When calculate the EMA in Eq.(4.5), we follow the empirical experience and set $\alpha = 0.999$. The batch size is 256, and the queue size is 4096. The network was trained by the stochastic gradient descent (SGD) algorithm with a learning rate of 0.0001 and a momentum of 0.9. All the experiments are trained for 500 iterations, and three repeat training are performed.

For the LapSVM classifier, all the parameters are chosen from $\{0.01, 0.1, 1, 10, 100\}$ by a three-fold cross validation.

4.5.3 Effetiveness of the proposed method

With 200 Macro data, 200 labeled Micro data, and 20158 unlabeled Micro data for the training phase, the proposed method achieves an accuracy of 95.10%. It is comparable to the reported accuracy of 95.70% [66] in a full supervised scenario, where there are 2118 Macro data, and all the Micro data are labeled.

A receiver operating characteristic (ROC) curve is created to visualize the classification performance of our proposed method for all four classes, and an overall area under the curve (AUC) value is computed to summarize the diagnostic performance. As illustrated in Fig.4.5, the overall ROC curve of the proposed method achieves an AUC of 0.95.

The confusion matrix is depicted in Fig.4.6. We can see that the proposed method gains impressive performance among parasites. Most confusions in Erythrocyte are false positive of the parasites, which is acceptable.

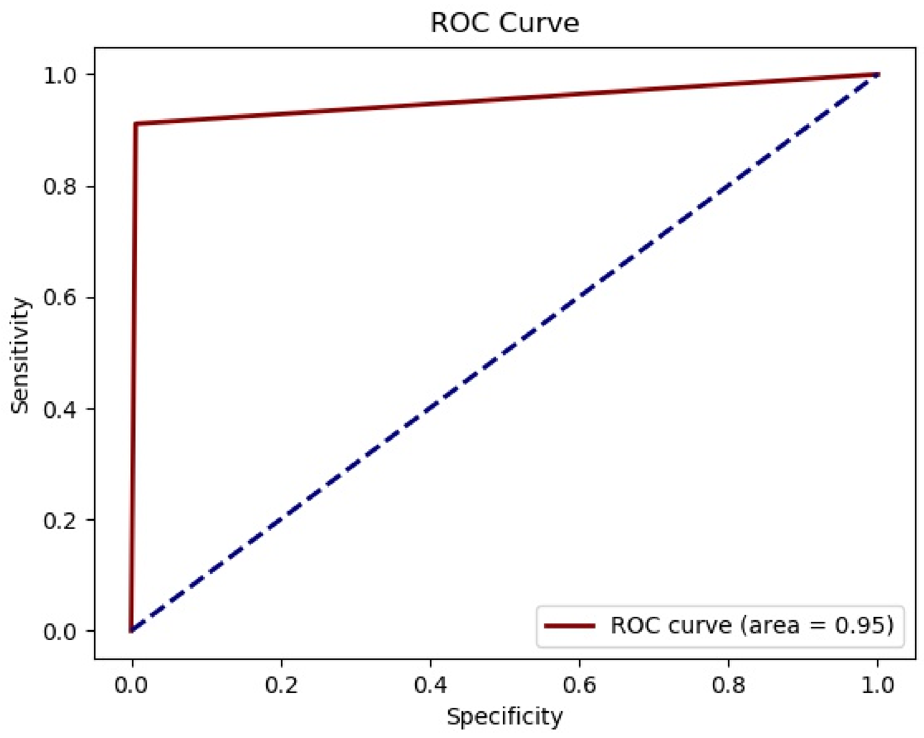

FIGURE 4.5: Overall receiver operating characteristic (ROC) curve.

4.5.4 Comparison among kernels

TABLE 4.1: Ablation of kernels in LapSVM.

Kernel of LapSVM	Evaluation		
	AC	F1	JA
linear	93.93	95.81	88.55
RBF	94.45	96.17	89.47
quasi-linear	**95.10**	**96.63**	**90.65**

From the aspect of LapSVM, the proposed method uses a quasi-linear kernel which is composed of the representation z and gate control signals $g_j(z)$ as Eq.(4.18). Linear kernel $k(z_i, z_j) = z_i^T z_j$ and radial basis function(RBF) kernel $k(z_i, z_j) = \exp(-\gamma \|z_i - z_j\|_2^2)$ are the most commonly used linear and non-linear kernel respectively. The performance of

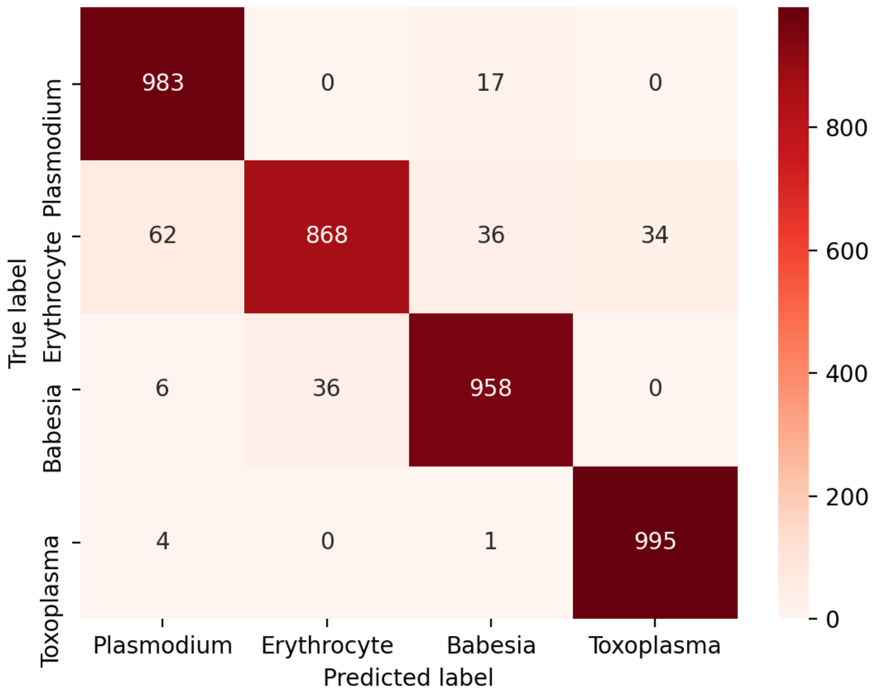

FIGURE 4.6: Confusion matrix of the proposed method.

LapSVM with three different kernels are listed in Table.4.1. Non-linear kernels have an advantage against the linear kernel, and the proposed method achieves the best performance.

We are sure the representations tend to be non-linear separable. And the advantage of the proposed method is due to the captured data manifold incorporated into the quasi-linear kernel.

4.5.5 Comparison between classifiers

TABLE 4.2: Ablation of classifiers.

Classifier	Evaluation		
	AC	F1	JA
MLP	94.90	96.48	90.28
ours	**95.10**	**96.63**	**90.65**

TABLE 4.3: Ablation of different labels.

Micro Data (l/u)	50*4/20158											
Macro Data (l)	50*4			100*4			200*4			500*4		
	AC	F1	JA	AC	F1	JA	AC	F1	JA	AC	F1	JA
super	87.08	90.74	77.40	89.50	92.58	81.29	91.68	94.17	84.66	93.93	95.74	88.45
MT	92.93	95.13	86.94	93.87	95.80	88.60	94.70	96.36	89.89	95.55	96.93	91.44
VAT	94.43	96.16	89.40	94.63	96.33	89.92	94.97	96.57	90.40	96.23	97.41	92.70
ours	**95.10**	**96.63**	**90.65**	**95.25**	**96.76**	**91.03**	**95.45**	**96.90**	**91.28**	**96.53**	**97.62**	**93.27**

From the aspect of classifier optimization, given the representation z, the simplest way is using the labeled data to train an MLP as the classifier [108]. We replace the metric embedding $\mathcal{V}$ in the signal generator with a softmax layer with four units as the MLP to make a comparison. As illustrated in Table.4.2, the proposed method achieves better results.

So it does help to use the unlabeled data in the optimization of the classifier.

4.5.6 Comparison with state-of-the-art semi-supervised methods

It is much easier to collect real-world images naturally labeled than annotated microscopic parasite images. We compare the proposed method with two state-of-the-art SSC methods under different numbers of Macro data. Mean Teacher (MT) [85] and Virtual Adversarial Training (VAT) [58] belong to consistency regularization stragety [84], which enforces that realistic perturbations of input should not significantly change the output. MT used the same EMA strategy as our proposed method from a student network to a teacher network with noise applying within their computation. VAT directly approximated a tiny perturbation to add to input which would most significantly affect the output of the prediction function, instead of relying on the built-in stochasticity. Quantitative results are in Table.4.3, and we draw the AC score in Fig.4.7.

As listed in Table.4.3, compared with other SSC methods, the proposed method achieves the most remarkable improvement over the supervised baseline, especially when there are only 200 labeled Macro data (+8.02/87.08). And the proposed method achieves the best performance under all the training sets.

Contrastive learning can be seen as an extension of consistency regularization, which considers the comparison between different inputs and encourages consistency of the same input. The results verify the effectiveness of contrastive learning as the extension of consistency regularization.

As shown in Fig.4.7, all these three semi-supervised methods perform better than the supervised baseline consistently, demonstrating that all the SSC methods effectively

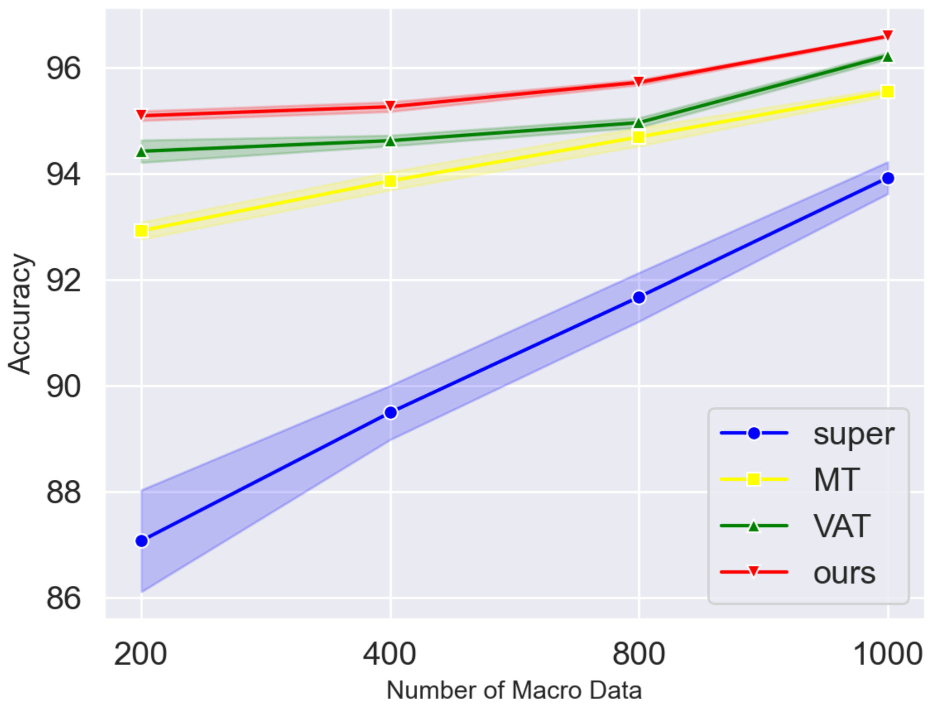

FIGURE 4.7: Accuracy with different number of Macro data.

utilize the unlabeled data and bring performance gains. As expected, the performances of all the methods increase when more labeled training data are available, and the gap between supervised baseline and semi-supervised methods narrows as more labeled training data is available.

Furthermore, it is worth noting that the proposed method has the lowest variance even with the least labeled data. The proposed method only uses deep CNN to extract representations, generate gate control signals, and then optimize the classifier by LapSVM. It compensates for the shortcomings of the end-to-end classification to some extent by improving stability.

4.5.7 Effectiveness of introducing Macro data

When analyzing how introducing Macro data help the final performance, both the supervised baseline and the proposed method use an MLP classifier for a fair comparison. As listed in Table.4.4, the first three rows are the results of the supervised baseline, and

TABLE 4.4: Ablation of Macro data introducing.

Training Data		Method	Evaluation (%)		
Marco	Micro		AC	F1	JA
l	l/u				
0	50*4/0	super	84.80	89.05	73.86
50*4 (o)	50*4/0		86.88	90.59	77.10
50*4 (a)	50*4/0		87.08	90.74	77.40
0	50*4/20158	ours	91.98	94.43	85.36
50*4 (o)	50*4/20158		92.75	95.00	86.65
50*4 (a)	50*4/20158		94.90	96.48	90.28

it helps a lot when introducing Macro data even in its original version (+2.08/84.80). However, the results of the semi-supervision scenario in the last three rows tell a different story. There is a tiny improvement when introducing original Marco data (+0.77/91.98), while a significant improvement when the Macro data is transformed to its adaptive version (+2.15/92.75).

These results conform to our assumption that 'domain shift' is remarkable in a semi-supervision scenario since there are a large amount of unlabeled data being trained together with the labeled data.

4.5.8 Effectiveness of contrastive loss

TABLE 4.5: Ablation of losses.

Training Data		Loss Setting	Evaluation		
Marco	Micro		AC	F1	JA
l	l/u				
50*4 (a)	50*4/0	CE J_S	87.08	90.74	77.40
		CON J_S	87.42	91.01	77.96
	50*4/20158	CE J_S + CON J_U	94.40	96.17	89.57
		ours	94.90	96.48	90.28

We further analyze the effectiveness of contrastive loss against the most commonly used cross-entropy loss. To have a fair comparison, the loss function is the only variable for the training data, and the results are listed in Table.4.5. The contrastive loss benefits both the supervised and semi-supervised methods.

We believe the contrastive loss connected the adaptive Macro data and the labeled Micro data by a direct comparison instead of simply using shared labels.

4.6 Summary

This chapter presents a novel and effective semi-supervised classification method for three parasites and Erythrocytes. We introduce real-world images and different appearance transformations to train the feature extractor using contrastive learning and optimize the gated linear network using LapSVM to realize a piecewise linear separation boundary. In this way, a large amount of unlabeled data and the limited labeled data are leveraged to tackle the structure and texture challenges. As a result, affordable and accurate diagnostic testing on microscopic parasite images is achieved.

Chapter 5

Conclusions

5.1 Conclusion

In this dissertation, we design three semi-supervised learning algorithms based on Laplacian kernel machines using the two-step modeling method: : 1) a Laplacian SVM based semi-supervised classifier using multi-local linear model; 2) a semi-supervised classifier based on piecewise linear model using gated linear network; 3) a semi-supervised classification method of parasites, which contains a semi-supervised feature extractor trained by contrastive learning and a semi-supervised classifier optimized by LapSVM. All of them fully leverage labeled and unlabeled data to achieve accurate classification performance on testing set.

Chapter 2 proposes a Laplacian SVM based semi-supervised classifier using multi-local linear model. The semi-supervised classifier is constructed in two steps. In the first step, by applying a pseudo-labeling approach, the input space is divided into multiple local linearly separable partitions along the potential separation boundary. A multi-local linear model is then built by interpolating multiple local linear models assigned to the partitions. In the second step, the multi-local linear model is formulated as a linear regression form with a new regression vector containing the information of potential separation boundary. Then all the linear parameters are optimized globally by a LapSVM algorithm using a quasi-linear kernel which is defined as the inner product of the new regression vectors. Furthermore, the quasi-linear kernel and the pseudo labels are used to construct a label-guided graph.

Chapter 3 proposes a semi-supervised classifier based on piecewise linear model using gated linear network. The semi-supervised classifier is constructed in two steps. In the first step, we design a label-guided autoencoder-based semi-supervised gating mechanism to generate binary gate sequences. By using a gated linear network, the binary sequences realize partitioning of a piecewise linear model indirectly. In the second step, the piecewise linear model is formulated as a linear regression form, and the linear parameters are then optimized globally by a LapRLS algorithm with a quasi-linear kernel comprising the binary sequences. Moreover, the quasi-linear kernel is used as a better similarity function for the graph construction.

Chapter 4 proposes a semi-supervised classification method of microscopic parasite images, which contains a semi-supervised feature extractor trained by contrastive learning and a semi-supervised classifier optimized by LapSVM. First, for the deep CNN feature extractor, we introduce real-world images with similar and clear semantic information to enhance the structure at the representation level. In addition, we introduce variant appearance transformations to eliminate the texture at the representation level. Second, a gated linear network is adopted as the classifier to realize a piecewise linear separation boundary. The linear parameters are optimized globally by a LapSVM algorithm using a quasi-linear kernel composed of the representations and the binary sequences generated from the learned feature extractor.

Compared to Laplacian kernel machines with the general nonlinear kernel, general kernels are black-box models with a fixed form. This dissertation models the nonlinear separation boundary in an interpretable way and detects prior knowledge from labeled and unlabeled data, and composes a data-dependent kernel using the prior, namely, the quasi-linear kernel. And we further use the quasi-linear kernel to build the graph. As a result, the prior is ingeniously incorporated into the kernel and graph of Laplacian kernel machines.

Previous studies on the two-step modeling method compose the quasi-linear kernel in a supervised or unsupervised manner. This dissertation considers learning under limited supervision and models the nonlinear separation boundary by leveraging a small amount of labeled data and a large amount of unlabeled data. As a result, the quasi-linear kernel is composed in a semi-supervised manner.

5.2 Topics for future research

We propose three two-step semi-supervised classification methods based on Laplacian Kernel machines in this dissertation. It still has many works to continue from very different perspectives. We name a few possible extensions here.

- Generally, structured data needs massive data to train a DNN as a feature extractor, especially when lacking labels. It is worth improving the efficiency of structured data use and further balancing the performance and data requirements.
- Medical imaging faces the biggest problem of lacking data. Annotations are lack due to the shortage of specialists, and unlabeled data is lack due to the privacy limitation. One possible way is to gather all the limited data from all the sources. In this situation, the challenge of domain shift is unavoidable. The domain adaptation in Chapter 4 happens to be between the medical domain and real-world domain. It is worth exploiting more possibilities of domain adaptation.
- This dissertation discusses objective-level classification in a semi-supervised context. Segmentation in a semi-supervised context, which is pixel-level classification, is a direction of future work.
- We exploit the two-step modeling method in a semi-supervised context in this dissertation. Researches of the two-step modeling can be extended to other problems, such as missing data.

Bibliography

[1] Y. Anzai, *Pattern recognition and machine learning*. Elsevier, 2012.

[2] J. H. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*. Springer Open, 2017.

[3] J. E. V. Engelen and H. H. Hoos, “A survey on semi-supervised learning.” *Machine Learning*, vol. 109, no. 2, pp. 373–440, Febraury 2020.

[4] Z.-H. Zhou, “A brief introduction to weakly supervised learning.” *National science review*, vol. 5, no. 1, pp. 44–53, January 2018.

[5] B. Settles, “Active learning literature survey,” University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., January 2009.

[6] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation.” in *Proc. of International conference on machine learning (ICML’2015) (Lille)*, July 2015.

[7] M. Long, H. Zhu, J. Wang, and M. I. Jordan, “Unsupervised domain adaptation with residual transfer networks,” in *Proc. of 30th Conference on Neural Information Processing Systems (NIPS’2016) (Barcelona)*, December 2016.

[8] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, “A comprehensive survey on transfer learning,” in *Proc. of the IEEE (P IEEE’2020) (online)*, July 2020.

[9] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, October 2009.

[10] X. Zhu, "Semi-supervised learning literature survey." University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2008.

[11] T. Jebara, J. Wang, and S.-F. Chang, "Graph construction and b-matching for semi-supervised learning." in *Proc. of the 26th Annual International Conference on Machine Learning (ICML'2009) (Montreal)*, June 2009.

[12] W. Liu, J. Wang, and S.-F. Chang, "Robust and scalable graph-based semisupervised learning." in *Proc. of International Conference on Information Science and Technology (ICIST'2012) (Wuhan)*, March 2012.

[13] A. Subramanya and P. P. Talukdar, "Graph-based semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 4, pp. 1–125, April 2014.

[14] C. A. R. de Sousa, S. O. Rezende, and G. E. Batista, "Influence of graph construction on semi-supervised learning," in *Proc. of Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'2013) (Prague)*, September 2013.

[15] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proc. of the 42nd annual meeting on association for computational linguistics, association for computational linguistics (ACL'2004) (Barcelona)*, July 2004.

[16] T. Joachims, "Transductive learning via spectral graph partitioning," in *Proc. of the 20th International Conference on Machine Learning (ICML'2003) (Washington)*, August 2003.

[17] A. Blum, J. Lafferty, M. R. Rwebangira, and R. Reddy, "Semi-supervised learning using randomized mincuts," in *Proc. of the twenty-first international conference on Machine learning (ICML'2004) (Banff)*, July 2004.

[18] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods." in *Proc. of 33rd Annual Meeting of the Association for Computational Linguistics (ACL'1995) (Cambridge)*, June 1995.

[19] A. Blum and T. Mitchell, “Combining labeled and unlabeled data with co-training.” in *Proc. of the Eleventh Annual Conference on Computational Learning Theory (COLT’1998) (Madison)*, July 1998.

[20] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system.” in *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’2016) (San Francisco)*, August 2016.

[21] C. Rosenberg, M. Hebert, and H. Schneiderman, “Semi-supervised self-training of object detection models,” in *Proc. of the 7th IEEE workshop on applications of computer vision (WACV’2005) (Breckenridge)*, January 2005.

[22] I. Dópido, J. Li, P. R. Marpu, A. Plaza, J. M. B. Dias, and J. A. Benediktsson, “Semisupervised self-learning for hyperspectral image classification,” *IEEE transactions on geoscience and remote sensing*, vol. 51, no. 7, pp. 4032–4044, July 2013.

[23] J. Du, C. X. Ling, and Z.-H. Zhou, “When does cotraining work in real data?” *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 5, pp. 788–799, May 2010.

[24] M. Chen, K. Q. Weinberger, and Y. Chen, “Automatic feature decomposition for single view co-training,” in *Proc. of the 28th international conference on machine learning (ICML’2011) (Washington)*, June 2011.

[25] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, January 1997.

[26] Z.-H. Zhou, *Ensemble methods: foundations and algorithms.* Chapman and Hall/CRC, 2019.

[27] I. Guyon and A. Elisseeff, “An introduction to feature extraction.” in *Feature Extraction.* Springer, 2006, pp. 1–25.

[28] A. Goldberg, X. Zhu, A. Singh, Z. Xu, and R. Nowak, “Multi-manifold semi-supervised learning.” in *Proc. of 12th International Conference on Artificial Intelligence and Statistics (ICAIS’2009) (Clearwater Beach)*, April 2009.

[29] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, “Why does unsupervised pre-training help deep learning?” in *Proc. of the 13th International Conference on Artificial Intelligence and Statistics (ICAIS’2010) (Sardinia)*, May 2010.

[30] R. Sheikhpour, M. A. Sarram, S. Gharaghani, and M. A. Z. Chahooki, “A survey on semi-supervised feature selection methods,” *Pattern Recognition*, vol. 64, no. 4, pp. 141–158, April 2017.

[31] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proc. of the 25th international conference on Machine learning (ICML’2008) (Helsinki)*, July 2008.

[32] A. Demiriz, K. P. Bennett, and M. J. Embrechts, “Semi-supervised clustering using genetic algorithms,” *Artificial neural networks in engineering*, no. 9, pp. 809–814, September 1999.

[33] R. Dara, S. C. Kremer, and D. A. Stacey, “Clustering unlabeled data with soms improves classification of labeled real-world data,” in *Proc. of the 2002 International Joint Conference on Neural Networks (IJCNN’2002) (Hawaii)*, May 2002.

[34] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, July 2006.

[35] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, “Contractive auto-encoders: Explicit invariance during feature extraction,” in *Proc. of the 28th international conference on machine learning (ICML’2011) (Washington)*, June 2011.

[36] N. Lawrence and M. Jordan, “Semi-supervised learning via gaussian processes.” in *Proc. of 18th Conference on Neural Information Processing Systems (NIPS’2004) (Vancouver)*, December 2004.

[37] Y. Grandvalet and Y. Bengio, “Semi-supervised learning by entropy minimization.” in *Proc. of 18th Conference on Neural Information Processing Systems (NIPS’2004) (Vancouver)*, December 2004.

[38] Y.-F. Li and Z.-H. Zhou, “Towards making unlabeled data never hurt,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 1, pp. 175–188, January 2014.

[39] O. Chapelle, V. Sindhwani, and S. S. Keerthi, “Optimization techniques for semi-supervised support vector machines.” *Journal of Machine Learning Research*, vol. 9, no. 2, pp. 203–233, February 2008.

[40] R. Collobert, F. Sinz, J. Weston, L. Bottou, and T. Joachims, “Large scale transductive SVMs.” *Journal of Machine Learning Research*, vol. 7, no. 8, pp. 1687–1712, August 2006.

[41] A. Corduneanu and T. S. Jaakkola, “On information regularization,” in *Proc. of 19th conference on uncertainty in artificial intelligence (UAI’2003) (Acapulco)*, August 2003.

[42] X. Liu, M. Song, D. Tao, Z. Liu, L. Zhang, C. Chen, and J. Bu, “Semi-supervised node splitting for random forest construction,” in *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR’2013) (Portland)*, June 2013.

[43] F. G. Cozman, I. Cohen, M. C. Cirelo *et al.*, “Semi-supervised learning of mixture models.” in *Proc. of the 20th International Conference on Machine Learning (ICML’2003) (Washington)*, August 2003.

[44] D. P. Kingma and M. Welling, “Auto-encoding variational bayes.” in *Proc. of 2nd International Conference on Learning Representations (ICLR’2014) (Banff)*, April 2014.

[45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets.” in *Proc. of 27th Neural Information Processing Systems (NIPS’2014) (Montreal)*, December 2014.

[46] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, January 1977.

[47] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans,” in *Proc. of 30th Conference on Neural Information Processing Systems (NIPS’2016) (Barcelona)*, December 2016.

[48] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. Salakhutdinov, “Good semi-supervised learning that requires a bad gan,” in *Proc. of 31st Conference on Neural Information Processing Systems (NIPS’2017) (Long Beach)*, December 2017.

[49] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, “Semi-supervised learning with deep generative models.” in *Proc. of 27th Neural Information Processing Systems (NIPS’2014) (Montreal)*, December 2014.

[50] M. Belkin, P. Niyogi, and V. Sindhwani, “On manifold regularization,” in *Proc. of International Workshop on Artificial Intelligence and Statistics (IWAIS’2005) (Montreal)*, June 2005.

[51] ——, “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples.” *Journal of Machine Learning Research*, vol. 7, no. 11, pp. 2399–2434, November 2006.

[52] X. Zhu and J. Lafferty, “Harmonic mixtures: Combining mixture models and graph-based methods for inductive and scalable semi-supervised learning.” in *Proc. of 22nd International Conference on Machine learning (ICML’2005) (Bonn)*, August 2005.

[53] V. Sindhwani and D. S. Rosenberg, “An rkhs for multi-view learning and manifold co-regularization.” in *Proc. of the 25th International Conference on Machine Learning (ICML’2008) (Helsinki)*, July 2008.

[54] Z. Qi, Y. Tian, and Y. Shi, “Laplacian twin support vector machine for semi-supervised classification.” *Neural Networks*, vol. 35, no. 8, pp. 46–53, August 2012.

[55] J. Weston, F. Ratle, and R. Collobert, “Deep learning via semi-supervised embedding.” in *Proc. of the 25th international conference on Machine learning (ICML’2008) (Helsinki)*, July 2008.

[56] P. Bachman, O. Alsharif, and D. Precup, “Learning with pseudo-ensembles.” in *Proc. of 27th Neural Information Processing Systems (NIPS’2014) (Montreal)*, December 2014.

[57] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko, “Semi-supervised learning with ladder networks.” in *Proc. of Twenty-ninth Conference on Neural Information Processing Systems (NIPS’2015) (Montreal)*, December 2015.

[58] T. Miyato, S.-I. Maeda, S. Ishii, and M. Koyama, “Virtual adversarial training: A regularization method for supervised and semi-supervised learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1979–1993, July 2018.

[59] S. Melacci and M. Belkin, “Laplacian support vector machines trained in the primal.” *Journal of Machine Learning Research*, vol. 12, no. 3, pp. 1149–1184, March 2011.

[60] B. Zhou, B. Chen, and J. Hu, “Quasi-linear support vector machine for nonlinear classification,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 97, no. 7, pp. 1587–1594, July 2014.

[61] W. Li, B. Zhou, B. Chen, and J. Hu, “A geometry-based two-step method for nonlinear classification using quasi-linear support vector machine,” *IEEJ Trans. on Electrical and Electronic Engineering C*, vol. 12, no. 6, pp. 883–890, November 2017.

[62] W. Li, P. Liang, and J. Hu, “An autoencoder based piecewise linear model for nonlinear classification using quasi-linear support vector machines,” *IEEJ Trans. on Electrical and Electronic Engineering C*, vol. 14, no. 8, pp. 1236–1243, August 2019.

[63] P. Liang, W. Li, and J. Hu, “Oversampling the minority class in a multi-linear feature space for imbalanced data classification,” *IEEJ Trans. on Electrical and Electronic Engineering C*, vol. 13, no. 10, pp. 1236–1243, October 2018.

[64] H. Zhu, Y. Ren, Y. Tian, and J. Hu, “A winner-take-all autoencoder based pieceswise linear model for nonlinear regression with missing data,” *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 16, no. 12, p. (10 pages), December 2021.

[65] B. Zhou, W. Li, and J. Hu, “A coarse-to-fine two-step method for semi-supervised classification using quasi-linear Laplacian SVM,” *IEEJ Transactions on Electrical and Electronic Engineering C*, vol. 14, no. 3, pp. 441–448, March 2019.

[66] L. Sen, Y. Qi, J. Hao, C.-V. J. A, and Z. Yang, “Parasitologist-level classification of apicomplexan parasites and host cell with deep cycle transfer learning (dctl),” *Bioinformatics*, vol. 36, no. 16, pp. 4498–4505, August 2020.

[67] O. Chapelle, S. Bernhard, and A. Zien, *Semi-Supervised Learning*. MIT Press, 2006.

[68] X. Zhu and A. B. Goldberg, “Introduction to semi-supervised learning,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, no. 1, pp. 1–130, January 2009.

[69] A. Blum and S. Chawla, “Learning from labeled and unlabeled data using graph mincuts,” in *Proc. of 18th international conference on machine learning (ICML’2001) (Williamstown)*, June 2001.

[70] X. Zhu, Z. Ghahramani, and J. D. Lafferty, “Semi-supervised learning using gaussian fields and harmonic functions,” in *Proc. of 20th international conference on machine learning (ICML’2003) (Washington)*, August 2003.

[71] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Sch
"olkopf, "Learning with local and global consistency," in *Proc. of 18th annual conference on neural information processing systems (NIPS'2004) (Vancouver)*, December 2004.

[72] S. Basu, A. Banerjee, and R. Mooney, "Semi-supervised clustering by seeding," in *In Proceedings of 19th International Conference on Machine Learning (ICML'2002) (Sydney)*, July 2002.

[73] B. Stephen and V. Lieven, *Convex optimization*. Cambridge university press, 2004.

[74] V. Franc and V. Hlaváč, "An iterative algorithm learning the maximal margin classifier," *Pattern recognition*, vol. 36, no. 9, pp. 1985–1996, September 2003.

[75] P. S. Dhillon, P. P. Talukdar, and K. Crammer, "Inference driven metric learning (idml) for graph construction," University of Pennsylvania, Department of Computer & Information Science, Tech. Rep., 2010.

[76] M. H. Rohban and H. R. Rabiee, "Supervised neighborhood graph construction for semi-supervised classification," *Pattern Recognition*, vol. 45, no. 4, pp. 1363–1372, April 2012.

[77] A. d. A. L. Lilian Berton, "Graph construction based on labeled instances for semi-supervised learning," in *Proc. of 22nd International Conference on Pattern Recognition (ICPR'2014) (Stockholm)*, August 2014.

[78] L. Zhuang, Z. Zhou, S. Gao, J. Yin, Z. Lin, and Y. Ma, "Label information guided graph construction for semi-supervised learning," *IEEE transactions on image processing*, vol. 26, no. 9, pp. 4182–4192, September 2017.

[79] U. von Luxburg, O. Bousquet, and M. Belkin, "On the convergence of spectral clustering on random samples: The normalized case," in *Proc. of International Conference on Computational Learning Theory (COLT'2004) (Banff)*, July 2004.

[80] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[81] L. K. Saul, K. Q. Weinberger, F. Sha, J. Ham, and D. D. Lee, “Spectral methods for dimensionality reduction,” *Semi-supervised learning*, no. 9, pp. 293–308, September 2006.

[82] N. Kamal, M. Andrew, and M. Tom, “Semi-supervised text classification using em,” *Semi-Supervised Learning*, pp. 33–56, September 2006.

[83] T. Joachims, “Transductive inference for text classification using support vector machines,” in *Proc. of the Sixteenth International Conference on Machine Learning (ICML ’1999) (Bled)*, June 1999.

[84] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow, “Realistic evaluation of deep semi-supervised learning algorithms,” in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 3235–3246.

[85] S. Laine and T. Aila, “Temporal ensembling for semi-supervised learning,” in *Proc. of Fifth International Conference on Learning Representations (ICLR’2017) (Toulon)*, April 2017.

[86] Y. Ren, W. Li, and J. Hu, “A semi-supervised classification using gated linear model,” in *Proc. of 2019 IEEE International Joint Conference on Neural Networks (IJCNN’2019) (Budapest)*, jul 2019.

[87] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, may 2015.

[88] A. Makhzani and B. Frey, “k-sparse autoencoders,” in *Proc. of 2014 International Conference on Learning Representations (ICLR’2014) (Banff)*, April 2014.

[89] W. Li, B. Chen, and J. Hu, “A mixture of multiple linear classifiers with sample weight and manifold regularization,” in *Proc. of 2017 IEEE International Joint Conference on Neural Networks (IJCNN’2017) (Anchorage)*, May 2017.

[90] W. Li, B. Zhou, B. Chen, and J. Hu, "A deep neural network based quasi-linear kernel for support vector machines," *IEICE Trans. on Fundamentals of Electronics, communications and Computer Sciences*, vol. E99-A, no. 12, pp. 2558–2565, December 2016.

[91] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" in *Proc. of International Conference on Database Theory (ICDT'1999) (Heidelberg)*, January 1999.

[92] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[93] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[94] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, pp. 55–67, November 2007.

[95] W. H. Organization, "The world health report 2004, changing history," WHO, Tech. Rep., 2004.

[96] J. A. A. Davila and A. H. De Los Rios, "An overview of peripheral blood mononuclear cells as a model for immunological research of toxoplasma gondii and other apicomplexan parasites," *Frontiers in cellular and infection microbiology*, vol. 9, no. 24, pp. 1–10, February 2019.

[97] W. H. Organization, "World malaria report(who)," WHO, Tech. Rep., 2016.

[98] P. J. Krause, "Human babesiosis," *International journal for parasitology*, vol. 49, no. 2, pp. 165–174, February 2019.

[99] A. H. Havelaar, M. D. Kirk, P. R. Torgerson, and et al, "World health organization global estimates and regional comparisons of the burden of foodborne disease in 2010," *PLOS Medicine*, vol. 12, no. 12, December 2015.

[100] L. Wu, L. van den Hoogen, H. Slater, P. Walker, A. Ghani, C. Drakeley, and L. Okell, "Comparison of diagnostics for the detection of asymptomatic plasmodium falciparum infections to inform control and elimination strategies," *Nature*, vol. 528, no. 3, pp. 86–93, December 2015.

[101] M. Mahmud, M. S. Kaiser, A. Hussain, and S. Vassanelli, "Applications of deep learning and reinforcement learning to biological data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2063–2079, June 2018.

[102] H. Lee and Y.-P. P. Chen, "Cell morphology based classification for red cells in blood smear images," *Pattern Recognition Letters*, vol. 49, no. 1, pp. 155–161, November 2014.

[103] K. E. D. Peñas, P. T. Rivera, and P. C. Naval, "Malaria parasite detection and species identification on thin blood smears using a convolutional neural network," in *Proc. of 2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE'2017) (Philadelphia)*, July 2017.

[104] Y. Dong, Z. Jiang, H. Shen, W. D. Pan, L. A. Williams, V. V. Reddy, W. H. Benjamin, and A. W. Bryan, "Evaluations of deep convolutional neural networks for automatic identification of malaria infected cells," in *Proc. of 2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI'2017) (Orland)*, February 2017.

[105] M. D. Kohli, R. M. Summers, and J. R. Geis, "Medical image data and datasets in the era of machine learning-whitepaper from the 2016 c-mimi meeting dataset session," *Journal of digital imaging*, vol. 30, no. 4, pp. 392–399, August 2017.

[106] S. Li, “Microdata,” Mendeley Data, v1, doi: 10.17632/7t3y7j6hh8.1, https://data.mendeley.com/datasets/7t3y7j6hh8/1.

[107] P. H. Le-Khac, G. Healy, and A. F. Smeaton, “Contrastive representation learning: A framework and review,” *IEEE Access*, vol. 8, pp. 193 907–193 934, 2020.

[108] Y. Ren, H. Deng, H. Jiang, H. Zhu, and J. Hu, “A semi-supervised classification method of apicomplexan parasites and host cell using contrastive learning strategy,” in *Proc. of 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC’2021) (online)*, oct 2021.

[109] Y. Ren, W. Li, and J. Hu, “A semi-supervised classifier based on piecewise linear regression model using gated linear network,” *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 15, no. 7, pp. 1048–1056, July 2020.

[110] Y. Ren, “Macroscopic data for apicomplexan parasites,” Mendeley Data, v1, doi: 10.17632/fwfj6zy36f.1, https://data.mendeley.com/datasets/fwfj6zy36f/1.

[111] T. Yao, Y. Pan, C.-W. Ngo, H. Li, and T. Mei, “Semi-supervised domain adaptation with subspace learning for visual recognition,” in *Proc. of 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR’2015) (Boston)*, June 2015.

[112] L. A. Gatys, A. S. Ecker, and M. Bethge, “Texture synthesis using convolutional neural networks,” in *Proc. of Twenty-ninth Conference on Neural Information Processing Systems (NIPS’2015) (Montreal)*, December 2015.

[113] L. a. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR’2016) (Las Vegas)*, June 2016.

[114] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *Proceedings of 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR’2020) (online)*, June 2020.

[115] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv preprint arXiv:1807.03748*, 2018.

[116] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” in *Proc. of 2015 annual conference on neural information processing systems (NIPS’2015) (Montreal)*, December 2015.

[117] Y. Ren, H. Zhu, Y. Tian, and J. Hu, “A Laplacian SVM based semi-supervised classification using multi-local linear model,” *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 16, no. 3, pp. 455–463, March 2021.

[118] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of 2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR’2017) (Hawaii)*, July 2017.

Publication List

Journal Papers

[J1] Y. Ren, H. Jiang, H. Zhu, Y. Tian and J. Hu, “A Semi-Supervised Classification Method of Parasites Using Contrastive Learning”, *IEEJ Trans. on Electrical and Electronic Engineering*, Vol.17, No.3, March, 2022. (9 pages)

[J2] H. Zhu, Y. Ren, Y. Tian and J. Hu, “A Winner-Take-All Autoencoder Based Pieceswise Linear Model for Nonlinear Regression with Missing Data”, *IEEJ Trans. on Electrical and Electronic Engineering*, Vol.16, No.12, December, 2021. (10 pages)

[J3] Y. Ren, H. Zhu, Y. Tian and J. Hu, “A Laplacian SVM Based Semi-Supervised Classification Using Multi-Local Linear Model”, *IEEJ Transactions on Electrical and Electronic Engineering*, Vol.16, No.3, pp.455-463, March, 2021.

[J4] H. Zhu, Y. Tian, Y. Ren and J. Hu, “A Hybrid Model for Nonlinear Regression with Missing Data Using Quasi-Linear Kernel”, *IEEJ Trans. on Electrical and Electronic Engineering*, Vol.15, No.12, pp.1791-1800, December, 2020.

[J5] Y. Ren, W. Li and J. Hu, “A Semi-Supervised Classifier Based on Piecewise Linear Regression Model Using Gated Linear Network”, *IEEJ Transactions on Electrical and Electronic Engineering*, Vol.15, No.7, pp.1048-1056, July, 2020.

Conference Papers with Review

[P1] Y. Ren, H. Deng, H. Jiang, H. Zhu and J. Hu, “A Semi-Supervised Classification Method of Apicomplexan Parasites and Host Cell Using Contrastive Learning Strategy”, in *Proceedings of 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC’2021) (online)*, October, 2021. (6 pages)

[P2] H. Zhu, Y. Ren and J. Hu, “ Establishing a Hybrid Pieceswise Linear Model for Air Quality Prediction Based Missingness Challenges”, in *Proceedings of 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC’2021) (online)*, October, 2021. (6 pages)

[P3] Y. Ren, W. Li and J. Hu, “A Semi-Supervised Classification Using Gated Linear Model”, in *Proceedings of 2019 IEEE International Joint Conference on Neural Networks (IJCNN’2019) (Budapest)*, July, 2019. (7 pages)