NAZARBAYEV
UNIVERSITY

Towards effective usage of unlabeled data in small labeled sample classification

by

Azamat Mukhamediya

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering

Date of Completion
July, 2025

Towards effective usage of unlabeled data in small labeled sample classification

by
Azamat Mukhamediya

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering

Electrical and Computer Engineering
School of Engineering and Digital Sciences
Nazarbayev University

July, 2025

Supervised by
Prof. Amin Zollanvari
Prof. Siamac Fazli
Prof. Reza Sameni

Declaration

I, Azamat Mukhamediya, declare that the research contained in this thesis, unless otherwise formally indicated within the text, is the author's original work. The thesis has not been previously submitted to this or any other university for a degree and does not incorporate any material already submitted for a degree.

Signature:

Date:

BLANK

Abstract

In numerous real-world applications, the scarcity of high-quality labeled data constitutes a significant impediment to the development of supervised machine learning models. This challenge primarily arises from the fact that manual annotation processes are often resource-intensive, requiring considerable time, expert knowledge, specialized equipment, or elaborate experimental procedures. Consequently, collecting a sufficiently large labeled dataset is frequently impractical. Conversely, unlabeled data are typically abundant and readily accessible.

This thesis addresses the problem of small labeled sample classification, wherein only a limited annotation budget is available. It demonstrates that deliberate exploitation of unlabeled data can: (i) substantially enhance the predictive performance of classifiers trained on small labeled datasets, and (ii) reduce the cost associated with data labeling. To this end, the research investigates two principal directions based on the accessibility of a labeling expert: semi-supervised learning (without direct expert access) and active learning (with expert access).

This raises two research questions: (i) how can we use unlabeled data to improve the performance of a classifier trained on small labeled data? and (ii) how can we leverage unlabeled data to effectively identify a set of labeled samples that are most informative in terms of predictive performance?

Within the scope of semi-supervised learning, the thesis introduces an enhanced self-training algorithm that mitigates the prevalent issue of noise accumulation—wherein incorrect pseudo-labels reinforce suboptimal model predictions—by randomly partitioning unlabeled data into mini-batches during self-training. The experimental results show that enhanced self-training outperforms standard self-training in 85% of cases considered in this work. Furthermore, to more systematically address noise accumulation, the research proposes a novel semi-supervised boosting algorithm. This algorithm is carefully designed to leverage three core assumptions of semi-supervised learning: (i) the smoothness assumption, which posits that data points situated closely within high-density regions should be assigned the same label; (ii) the cluster assumption, which suggests that data points belonging to the same cluster are likely to share the same class; and (iii) the manifold assumption, which holds that high-dimensional data often lie on an underlying lower-dimensional manifold that captures

its intrinsic geometric structure. By incorporating these assumptions, the proposed algorithm improves pseudo-label quality and reduces the risk of error reinforcement, thereby enhancing overall classifier performance. In particular, proposed algorithm outperforms other methods in 91% comparisons investigated in this work.

In the domain of active learning, the thesis investigates the estimated error reduction (EER) approach, which prioritizes the selection of unlabeled data points based on their potential to reduce classifier error. In comparison with traditional methods that primarily focus on data diversity, the EER approach directly considers the effect of labeling specific data points on predictive performance. Despite its theoretical promise, the original EER method suffers from high computational demands due to the necessity of retraining the classifier for every candidate data point and each possible label. To address this limitation, the research introduces a novel, computationally efficient active learning algorithm. By formulating an innovative objective function, the method enables recursive, closed-form updates that avoid the need for repeated retraining, thereby significantly reducing computational cost. As a result, proposed active learning method shows better results than other methods in 81% cases considered in this work.

Through extensive empirical evaluations, the thesis demonstrates that effective utilization of unlabeled data can meaningfully improve classification performance and decrease the reliance on expensive labeled data. The contributions presented herein advance the understanding and practical implementation of semi-supervised and active learning, particularly under constraints of limited labeled data.

Contents

List of Tables

List of Figures

List of Abbreviations

ML	Machine Learning
SSL	Semi-Supervised Learning
ST	Self-Training
F-ST	Full-batch ST
SRPM-ST	Sequential Retraining and Pseudo-labeling in Mini-batches for ST
BSSL++	Boosting algorithm for SSL inspired by Learn++
AdaBoost	Adaptive Boosting
ASSEMBLE	Adaptive Semi-Supervised ensEMBLE
RegBoost	Regularized Semi-Supervised Boosting
SemiBoost	Semi-Supervised Boosting
AdaBoost.SDM	AdaBoost with Similarity and Dissimilarity-based Manifold regularization
BSSL++.SDM	BSSL++ with Similarity and Dissimilarity-based Manifold regularization
AL	Active Learning
EER	Estimated Error Reduction
EER-P	Estimated Error Reduction with Prior probabilities
RLDA	Regularized Discriminant Analysis
AL-RLDA	AL method that minimizes an accurate closed-form error estimation of RLDA
KNN	k-Nearest Neighbors
LDA	Linear Discriminant Analysis
SVM	Support Vector Machine
S3VM	Semi-Supervised Support Vector Machine
LSVM	Linear Support Vector Machine
LapSVM	Laplacian Support Vector Machine
LapRLS	Laplacian Regularized Least Squares
LRR	Logistic Regression
RF	Random Forest
MLP	Multi-layer Perceptron
NB	Naive Bayes

DNN	Deep Neural Network
TabNet	Deep Neural Network for Tabular data
ProbCover	Probability Coverage
BALD	Bayesian Active Learning by Disagreement
SCARF	Self-Supervised Contrastive Learning using Random Feature Corruption
CV	Cross-validation

Chapter 1

Introduction

1.1 Small Labeled Sample Classification

The data is a crucial part of any machine learning (ML) pipeline. In supervised learning, the raw data is usually required to be analyzed and assigned with certain labels or numerical values according to some predefined criterion. These labeled data can be used to train a classifier to make predictions on future, unseen data. In general, the performance of the predictive model depends on the number of available labeled data points. This well-known traditional wisdom of "the more data, the better" has been justified for many statistically consistent learning rules (Braga-Neto, 2020). That being said, the dependence of the model on the large volumes of high-quality labeled datasets can be a major problem. That is to say, in many real-world applications, obtaining such datasets is often prohibitively expensive or practically infeasible, as it may require domain experts or additional empirical experiments. For example,

- in speech recognition (Yu et al., 2010; Drugman et al., 2016), obtaining accurate transcription of speech utterances is extremely time-consuming;
- in natural language processing (Zhou et al., 2013; Tomanek and Hahn, 2009), annotating sentences and categorizing texts are laborious and require linguists expert knowledge; and
- in medical and biological data processing (Zhang et al., 2022; Camargo et al., 2020; Su et al., 2015), labeling the data can require expensive laboratory experiments by experts.

At the same time, a large number of unlabeled data points can be easily collected from a wide range of sources, including podcasts, web pages, and medical databases. Therefore, there is a growing need for algorithms that can effectively use unlabeled data in small labeled sample classification problem, which appears when the annotation budget allows labeling only a limited amount of data.

Since conventional supervised learning algorithms are not capable of utilizing unlabeled data in the training process, multiple approaches have been proposed to effectively leverage unlabeled data to reduce labeling costs. In principle, we can highlight two main directions for using unlabeled data based on access to the oracle (i.e., data labeling expert): (i) semi-supervised learning, in which we do not have access to the oracle during model training; and (ii) active learning, where it is assumed that we have access to the oracle. Both approaches lead to a reduction in human labeling efforts, especially, in cases where the amount of available annotation budget or time is very limited.

1.2 Semi-Supervised Learning

Semi-supervised learning (SSL) is one of the major ML paradigms that focuses on constructing a model that can encode both labeled and unlabeled data into the training process. In general, most of the SSL methods either modify or extend a supervised learning algorithm to leverage additional information from unlabeled data (Zhu, 2005; Van Engelen and Hoos, 2020; Yang et al., 2022). In the classification task, the aim of SSL is to use both labeled and unlabeled data to train a classifier, such that the obtained classifier is better than a supervised classifier trained on merely labeled data. The working principle of most SSL methods is generally based on assumptions that link the distribution of unlabeled data and the target label. Traditionally, there are three main SSL assumptions: semi-supervised smoothness, cluster, and manifold assumptions (Chapelle et al., 2006; Van Engelen and Hoos, 2020).

- Semi-supervised smoothness assumption states that the output of data points in a high-density region should vary smoothly with the distance. That is to say, two data points that are close to each other in a high-density region should share similar output labels.

- Cluster assumption is based on the observation that if data points of each class tend to form a cluster, then unlabeled data could help to identify the boundary of each cluster. In this regard, data points in the same clusters are likely to be from the same class.

- Manifold assumption is described as the data in high-dimensional space may lie on a low-dimensional manifold, and their outputs vary smoothly along the manifold.

These assumptions have been used as a foundation of most SSL methods (Zharmagambetov and Carreira-Perpiñán, 2022; Chen and Wang, 2011; Zhang et al., 2021), in which one or more are usually satisfied either explicitly or implicitly.

In a broader context, SSL can be used in two settings: inductive and transductive. This contrasts with supervised learning, where the only goal is to use a trained model on future test data, which is an inductive setting. In SSL, an inductive setting employs unlabeled data in addition to labeled data to train a model. On the other hand, a transductive setting uses labeled and unlabeled to assign labels for the unlabeled data used in training. Hence, in this setting, the prediction on the test data is not required as it is only concerned with predicting labels for used unlabeled data. In this thesis, we focus on an inductive setting.

Inductive SSL methods can be differentiated based on how unlabeled data is treated in the training process: in the preprocessing step, in the objective function, and using pseudo-labeling. The first approach corresponds to unsupervised preprocessing methods, which use labeled and unlabeled data in two steps (Van Engelen and Hoos, 2020). One method in this approach is cluster-then-label (Demiriz et al., 1999; Goldberg et al., 2009), in which: (i) labeled and unlabeled data are clustered using an unsupervised clustering algorithm; (ii) for each cluster, a supervised classifier is trained on a labeled data that is allocated to that cluster; and (iii) unlabeled data is assigned with labels using the classifiers from their respective clusters.

The second approach corresponds to intrinsically semi-supervised methods (Van Engelen and Hoos, 2020), which directly optimizes the objective function with encoded components for labeled and unlabeled data. Unlike other approaches, these methods do not rely on any supervised or unsupervised algorithms. One of the natural extensions of objective functions for the semi-supervised setting is semi-supervised Support Vector Machines (S3VM) (Bennett and Demiriz, 1998; Vapnik, 1995). In supervised SVM, the decision boundary is found by maximizing the margin using labeled data, where the margin is defined as the distance between the decision boundary and the nearest data points to it. At the same time, given the unlabeled data, S3VM aims to ensure that the decision boundary would cut through the low-density region of unlabeled data in addition to margin maximization.

The third and the simplest approach to extending supervised learning algorithms to the semi-supervised setting corresponds to the pseudo-labeling or wrapper methods (Van Engelen and Hoos, 2020; Zhu, 2005). In particular, these methods use a supervised classifier trained on labeled data to obtain predictions for unlabeled data, then retrains the classifier using both *pseudo-labeled* data and existing labeled data. Since the pseudo-labels are obtained using a wrapper procedure, a classifier is unaware of the difference between pseudo-labeled and labeled data. In this thesis, we investigate the wrapper SSL methods.

One of the earliest and easy-to-apply wrapper SSL methods is self-training (ST) (Scudder, 1965; Xie et al., 2020). In principle, ST is an iterative learning procedure that employs its own predictions to teach itself using one supervised classifier. The extension

of ST for multiple classifiers corresponds to the co-training method (Blum and Mitchell, 1998). This method trains two (or more) classifiers on different feature sets of labeled data. Then, each classifier is used to obtain pseudo-labels for unlabeled data. The most confident pseudo-labels generated by one classifier are added to the labeled set of another classifier and vice versa. Ultimately, the classifiers are iteratively retrained on combined sets. Another approach that can be considered as a wrapper method corresponds to semi-supervised boosting ensemble learning (Bennett et al., 2002). In semi-supervised boosting, an ensemble classifier is constructed by sequentially learning base classifiers, where at each iteration, the base classifier is trained on both labeled and pseudo-labeled unlabeled data generated using the ensemble classifier constructed so far. In principle, ST and usually most wrapper methods are based on the assumption that their own predictions, or at least high-confidence ones, tend to be correct. This assumption is more likely to hold when the data can be separated into distinct class clusters. That is to say, the underlying assumption coincides with the cluster assumption (Goldberg and Zhu, 2010; Amini et al., 2025).

The obvious drawback of wrapper methods is that they depend on initial pseudo-labeled data that may include incorrectly produced pseudo-labels. These incorrect pseudo-labels are then added to labeled data, which may be magnified in successive iterations. This leads to the generation of more and more incorrect pseudo-labels, resulting in performance degradation. This phenomenon is known as the noise accumulation phenomenon. (Arazo et al., 2020; Lee et al., 2023; Zhang et al., 2016; Wang et al., 2023b).

In this regard, this thesis investigates wrapper semi-supervised methods with a focus on mitigating noise accumulation phenomenon.

1.3 Active Learning

Another direction that aims to use efficiently unlabeled data to reduce the labeling cost corresponds to *active labeling* or *active learning* (AL). In particular, AL constructs a labeled set by selecting the most beneficial unlabeled data for labeling in terms of enhancing predictive performance. This is in contrast to classical passive learning, where a labeled set is constructed by randomly selecting a set of unlabeled data points for obtaining labels. That is to say, AL facilities prioritization of most informative unlabeled data points for labeling. The primary motivation of AL is based on the observation that in some applications cost of labeling the data points is substantially high. That is the reason why, for a given classification task, it is reasonable to label the most beneficial data points that can help to better distinguish class boundaries. Hence, AL relies on the assumption that not all data points are equally important from a labeling perspective (Aggarwal et al., 2014). For example, some data points may contain noises

that cannot improve the generalization performance of the classifier. In other words, some data points are less useful compared to ones that can enhance the classifier. In this regard, given an annotation budget, AL methods can identify the most informative data points from unlabeled data. An important note that distinguishes active learning from passive learning is that the oracle is available for interactively labeling selected data points. That is to say, AL comprises two main components: oracle and unlabeled data point selection strategy.

There are a few possible scenarios that can be posed to AL based on how unlabeled data are collected (Aggarwal et al., 2014; Settles, 2009). One of the earliest scenarios is membership query synthesis (Schumann and Rehbein, 2019), in which a synthetic unlabeled data point is generated and requested for labeling. The rationale behind this strategy is to generate a data point based on areas where it is assumed will bring additional helpful information. Another scenario is stream-based selective sampling (Cacciarelli et al., 2022), in which unlabeled data points become available one by one, and the decision to label or discard the arrived data point needs to be made in real-time. The key assumption in this scenario is that obtaining unlabeled data points is relatively inexpensive. Another and perhaps most common scenario is pool-based active learning (Zhan et al., 2021). Since, for many real-world applications, a large amount of unlabeled data can be collected at once, in this strategy, it is needed to identify the most important data points from a pool of unlabeled data points.

In classification, AL constructs a labeled set to achieve either a specified classification accuracy level or a predefined stopping criterion using as few labeled data points as possible. As a result, the essential part of AL is the query strategy determining which data points to select for labeling. There are various query strategies based on the approach of defining the informativeness of unlabeled data points. Perhaps the simplest strategy is to randomly select unlabeled data points, which is commonly known as random sampling. However, more widely used strategies focus on selecting heterogeneous data points. For example, uncertainty sampling (Kirsch et al., 2019; Houlsby et al., 2011) queries unlabeled data points that the classifier trained on given labeled data is less certain about. Diversity sampling (Gao et al., 2020; Shui et al., 2020) selects data points from diverse regions of the data distribution. Query-by-committee sampling (Zhao et al., 2006; Krawczyk and Woźniak, 2017) trains a committee of different classifiers using labeled data and selects data points where classifiers disagree the most. These strategies attempt to query data points that are dissimilar to data points that labeled data already contains. However, they do not take into account the direct effect of including an unlabeled data point with its potential class label to the labeled set on the performance of the classifier (Aggarwal et al., 2014). On the other hand, the estimated error reduction-based (EER) strategy focuses on reducing the generalization error (Roy and McCallum, 2001; Moskovitch et al., 2007). In particular, this strategy

aims to select data points, that when used with labeled data for training, reduce the estimate of the error rate the most. However, in its original form, the estimated error reduction-based strategy is computationally expensive. Since, at its core, it requires evaluating and retraining the estimator using each candidate unlabeled data point for all possible class labels (Aggarwal et al., 2014; Tharwat and Schenck, 2023).

In this regard, this thesis focuses on a pool-based active learning scenario and investigates EER-based AL algorithms with an attention on efficiency and effectiveness.

Table 1.1: Taxonomy-based comparison of SSL and AL.

	Semi-Supervised Learning	Active Learning
Data setup	Labeled and large unlabeled pool	Labeled and access to labeling oracle
Data flow	Static (all data known in advance)	Interactive (query-based)
Labeling	Pseudo-labels from model	True labels via queries
Data selection	Implicit (uses all unlabeled data)	Explicit (selects informative samples)
Objective	Improve generalization	Maximize label efficiency

Table 1.1 presents a taxonomy-based comparison of SSL and AL, which highlights their methodological distinctions and complementary nature. Particularly, SSL works in a passive setting by leveraging the structure of unlabeled data through pseudo-labeling. On the other hand, AL actively interacts with a labeling oracle to acquire the most informative samples. The table outlines differences across key dimensions such as data flow, data setup, label acquisition strategy, data selection, and objectives.

1.4 Research Questions

The thesis explores two main directions to effectively leveraging unlabeled data for small labeled sample classification tasks and aims to study SSL and AL for reducing labeling costs and improving predictive performance.

The first direction is to use available large number of unlabeled data points to building a better classification model compared to one that is trained on labeled data only. In this regard, three main SSL assumptions can possibly be employed to achieve this goal. That being said, in the absence of effective mechanism, SSL methods are prone to noise accumulation. Particularly, incorrectly used unlabeled data can harm the performance. Hence, we also take into account the effect of noise accumulation.

The second direction is to carefully construct a labeled dataset by querying unlabeled data points for labeling that are most beneficial and informative in terms of classification performance. In this regard, the EER-based approach is readily applicable to consider the reduction of generalization error as the main objective for the selection of unlabeled data points. However, this approach generally depends on the repetitive retraining and

evaluation procedure that demands highly computational resources. As a result, we aim to avoid this repetitive procedure to efficiently and accurately select data points.

The research questions of this thesis can be formulated as follows.

Q1. Is it possible to use unlabeled data to improve the performance of the classifier trained on small labeled data and, at the same time, safeguard against noise accumulation by employing SSL assumptions?

Q2. Is it possible to use unlabeled data to identify a set of labeled samples that are most informative in terms of predictive performance by employing the AL algorithm with guaranteed efficiency and effectiveness?

1.5 Contributions

The contributions of this thesis are as follows.

- We propose a sequential semi-supervised method for ST. The method sequentially retrains a classifier and pseudo-labels unlabeled data in the form of mini-batches—we refer to this method as SRPM. We show that SRPM could improve the performance of the classifier with respect to the pervasive form of ST. Moreover, we show that there is a data-dependent mini-batch size that leads to the least error rate.

- Inspired by an incremental supervised boosting technique, namely, Learn++, we introduce a semi-supervised boosting algorithm, named BSSL++. In particular, by taking into account the semi-supervised smoothness and cluster assumptions, BSSL+ is designed to mitigate noise accumulation.

- In order to extend BSSL++ to encode the manifold assumption, we first introduce a novel supervised boosting algorithm with manifold regularization. In particular, based on manifold assumption, we define a convex objective function that promotes a smooth change of predictions over data manifold using the mixed-graph Laplacian. Then, having defined the objective function, we extend a supervised boosting algorithm to encode manifold regularization for the case of binary classification. With the developed boosting algorithm, we extend BSSL++ to include manifold regularization.

- We introduce a novel efficient AL algorithm that uses an accurate closed-form error estimator of regularized linear discriminant analysis (AL-RLDA) for the case of binary classification. At its core, AL-RLDA employs a new EER objective function that uses class prior probabilities.

These novel methods contribute both theoretically and practically to the fields of SSL and AL. Particularly, they enhance model performance and significantly reduce labeling costs by improving pseudo-labeling reliability and enabling efficient sample selection. The proposed frameworks are scalable and practically applicable to real-world domains such as healthcare, genomics, and affective computing, where labeled data is often scarce and expensive. Furthermore, the theoretical advancements provide a strong foundation for future extensions and adaptations of SSL and AL methods to more complex and structure-aware scenarios.

1.6 Datasets

Throughout the thesis, all numerical experiments were conducted using datasets collected from OpenML and UCI publicly available dataset repositories, with the corresponding links provided in Appendix A.1.

1.7 Source code

The link of source code for reproducing the experimental results presented in this thesis is presented in Appendix A.2.

1.8 Outline

The rest of the thesis is organized as follows.

Chapter 2 presents a novel ST algorithm that is based on sequential retraining and pseudo-labeling in mini-batches (SRPM-ST).

Chapter 3 introduces a novel boosting algorithm for semi-supervised learning based on semi-supervised smoothness and cluster assumptions inspired by the incremental learning algorithm Learn++ (BSSL++).

Chapter 4 features the extending BSSL++ to take into account the manifold assumption. As a result, it also describes a novel supervised boosting algorithm with manifold regularization.

Chapter 5 shifts the focus to introducing a novel efficient AL algorithm based on the EER approach that uses a closed-form error estimator of RLDA (AL-RLDA).

Chapter 6 concludes the thesis and describes future directions.

1.9 Related Publications

1. A. Mukhamediya and A. Zollanvari, "SRPM-ST: Sequential retraining and pseudo-labeling in mini-batches for self-training," *Neurocomputing*, vol. 605, p. 128343, 2024.

2. A. Mukhamediya, R. Sameni, and A. Zollanvari, "BSSL++: A Boosting Algorithm for Semi-Supervised Learning Inspired by Learn++," *Under review*.

3. A. Mukhamediya and A. Zollanvari, "AdaBoost.SDM: Similarity and Dissimilarity-based Manifold Regularized Adaptive Boosting Algorithm," *Pattern Recognition Letters*, vol. 196, p. 66, 2025.

4. A. Mukhamediya, R. Sameni, and A. Zollanvari, "Efficient Active Learning using Recursive Estimation of Error Reduction," *Neurocomputing*, vol. 638, p. 130131, 2025.

The first research article provides foundations and experimental results for Chapter 2. The second (submitted) and third (published) articles are presented in Chapters 3 and 4, respectively, whereas the fourth research article provides results that are described in Chapter 5.

During my PhD, I published a research article that investigates the effect of tuning log-Mel spectrogram parameters in speech emotion recognition applications. This work served as an ideological foundation for exploring the usage of unlabeled data for small labeled sample classification problems. Speech emotion recognition is one of the applications where scarcity of labeled data, as well as difficulty in collecting annotations, is an issue.

1. A. Mukhamediya, S. Fazli, and A. Zollanvari, "On the Effect of Log-Mel Spectrogram Parameter Tuning for Deep Learning-Based Speech Emotion Recognition," *IEEE Access*, vol. 11, pp. 61950–61957, 2023.

As this work does not contribute to a small labeled sample classification problem, this paper is not presented within this thesis.

BLANK

Chapter 2

SRPM-ST: Sequential Retraining and Pseudo-labeling in Mini-batches for Self-Training

In this chapter, we investigate the first direction of this thesis research, in which we aim to effectively use unlabeled data in the self-training (ST) framework.

2.1 Introduction

ST is one of the well-known forms of semi-supervised learning (SSL) (Scudder, 1965; Xie et al., 2020). From an algorithmic perspective ST proceeds as follows: (i) a supervised learning algorithm is called to train a classifier using an initial small-sized labeled dataset; (ii) the classifier is used to obtain predictions on entire unlabeled data, that is, to produce *pseudo-labels* for unlabeled data; (iii) either entire set or a subset of pseudo-labeled unlabeled data (e.g., using some confidence score) is merged with labeled data to form an updated labeled set; (iv) the updated set is used to retrain the classifier; and (v) steps (ii)-(iv) are repeated until some stopping criteria are met. ST has been effectively applied across various applications, such as image classification (Xie et al., 2020), semantic segmentation (Du et al., 2022), and text classification (Mukherjee and Awadallah, 2020).

There are different strategies that have been proposed to select either a subset or an entire set of pseudo-labeled data in step (ii) (Xie et al., 2020; Sohn et al., 2020; Zhang et al., 2021; Wang et al., 2023a). For example, Xie *et al.* (Xie et al., 2020) proposed a form of ST using the teacher-student framework. In particular, the teacher model is trained using labeled data and is used to obtain pseudo-labels for entire unlabeled data. A student model is trained on combined labeled and pseudo-labeled data. This process is repeated a few times by exchanging the roles of student and teacher models to generate updated pseudo-labels and to train a new student.

Sohn *et al.* (Sohn et al., 2020) proposed the FixMatch algorithm that aims to match predictions of the model on "weakly" and "strongly" augmented unlabeled images. In particular, they generated pseudo-labels for "weakly" augmented images and selected those with high confidence given a fixed (hard) threshold value. And trained the model to predict the pseudo-label given the same "strongly" augmented image. A similar strategy was employed by Zhang *et al.* (Zhang et al., 2021) where they selected pseudo-labels using a soft thresholding strategy instead of relying on hard threshold values. On the other hand, Wang *et al.* (Wang et al., 2023a) proposed to filter out pseudo-labels with high confidence scores for the domain adaptation framework in order to prevent bias in favor of the source domain.

Nevertheless, regardless of the subset selection technique, the pervasive form of ST implicitly assumes that during the ST procedure, the classifier is retrained only after pseudo-labels for entire unlabeled data are obtained. This form of ST will henceforth be referred to as full-batch ST (F-ST).

In this chapter, we focus on the connection between steps (ii)-(iv), and therefore we make the following assumptions.

Assumption 2.1.1. *All pseudo-labeled data is used in step (iii).*

Assumption 2.1.2. *No repetition is involved in step (v).*

In particular, we investigate the effect of unlabeled set size used in steps (ii)-(iv) on the performance of the classifier obtained at the end of the ST procedure. As a result, motivated by the online learning setting, where data commonly become available in batches of unlabeled or labeled data points (He et al., 2020; Hoi et al., 2021; Gu et al., 2013), we introduce sequentially retraining and pseudo-labeling mini-batches for ST (SRPM-ST).

We show that SRPM-ST can, on average, lead to better classification performance compared to F-ST. Notably, the performance of the classifier is improved, although SRPM retrains the classifiers in an implicit order, while mini-batches of unlabeled data are used in an *arbitrary* order. That being said, in practice, the size of the mini-batches is data-dependent. Hence, it can be considered as a hyperparameter to estimate, for instance, applying cross-validation. Nevertheless, the precision of such estimation is not guaranteed *a priori* due to the large variance of cross-validation estimation, especially in small labeled sample classification (Braga-Neto, 2020). Therefore, we also investigate the capability of cross-validation in estimating the mini-batch size hyperparameter.

2.2 Motivation

Consider a binary classification task in a univariate setting, where the data are generated from a mixture of two Gaussian distributions, denoted P_X. Particularly, a sample

point $X = x$ is generated from either population $\pi^+ \sim N(\mu^+, \sigma^2)$ or population $\pi^- \sim N(\mu^-, \sigma^2)$ with equal probability of 0.5. Hence, the optimal Bayes classifier ψ assigns class "+" to x if the discriminant function $f(x) > 0$, where (Duda et al., 2001)

$$f(x) = \Big(x - \frac{\mu^+ + \mu^-}{2}\Big)(\mu^+ - \mu^-); \tag{2.1}$$

otherwise, x is classified to class "−". Without loss of generality, we assume $\mu^+ > \mu^-$. Therefore, $\psi(x)$ is equivalent to

$$\psi(x) = \begin{cases} +, & \text{if } x > \theta, \\ -, & \text{otherwise,} \end{cases} \tag{2.2}$$

where $\theta = \frac{\mu^+ + \mu^-}{2}$.

The following assumptions are underlying for ST:

Assumption 2.2.1. *There is an initial classifier $\psi_0(x)$ that was trained on a small number of labeled data points.*

Assumption 2.2.2. *There is a large-size unlabeled dataset $\mathcal{U} = \{x_i\}_{i=1}^u$ of size u that can be utilized to update $\psi_0(x)$.*

For simplicity in the discussion, we split $\mathcal{U}$ into two disjoint mini-batch sets $\mathcal{U}_1 = \{x_i\}_{i=1}^m$ and $\mathcal{U}_2 = \{x_i\}_{i=m+1}^{2m}$ of size $m = u/2$. We note that in practice, we consider the size of the mini-batches or equivalently the number of mini-batches as a hyperparameter. Given two mini-batches of unlabeled data, SRPM-ST produces pseudo-labels for $\mathcal{U}_1$ and uses them to retrain $\psi_0(x)$, which will result in classifier $\psi_1(x)$. Then, $\psi_1(x)$ is used to produce pseudo-labels for $\mathcal{U}_2$, and is updated by the new pseudo-labels. In principle, both SRPM-ST and F-ST generates the same pseudo-labels for $\mathcal{U}_1$; hence, in what follows the primary focus in centered around the quality of pseudo-labels assigned for $\mathcal{U}_2$.

In SRPM-ST, generation of pseudo-labels for $\mathcal{U}_1$ yields m_1^+ and m_1^- observations ($m_1^+ + m_1^- = m$) pseudo-labeled as "+" and "−", denoted as $\{x_{i,1}^+\}$ and $\{x_{i,1}^-\}$, respectively. Let p_0^+ and p_0^- denote the probability of correctly classifying a sample point that belongs to class "+" and class "−" using $\psi_0(x)$, respectively. Based on Assumption 2.2.1, we neglect the effect of having a small amount of labeled data compared to a large amount of unlabeled data in training $\psi_1(x)$. In other words, we assume $\psi_1(x)$ is only trained using $\{x_{i,1}^+\}$ and $\{x_{i,1}^-\}$. Supposing that $\psi_1(x)$ is a plug-in classifier in the same family of classifiers described by (2.2), then the estimator of θ solely defines $\psi_1(x)$. Hence, a natural estimator of θ, denoted $\hat{\theta}_1^{\text{SRPM}}$, given by

$$\hat{\theta}_1^{\text{SRPM}} = \frac{1}{2}\Big(\frac{\sum_{i=1}^{m_1^+} x_{i,1}^+}{m_1^+} + \frac{\sum_{i=1}^{m_1^-} x_{i,1}^-}{m_1^-}\Big). \tag{2.3}$$

At this point, to estimate the optimal θ, we leverage the results of (Yang and Xu, 2020). In this regard, we use our ability in estimating the optimal θ as a proxy for the

quality of pseudo-labels utilized in the estimation. Particularly, we have (Theorem 1 in (Yang and Xu, 2020))

$$
\begin{aligned}
&P\left(\left|\hat{\theta}_1^{\text{SRPM}} - \frac{\mu^+ + \mu^-}{2} - \Delta_0 \frac{\mu^+ - \mu^-}{2}\right| \geq \epsilon\right) \leq \\
&\leq 2e^{-\frac{2\epsilon^2}{9\sigma^2}\frac{m_1^+ m_1^-}{m_1^+ + m_1^-}} + 2e^{-\frac{8m_1^+\epsilon^2}{9(\mu^+ - \mu^-)^2}} + 2e^{-\frac{8m_1^-\epsilon^2}{9(\mu^+ - \mu^-)^2}},
\end{aligned}
\tag{2.4}
$$

for any $\epsilon > 0$, where $\Delta_0 \triangleq p_0^+ - p_0^-$.

In principle, m_1^+ comprises two sets of samples. One is m_1^{++}, set of samples that were generated from population π^+ and were pseudo-labeled as class "+", and another one is m_1^{+-} set of samples that were generated from population π^- but have been pseudo-labeled as class "+". We refer to the former and latter sets as $\{x_{i,1}^{++}\}$ and $\{x_{i,1}^{+-}\}$, respectively. Since the samples from both classes are generated with equal probability and in the light of Assumption 2.2.2, each set of mini-batches includes approximately $m/2$ observations from each of the positive and negative classes. At the same time, we have $m_1^+ \approx \frac{m}{2}\Big(p_0^+ + (1 - p_0^-)\Big)$. Moreover, we can write

$$
\frac{\sum_{i=1}^{m_1^+} x_{i,1}^+}{m_1^+} = \frac{\sum x_{i,1}^{++}}{m_1^+} + \frac{\sum x_{i,1}^{+-}}{m_1^+}. \tag{2.5}
$$

Based on Assumption 2.2.2 and the law of large numbers, we have

$$
\sum x_{i,1}^{++} \approx \frac{m p_0^+}{2}\mu^+, \tag{2.6}
$$

$$
\sum x_{i,1}^{+-} \approx \frac{m(1 - p_0^-)}{2}\mu^-. \tag{2.7}
$$

Replacing (2.6), (2.7), and m_1^+ in (2.5), we have

$$
\frac{\sum_{i=1}^{m_1^+} x_{i,1}^+}{m_1^+} \approx \alpha_0^+ \mu^+ + (1 - \alpha_0^+)\mu^-, \tag{2.8}
$$

where $\alpha_0^+ = p_0^+ / \Big(p_0^+ + (1 - p_0^-)\Big)$. In similar manner, we can demonstrate that

$$
\frac{\sum_{i=1}^{m_1^-} x_{i,1}^-}{m_1^-} \approx (1 - \alpha_0^-)\mu^+ + \alpha_0^- \mu^-, \tag{2.9}
$$

where $\alpha_0^- = p_0^- / \Big(p_0^- + (1 - p_0^+)\Big)$. Replacing (2.8) and (2.9) in (2.3) yields

$$
\hat{\theta}_1^{\text{SRPM}} \approx \frac{1}{2}(1 - \alpha_0^- + \alpha_0^+)\mu^+ + \frac{1}{2}(1 - \alpha_0^+ + \alpha_0^-)\mu^- \tag{2.10}
$$

Recall that in SRPM-ST, $\psi_1(x)$ (which is described by $\hat{\theta}_1^{\text{SRPM}}$) assigns pseudo-labels for $\mathcal{U}_2$. This results in m_2^+ positively pseudo-labeled and m_2^- negatively pseudo-labeled samples denoted as $\{x_{i,2}^+\}$ and $\{x_{i,2}^-\}$, respectively, where $m_2^+ + m_2^- = m$. To analyze the quality of produced pseudo-labels for $\mathcal{U}_2$, we can construct estimator of θ using

$$
\hat{\theta}_2^{\text{SRPM}} = \frac{1}{2}\Big(\frac{\sum_{i=1}^{m_2^+} x_{i,2}^+}{m_2^+} + \frac{\sum_{i=1}^{m_2^-} x_{i,2}^-}{m_2^-}\Big). \tag{2.11}
$$

To estimate the optimal θ and $\hat{\theta}_2^{\text{SRPM}}$ using similar inequality given in (2.4), we first need the probability of correctly classifying a sample point from class "+" and the class "−" using $\psi_1(x)$, denoted p_1^+ and p_1^-, respectively. Note that these probabilities are different form p_0^+ and p_0^-. Recall that $\Delta_1 = p_1^+ - p_1^-$. In particular, p_1^+ and p_1^- are given by

$$p_1^+ = P\Big(x > \hat{\theta}_1^{\text{SRPM}} \mid x \in \pi^+\Big), \tag{2.12}$$

$$p_1^- = P\Big(x \leq \hat{\theta}_1^{\text{SRPM}} \mid x \in \pi^-\Big), \tag{2.13}$$

where $\hat{\theta}_1^{\text{SRPM}}$ is described in (2.10). We can show that (the proof is provided in Appendix B)

$$p_1^+ = \Phi\Big(\frac{\delta}{2}(1 - \alpha_0^+ + \alpha_0^-)\Big), \tag{2.14}$$

$$p_1^- = \Phi\Big(\frac{\delta}{2}(1 - \alpha_0^- + \alpha_0^+)\Big), \tag{2.15}$$

where $\Phi(.)$ denotes the cumulative distribution function of a standard normal random variable and $\delta = \frac{\mu^+ - \mu^-}{\sigma}$. Hence, the similar inequality (2.4) for or $\hat{\theta}_2^{\text{SRPM}}$ can be written as

$$\begin{aligned} &P\left(\left|\hat{\theta}_2^{\text{SRPM}} - \frac{\mu^+ + \mu^-}{2} - \Delta_1 \frac{\mu^+ - \mu^-}{2}\right| \geq \epsilon\right) \leq \\ &\leq 2e^{-\frac{2\epsilon^2}{9\sigma^2}\frac{m_2^+ m_2^-}{m_2^+ + m_2^-}} + 2e^{-\frac{8m_2^+\epsilon^2}{9(\mu^+ - \mu^-)^2}} + 2e^{-\frac{8m_2^-\epsilon^2}{9(\mu^+ - \mu^-)^2}}, \end{aligned} \tag{2.16}$$

for any $\epsilon > 0$. We note that for large m_2^+ and m_2^- (based on Assumption 2.2.2), the right-hand-side of (2.16) converges to 0. Therefore, we can write

$$\hat{\theta}_2^{\text{SRPM}} \approx \theta + \Delta_1 \frac{\mu^+ - \mu^-}{2}. \tag{2.17}$$

In contrast, conventional ST utilizes $\psi_0(x)$ to assign pseudo-labels for both $\mathcal{U}_1$ and $\mathcal{U}_2$ indifferently. Hence, we construct the estimator of θ, denoted as $\hat{\theta}_2^{\text{ST}}$, using $\mathcal{U}_2$ and their pseudo-labels. In particular, we have

$$\hat{\theta}_2^{\text{ST}} \approx \theta + \Delta_0 \frac{\mu^+ - \mu^-}{2}. \tag{2.18}$$

Figure 2.1 presents the numerically comparison of Δ_0 and Δ_1 for a wide range of δ, p_0^-, and p_0^+ as it is not straightforward to analytically compare them. Particularly, Figure 2.1 demonstrates $|\Delta_0| - |\Delta_1|$ as a function of p_0^- using three different values of δ and p_0^+, where

$$|\Delta_0| - |\Delta_1| = |p_0^+ - p_0^-| - \left|\Phi\Big(\frac{\delta}{2}(1 - \alpha_0^+ + \alpha_0^-)\Big) - \Phi\Big(\frac{\delta}{2}(1 - \alpha_0^- + \alpha_0^+)\Big)\right|. \tag{2.19}$$

We observe that $|\Delta_0| \geq |\Delta_1|$, and therefore we can conclude that $\hat{\theta}_2^{\text{SRPM}}$ is a better estimator of θ than $\hat{\theta}_2^{\text{ST}}$. Recall that the quality of estimation of θ is employed as proxy for the quality of produced pseudo-labels.

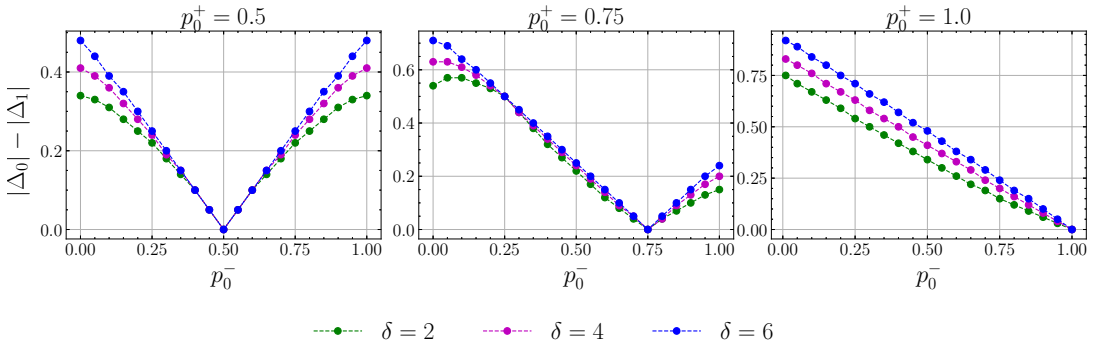

Figure 2.1: Values of $|\Delta_0| - |\Delta_1|$ as a function of p_0^- for different δ and p_0^+.

2.3 Method: SRPM-ST

Given a supervised classification algorithm, a small-labeled dataset $\mathcal{L}$ with size l and a large unlabeled dataset $\mathcal{U}$ with size u, the pipeline of SRPM-ST is as follows: 1) an initial classifier is trained using $\mathcal{L}$; 2) $\mathcal{U}$ is randomly split into $B = \left\lfloor \frac{u}{m} \right\rfloor$ mini-batches of size m; 3) each mini-batch of unlabeled data is successively pseudo-labeled; and 4) the classifier is retrained after each pseudo-labeled mini-batch of unlabeled data. The step-by-step implementation is presented in Algorithm 2.

In practice, the size of the mini-batches in step (2) is data-dependent and therefore can be considered as a hyperparameter. Hence, we also examine the applicability of a univariate search-based K-fold cross-validation strategy adapted to ST (we refer to it as ST K-fold CV) to estimate the optimal mini-batch size. The pipeline of ST K-fold CV is as follows: (i) a labeled set is split into K folds; (ii) Algorithm 2 is applied on $K-1$ folds and the unlabeled set to train a surrogate classifier; (iii) the surrogate classifier is evaluated on the held-out fold; (iv) steps (ii)-(iii) are repeated K times, with each fold used once to evaluate the classifier; and (v) the average of the performance of the classifier across each held-out fold is computed. We note that ST K-fold CV can be seamlessly included into common search procedures to optimize multiple hyperparameters, including the mini-batch size. For example, with a "random" search or "grid" search. The step-by-step implementation of ST K-fold CV is detailed in Appendix C.

2.4 Experimental settings

In the numerical experiments we: 1) compare the performance of the classifiers that are trained using F-ST and SRPM-ST, and 2) examine the capability of ST K-fold CV in estimating the optimal mini-batch size in SRPM-ST.

Algorithm 1 ST

Input:
an initial labeled set $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ of size l
an unlabeled set $\mathcal{U} = \{\mathbf{x}_i\}_{i=1}^{u}$ of size u
Parameters:
a classification algorithm Ψ
a mini-batch size m
Return:
classifier ψ

Initialize training set: $\tilde{\mathcal{S}} \leftarrow \mathcal{L}$
Use Ψ to train classifier ψ on $\tilde{\mathcal{S}}$
Split $\mathcal{U}$ into $B = \left\lfloor \frac{u}{m} \right\rfloor$ mini-batches: $\mathcal{U}_b = \{\mathbf{x}_i\}_{i=(b-1)m+1}^{bm}$, $b = 1, \ldots, B$
for $b = 1, \ldots, B$ **do**
    Generate pseudo-labels: $\tilde{y}_i \leftarrow \psi(\mathbf{x}_i)$ for $\mathbf{x}_i \in \mathcal{U}_b$
    Construct pseudo-labeled set: $\tilde{\mathcal{U}}_b \leftarrow \{(\mathbf{x}_i, \tilde{y}_i) \mid \mathbf{x}_i \in \mathcal{U}_b\}$
    Update the training set: $\tilde{\mathcal{S}} \leftarrow \tilde{\mathcal{S}} \cup \tilde{\mathcal{U}}_b$
    Retrain the classifier ψ on $\tilde{\mathcal{L}}$
end for
return ψ

Algorithm 2 SRPM-ST

Input:
an initial labeled set $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ of size l
an unlabeled set $\mathcal{U} = \{\mathbf{x}_i\}_{i=1}^{u}$ of size u
Parameters:
a classification algorithm Ψ
a mini-batch size m
Return:
classifier ψ

Initialize training set: $\tilde{\mathcal{S}} \leftarrow \mathcal{L}$
Use Ψ to train classifier ψ on $\tilde{\mathcal{S}}$
Split $\mathcal{U}$ into $B = \left\lfloor \frac{u}{m} \right\rfloor$ mini-batches: $\mathcal{U}_b = \{\mathbf{x}_i\}_{i=(b-1)m+1}^{bm}$, $b = 1, \ldots, B$
for $b = 1, \ldots, B$ **do**
    Generate pseudo-labels: $\tilde{y}_i \leftarrow \psi(\mathbf{x}_i)$ for $\mathbf{x}_i \in \mathcal{U}_b$
    Construct pseudo-labeled set: $\tilde{\mathcal{U}}_b \leftarrow \{(\mathbf{x}_i, \tilde{y}_i) \mid \mathbf{x}_i \in \mathcal{U}_b\}$
    Update the training set: $\tilde{\mathcal{S}} \leftarrow \tilde{\mathcal{S}} \cup \tilde{\mathcal{U}}_b$
    Retrain the classifier ψ on $\tilde{\mathcal{L}}$
end for
return ψ

2.4.1 Dataset

We collected eight real datasets from OpenML dataset repository. Table 2.1 presents the summary of datasets.

Table 2.1: Summary of datasets.

Dataset	No. of features	No. of samples	No. of classes
Handwritten digits (`mnist`)	784	70,000	10
Handwritten digits (`usps`)	256	9,298	10
Skin segmentation (`skin`)	3	245,057	2
Human activity recognition (`har`)	561	10,299	6
Churn prediction (`churn`)	20	5,000	2
Texture classification (`texture`)	40	5,500	11
White wine quality (`wine`)	11	4,898	7
KDDCup upselling (`ups`)	45	5,032	2

2.4.2 Supervised Learning Algorithm

For both SRPM-ST and F-ST, we use five classification algorithms: k-Nearest Neighbors (KNN) (Fix and Hodges, 1989), Linear Discriminant Analysis (LDA) (Anderson, 1951), Linear Support Vector Machine (LSVM) (Vapnik, 1995), Logistic Regression (LRR) (Anderson and Richardson, 1979), and Random Forest (RF) (Breiman, 2001). We excluded the effect of tuning hyperparameters of all classifiers by using the predefined values of `scikit-learn` library (Pedregosa et al., 2011) (as of version 1.2.2). That is to say, the comparison between SRPM-ST and F-ST was conducted using identical hyperparameter values for all classifiers. The summary of fixed hyperparameter values is shown in Table 2.2.

Table 2.2: Summary of fixed hyperparameter values.

Model	Parameter	Value
LDA	solver	singular value decomposition
	absolute threshold of for singular values	10^{-4}
LRR	regularization parameter C	1
	penalty	L_2
LSVM	regularization parameter C	1
	penalty	L_2
	loss	squared hinge
KNN	number of nearest neighbors	5
RF	number of trees	100
	maximum depth	fully grown trees
	maximum number of features	square root of number of features

2.4.3 Evaluation

To perform experiments, we randomly split each dataset into two disjoint sets with a $90 : 10\%$ ratio, where the smaller portion designated as the test set. The classifiers trained using SRPM-ST and F-ST are evaluated on this test set. To simulate ST scenario (i.e., having labeled and unlabeled sets), the larger portion was further split into two disjoint sets to form the unlabeled and labeled sets. To have the effect of a large number of unlabeled data points with respect to a small number of labeled data points, we consider two scenarios with $99 : 1\%$ and $98 : 2\%$ splitting ratios. Hence, we have a total of 80 comparison cases, derived from 8 datasets, 5 classifiers, and 2 data splitting ratios. All splits were performed in a stratified manner, that is, the class proportions in all splits were as in the entire dataset.

To assess SRPM-ST, we perform experiments with different mini-batch sizes, which are determined as $m_i = \left\lfloor \frac{u}{2^i} \right\rfloor$, where $i = 0, \ldots, 6$. Note that for $i = 0$, SRPM-ST becomes equivalent to F-ST. Additionally, we perform an estimation of optimal mini-batch size using ST K-fold CV. In particular, a comparison between test-set and CV accuracies sheds light on the applicability of ST K-fold CV in estimating the optimal mini-batch size.

We repeat the experiments 30 times to obtain the average test-set accuracies using different labeled, unlabeled, and test sets.

2.5 Results

Figures 2.2 and 2.3 demonstrate the average test-set accuracy of SRPM-ST and the estimate of the accuracy of ST K-fold CV across different mini-batch sizes for all datasets and classifiers. Specifically, Figures 2.2 and 2.3 contain plots with two straight (solid) lines and four (dashed or dotted) curves. The straight lines show the average test-set accuracy of an initial classifier, denoted a_0, that was trained merely on labeled set. The average classification performance of the classifier obtained using SRPM-ST is depicted by dashed curve in each plot. Recall that in each plot, $i = 0$ corresponds the average classification performance of the classifier obtained using F-ST. The estimate of the accuracy of ST K-fold CV (STCV) is shown using dotted curve. Each straight line and curve is given in two colors: purple and green colors that distinguishes $u : l = 99 : 1$ and $98 : 2$ cases, respectively. In addition, in each curve the optimal mini-batch size (i.e., the highest accuracy) marked with bullet point $\bullet$. For ease of visualization Figures 2.2 and 2.3 do not show the error bars. Similar figures with error bars can be found in Appendix D.

The results show that:

1. in 96.2% of the cases (77 out of 80), the maximum of the average test-set accuracy

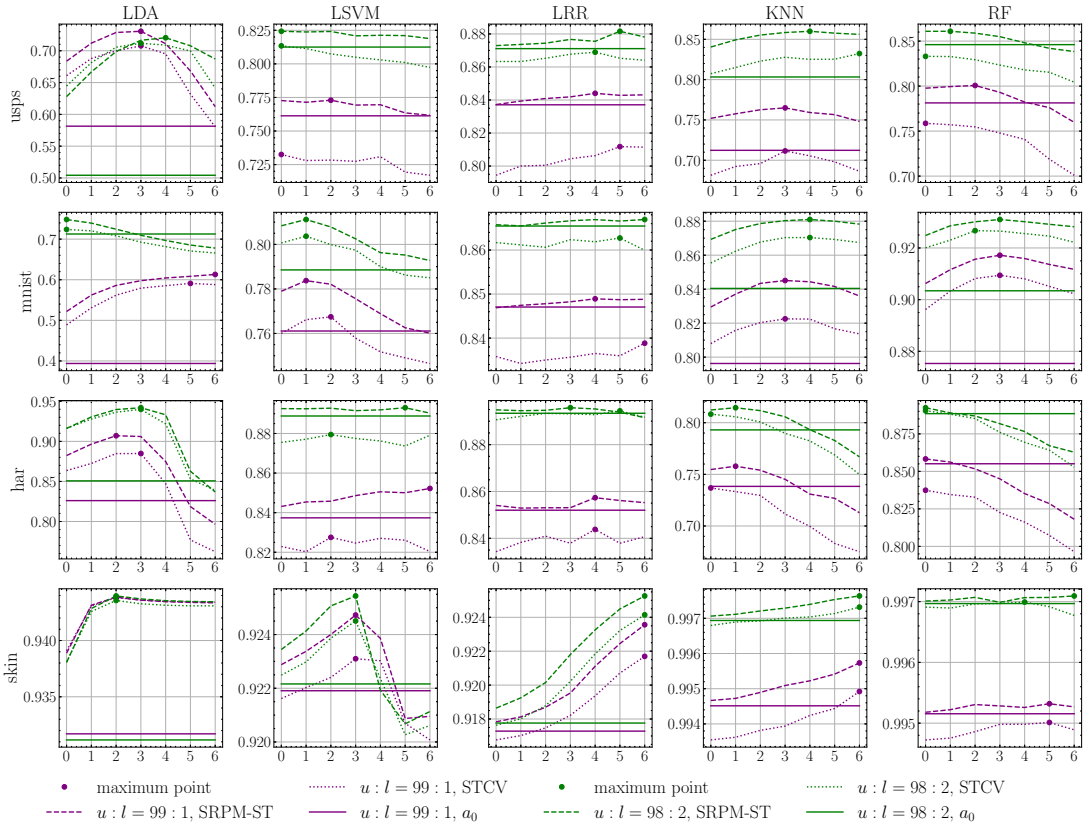

Figure 2.2: The average classification results in terms of test-set accuracy across different mini-batch sizes for `usps`, `mnist`, `har`, and `skin`, datasets and all considered classifiers.

(over mini-batch sizes) obtained using SRPM-ST is higher than a_0. That is to say, there exists a mini-batch size that leads to better performance, on average, using SRPM-ST compared to the initial classifier in each case. In particular cases, the improvement of performance is remarkable. For example, the difference between the maximum average test-set accuracy of SRPM-ST and a_0 could reach $\sim$33%, $\sim$22%, and $\sim$22% in `texture`, `mnist`, and `usps` datasets using LDA classifier, respectively;

2. in 85% of the cases (68 out of 80), the maximum of the average test-set accuracy of the classifier constructed using SRPM-ST, is greater than the accuracy of the classifier obtained using F-ST. For example, on average, the accuracy performance may improve by $\sim$16%, $\sim$9%, and $\sim$9% for `texture`, `mnist`, and `usps` datasets using LDA classifier, respectively; and

3. in 50% of the cases (40 out of 80), the optimal mini-batch size estimated using STCV matches the mini-batch size that leads to the maximum average test-set accuracy of SRPM-ST. To the extent that the results of STCV can be analyzed, they indicate that the estimated value of the optimal mini-batch size coincides with the optimal mini-batch size in half of the considered cases. This is a considerable

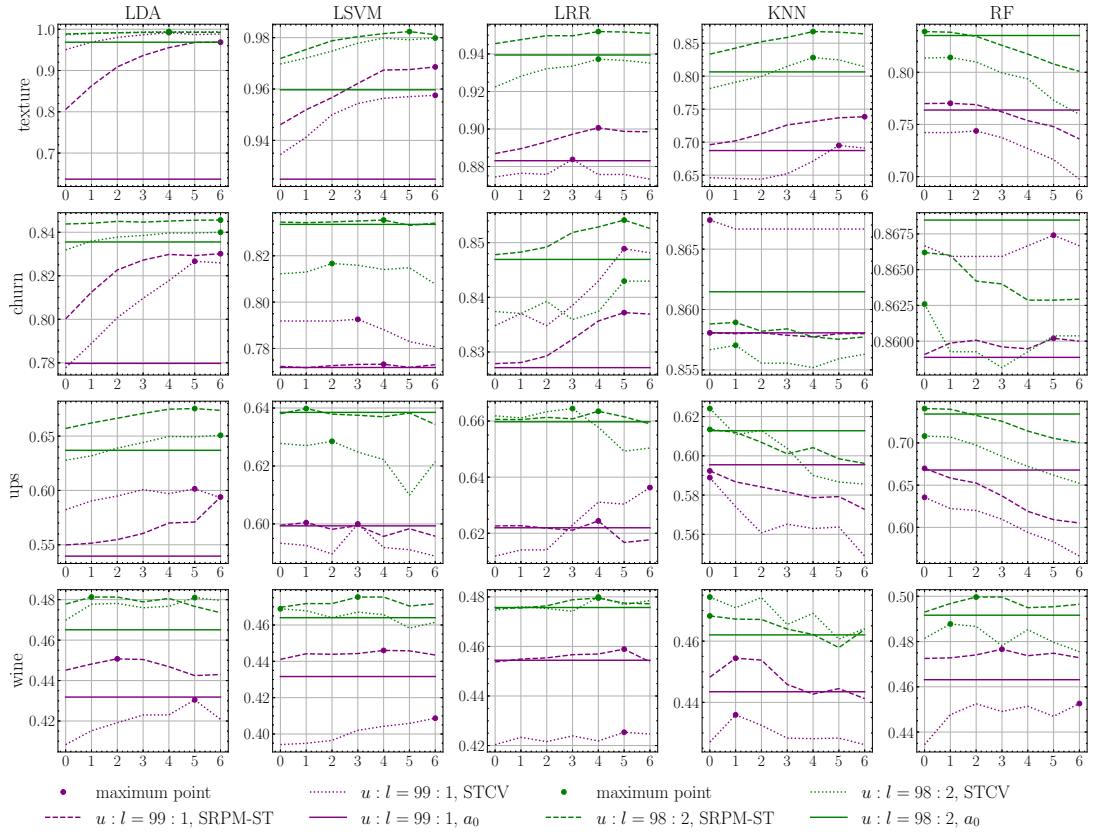

Figure 2.3: The average classification results in terms of test-set accuracy across different mini-batch sizes for `texture`, `churn`, `ups`, and `wine`, datasets and all considered classifiers.

result considering the variance of resampling-based error estimation methods such as CV.

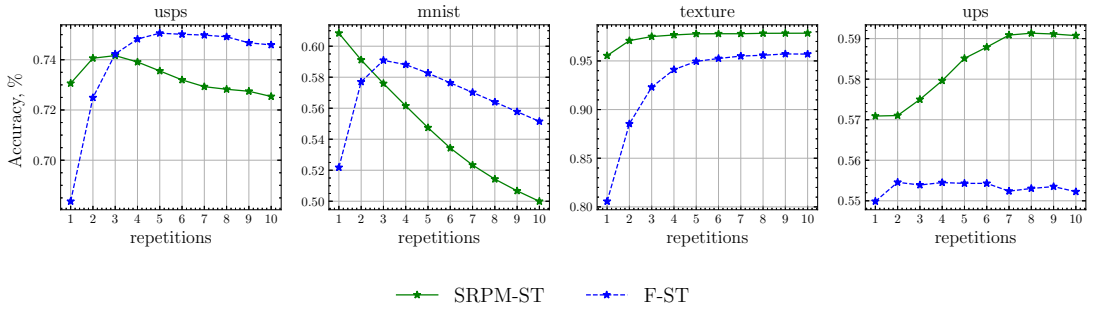

Figure 2.4: The average performance in terms of test-set accuracy of the final classifiers trained using SRPM-ST and F-ST across 10 repetitions for `usps`, `mnist`, `texture`, and `ups` datasets.

2.6 Discussion

On the effect of repetition. As discussed in Section 2.1, based on Assumption 2.1.2, there is no repetition in SRPM-ST and F-ST frameworks after all unlabeled data

points are assigned with pseudo-labels and merged with the labeled set. In order to assess how repetition influences the performance of both algorithms, we performed the experiment with ten repetitions and compared their performances. The experimental results are presented in Figure 2.4, where the average test-set accuracy is shown across ten repetitions for $u : l = 99 : 1$ case.The experimental setup followed a similar procedure to that described in Section 2.4, except that we trained a classifier using SRPM-ST with the mini-batch size that was obtained using ST K-fold CV. The results show that:

- in `usps` dataset, F-ST shows better results than SRPM-ST starting from the 4^{th} repetition;
- in `mnist` dataset, there is a degradation in the performance of both algorithms, with F-ST after three repetitions and SRPM-ST after one repetition;
- in `texture` dataset, despite both methods follow the same trend, SRPM-ST outperforms F-ST; and
- in `ups` dataset, the performance of F-ST saturates and does not reach SRPM-ST.

In summary, we can note that although repeating the F-ST several times may improve the performance, F-ST may not reach the level of performance SRPM-ST trained without repetitions. Furthermore, in practice, repetition of F-ST (and SRPM-ST) without knowing the optimal number of repetitions requires additional criteria to terminate training. This is because repetition is prone to noise accumulation, as it may reinforce the effect of incorrect pseudo-labels generated in previous repetition. This is can be seen in Figure 2.4, where the classifier performance may degrade over repetitions.

The size of the mini-batch in SRPM-ST. As discussed in Section 2.4.3, the considered set of candidate mini-batch size values was determined exponentially as $\left\lfloor \frac{u}{2^i} \right\rfloor$, for $i = 0, \ldots, 6$. The selection of such a modest number of candidate values was mainly constrained by the feasibility of computation to obtain the results of SRPM-ST and examine the applicability of the CV strategy for all cases and repetitions. Nevertheless, the optimal mini-batch size can be estimated using a larger number of mini-batch sizes. However, even with a small number of candidate values, the results confirm that there is a data-dependent mini-batch size that can make SRPM-ST outperform F-ST.

Optimal mini-batch size estimation using ST K-fold CV. We examined the capability of ST K-fold CV in estimating the optimal mini-batch size parameter employed in SRPM-ST. In 50% of the considered cases, the mini-batch size estimated using ST K-fold CV is identical with the optimal one within the candidate mini-batch sizes. Nevertheless, we note that this result depends on the data and the candidate

values. However, the result also shows that in almost all cases (38 out of 40), where the optimal and estimated mini-batch size did not match, the difference of corresponding the average test-set accuracies was less than one percent. This observation suggests that, in practice, the utility of ST K-fold CV seems to be a reliable estimator of a "good" mini-batch size.

Table 2.3: The test-set accuracy of SRPM-ST and F-ST with Mean Teacher and FlexMatch using `usps` dataset.

	Mean Teacher	FlexMatch
F-ST	88.5	79.1
SRPM-ST ($B = 2$)	90.2	89.0
SRPM-ST ($B = 3$)	92.5	86.7

SRPM-ST with SSL algorithms. To test the applicability of SRPM-ST as a wrapper method for other semi-supervised algorithms, we conducted numerical experiments with Mean Teacher (Tarvainen and Valpola, 2017) and FlexMatch (Zhang et al., 2021) semi-supervised image classification algorithms with `usps` dataset. The experimental pipeline was as follows: (i) the unlabeled data is randomly split into $B = 2, 3$ mini-batches of size m; (ii) the labeled data and a mini-batch of unlabeled data is used to train Mean Teacher or FlexMatch classifier; (iii) pseudo-labels for that mini-batch of unlabeled data are assigned using the obtained classifier; (iv) pseudo-labeled mini-batch of unlabeled data is added to the labeled data; and (v) steps (ii)-(iv) are repeated until all mini-batches of unlabeled data are used. The data splitting strategy was similar to Section 2.4.1. We fixed the parameters of both algorithms as suggested in (Tarvainen and Valpola, 2017; Zhang et al., 2021), except for the batch size of 32 and the use of 200 training iterations per epoch. We trained both classifiers with a Wide ResNet-28-2 network (Zagoruyko and Komodakis, 2016) and set the epochs to 10 and 15 for FlexMatch and Mean Teacher, respectively. Table 2.3 presents the accuracy results on the test set of Mean Teacher and FlexMatch with F-ST (i.e., their original forms) and SRPM-ST using `usps` dataset in one random split for $u : l = 99 : 1$ case. The results show that using SSL algorithms with SRPM-ST can lead to improved performance compared to using F-ST. In particular, the use of SRPM-ST resulted in $\sim 4.0\%$ and $\sim 9.9\%$ test-set accuracy improvement in comparison with F-ST with Mean Teacher and FlexMatch, respectively.

Limitations. The main limitation of the ST framework is attributed to noise accumulation, in which the cumulative effect of incorrectly generated pseudo-labels can lead to performance degradation. The results suggest that SRPM-ST, which uses mini-batch-wise retraining and pseudo-labeling seems more resilient to noise accumulation compared to F-ST. Nevertheless, the common approaches to address this issue are based

on using some thresholding strategy. This strategy is also applicable for SRPM-ST, for example, by selecting a subset of pseudo-labeled mini-batch to retrain the classifier and repeating this process for each mini-batch. The investigation of such approaches is left for future studies.

Another limitation can be attributed to the computational expenses of tuning the mini-batch size. Specifically, mini-batch size is tuned using K-fold CV, which runs SRPM-ST K times. That being said, running K-fold CV in parallel settings can reduce the computational expenses.

Order of mini-batches. The numerical experiments reveal that we can expect better results using SRPM-ST compared to the use of F-ST, even though the splitting of the unlabeled data into mini-batches was performed randomly. Nevertheless, an important direction for further investigation corresponds to finding the "successful" or optimal order of mini-batches to be used in SRPM-ST. Since the quality of the pseudo-labels produced in the first min-batches generally will affect the quality of pseudo-labels in subsequent min-batches. Hence, the better the quality of pseudo-labels, therefore the better the classifier.

2.7 Summary

In this chapter, we presented a novel approach to the self-training framework, which is sequentially retraining and pseudo-labeling in mini-batches (SRPM-ST). Particularly, SRPM-ST generates pseudo-labels and retrains the classifier using the mini-batches one after another. In contrast, full-batch self-training (F-ST) pseudo-labels the entire unlabeled set at once. The experimental results demonstrated that SRPM-ST shows better results than F-ST in 85% of cases investigated here.

Chapter 3

BSSL++: A Boosting Algorithm for Semi-Supervised Learning Inspired by Learn++

This chapter continues the investigation of using unlabeled data in the semi-supervised learning (SSL) framework and focuses on the mitigation of the noise accumulation phenomenon.

3.1 Introduction

Boosting constructs an ensemble classifier by sequentially learning base classifiers. Similar to self-training (ST), boosting can be readily extended to the semi-supervised setting by using an ensemble classifier constructed so far to generate *pseudo-labels* for unlabeled data and use them in training subsequent base classifiers. In this regard, given the efficacy of ensemble boosting techniques and their inherent dependency between base learners, they have been successfully extended to SSL (Bennett et al., 2002; d'Alché-Buc et al., 2001; Mallapragada et al., 2009; Chen and Wang, 2011; Wu et al., 2018; Guo et al., 2015).

Some of the early SSL algorithms that utilized boosting correspond to ASSEMBLE (Bennett et al., 2002) and Semi-Supervised MarginBoost (d'Alché-Buc et al., 2001). These semi-supervised boosting algorithms were designed based on the *cluster assumption*. Nonetheless, they used both labeled and unlabeled data to minimize the total classification margin cost. To be specific, the algorithms use an iterative procedure where at each boosting iteration: (i) the ensemble classifier constructed so far is used to generate pseudo-labels for unlabeled data; (ii) the sample weights of labeled and pseudo-labeled data with lower classification *margin* (i.e., that are closer to current decision boundary) are increased; (iii) these weights are used to train the next base classifier; and (iv) all base classifiers are combined linearly to form the ensemble. However, the

drawback of this approach is attributed to entirely depending on the definition of margin and margin costs for the labeled and unlabeled data. Subsequently, at each boosting iteration, even incorrectly classified unlabeled data points may have high confident predictions (i.e., have lower margin costs) (Hertz et al., 2004). As a result, additional mechanisms are required besides minimum margin cost.

State-of-the-art SSL boosting algorithms, namely, SemiBoost (Mallapragada et al., 2009) and RegBoost (Chen and Wang, 2011), leveraged pairwise similarity between labeled and unlabeled data points in addition to minimizing the margin cost. Particularly, Mallapragada *et al.* (Mallapragada et al., 2009) proposed to calculate the confidence scores using similarity knowledge between data points alongside predictions of ensemble classifier. Based on these scores, they generate pseudo-labels and identify a subset of pseudo-labeled data to select for training a subsequent base learner. On the other hand, Chen and Wang (Chen and Wang, 2011) proposed regularized semi-supervised boosting (RegBoost) algorithm, in which they introduced a regularization penalty on unlabeled data utilizing three main SSL assumptions and pairwise affinity information in addition to margin cost on labeled data. That being said, since the generation of pseudo-labels in each boosting iteration largely depends on previously assigned pseudo-labels (via the classifier constructed so far), incorrectly generated pseudo-labels may be potentially reinforced in subsequent iterations. Hence, without an effective approach, the algorithm can construct classifiers that are more confident about inaccurate predictions over boosting iterations—which corresponds to noise accumulation.

In this chapter, we focus on reducing the dependence on the ensemble classifier constructed so far in order to alleviate the noise accumulation. As a result, inspired by an incremental supervised boosting technique namely, Learn++ (Polikar et al., 2001), we introduce BSSL++, which is a novel semi-supervised boosting algorithm. In principle, Learn++ (inspired by AdaBoost (Freund and Schapire, 1996)) incrementally learns and constructs an ensemble classifier using a batch of data that arrives in a stream. That is to say, Learn++ does not need to have access to the previously used training data. That being said, BSSL++ utilizes the *semi-supervised smoothness* and *cluster assumptions* to direct both the selection and the utility of unlabeled data in the training process.

In order to guard against effect of noise accumulation, at each iteration, the following approach is considered: (i) clustering the data (both labeled and selected unlabeled) and turning these clusters into meaningful classes based on the original labeled data; (ii) using these classes to assign “initial" pseudo-labels for a selected unlabeled data; and (iii) using these newly pseudo-labeled data with previously pseudo-labeled and labeled data to train an AdaBoost-like classifier. At the end of this process, we update *all* generated pseudo-labels by the ensemble classifier constructed so far. These updated pseudo-labels are then partially used in training the AdaBoost-like classifier in the next

iteration. As a result, we reduce dependence on the ensemble classifier and yet we do not diminish the knowledge learned so far.

3.2 Method: Learn++

Suppose that a labeled dataset $\mathcal{L}$ arrives in form of batches $\mathcal{L}_b, b = 1, \ldots, B$, where $\mathcal{L}_b = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l_b}$ of size l_b, in which $\mathbf{x}_i \in \mathbb{R}^p$ is a p-dimensional feature vector and $y_i \in \mathbb{R}$ is the corresponding label.

The algorithmic overview of Learn++ is as follows (Polikar et al., 2001). For each batch of dataset $\mathcal{L}_b$, Learn++ constructs an ensemble classifier. In particular, at each boosting iteration $t = 1, \ldots, T$, a training set $\mathcal{L}_b^{tr}$ is drawn from $\mathcal{L}_b$ according to sample weights $w_{t,b}(i), i = 1, \ldots, l_b$. Note that when $t = 1$, all data points in the training set are set to have equal weights. Next, a base classifier $h_{t,b} : \mathcal{X} \to \mathcal{Y}$ is trained using $\mathcal{L}_b^{tr}$, where $\mathcal{X}$ and $\mathcal{Y}$ are the input and the output spaces, respectively. The error of $h_{t,b}$ is calculated on the entire training set $\mathcal{L}_b$ as

$$\epsilon_{t,b} = \sum_{\mathbf{x}_i \in \mathcal{L}_b} w_{t,b}(i) I_{\{h_{t,b}(\mathbf{x}_i) \neq y_i\}} \,, \tag{3.1}$$

where $I_{\{a\}}$ is an indicator variable that equals 1 if a is true and 0 otherwise. The normalized error $\beta_{t,b}$ is computed as $\beta_{t,b} = \epsilon_{t,b}/(1 - \epsilon_{t,b})$. At the end of each boosting iteration t weighted majority vote classifier, denoted $H_{t,b}$, is constructed as

$$H_{t,b}(\mathbf{x}) = \underset{y \in Y}{\operatorname{argmax}} \sum_{j=1}^{t} \log(\frac{1}{\beta_{j,b}}) I_{\{h_{j,b}(\mathbf{x}) = y\}} \,, \tag{3.2}$$

where $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$. Then, sum of the sample weights of misclassified observations by $H_{t,b}$ is used to compute its error, denoted $E_{t,b}$. In particular,

$$E_{t,b} = \sum_{\mathbf{x}_i \in \mathcal{L}_b} w_{t,b}(i) I_{\{H_{t,b}(\mathbf{x}_i) \neq y_i\}} \,. \tag{3.3}$$

The normalized error of $H_{t,\kappa}$ is obtained by $\mathcal{B}_{t,b} = E_{t,\kappa}/(1 - E_{t,b})$. At this point, the sample weights are updated such that in subsequent iterations, an additional classifier is learned by focusing more on incorrectly classified data points. Specifically, the weight update rule in an analogy with AdaBoost is

$$w_{t+1,b}(i) = w_{t,b}(i) \times \begin{cases} \mathcal{B}_{t,b}, & \text{if } H_{t,b}(\mathbf{x}_i) = y_i \,, \\ 1, & \text{otherwise} \,. \end{cases} \tag{3.4}$$

In order to interpret the updated sample weights as an appropriate probability distribution (of being selected in subsequent iteration), they are generally normalized.

As soon as all batches of dataset $\mathcal{L}_b$, $b = 1, \ldots, B$ are employed in the training procedure, the final classifier is given as

$$H_{\text{final}}(\mathbf{x}) = \underset{y \in Y}{\operatorname{argmax}} \sum_{b=1}^{B} \sum_{t=1}^{T} \log(\frac{1}{\mathcal{B}_{t,b}}) I_{\{H_{t,b}(\mathbf{x}) = y\}} \,. \tag{3.5}$$

The stopping criteria for terminating the boosting iterations is similar to the AdaBoost, where the weak classifier is expected to perform not worse than random guessing, that is, in the multi-class classification task, $\epsilon_{t,\kappa} < (k-1)/k$, where k is the number of classes.

3.3 Method: BSSL++

Given a small-labeled dataset $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ of size l, and a large size unlabeled dataset $\mathcal{U} = \{\mathbf{x}_i\}_{i=1}^{u}$ of size u, BSSL++ iteratively builds an ensemble (AdaBoost-like classifier) of ensemble classifiers similarly to Learn++. In particular, BSSL++ proceeds as follows: (1) constructs a training set consisting of labeled and selected unlabeled data, denoted as $\tilde{S}$, that initially comprises only labeled data; (2) divides $\tilde{S}$ into clusters and assigns class labels to these clusters based on the original labeled data; (3) selects nearest neighbors of $\tilde{S}$ among remaining unlabeled data; (4) generates "initial" pseudo-labels for selected unlabeled data using class labels of clusters; (5) adds the pseudo-labeled unlabeled data to $\tilde{S}$ and removes them from unlabeled set $\mathcal{U}$; (6) constructs AdaBoost-like ensemble classifier using a similar approach as Learn++; (7) updates all pseudo-labels that are generated at current iteration as well as all previously generated pseudo-labels using the ensemble classifier constructed so far; and (8) repeats steps (2)-(7) until all unlabeled data are pseudo-labeled and utilized during the training.

BSSL++ selects and assigns pseudo-labels based on guidelines of SSL smoothness and cluster assumptions. Particularly, the selection procedure is directed by employing the distance (e.g., Manhattan distance) between labeled and unlabeled data points. That is, at each iteration, BSSL++ selects the nearest unlabeled data point to each labeled (or pseudo-labeled) data point in $\tilde{S}$ (steps 3 and 4). Let $\mathbf{x}^{(1)}(\mathbf{x}_i)$ denote the (first) nearest neighbor of $\mathbf{x}_i \in \tilde{S}$ among $\mathbf{x}_j \in \mathcal{U}$. Then, the selected unlabeled data points compose a set $\mathcal{U}_{NN}$, that is,

$$\mathcal{U}_{NN} = \cup_{i=1}^{s} \{\mathbf{x}^{(1)}(\mathbf{x}_i)\}\,, \tag{3.6}$$

where s is the size of set $\tilde{S}$.

At the same time, the "initial" pseudo-labeling procedure is directed by partitioning $\tilde{S}$ into k clusters (k is set to the number of classes in labeled data) utilizing the k-means++ clustering algorithm (Arthur and Vassilvitskii, 2007). We note that, in the first iteration, class-specific sample means are used to obtain centroids. Let $\xi_1, \ldots, \xi_k$ denote the obtained k clusters with centroids $o_1, \ldots, o_k$, respectively. Let l_c denote the number of labeled data points from class c in $\tilde{S}$, where $c = 1, \ldots, k$, that is

$$l_c = \Big|\{\mathbf{x}_j \in \tilde{S} \mid \mathbf{x}_j \text{ belongs to class } c\}\Big|\,, \quad c = 1, \ldots, k, \tag{3.7}$$

where $|.|$ denotes the cardinality of the set. Let p_{cz} denote the number of labeled data points from class c in cluster ξ_z, where c and $z = 1, \ldots, k$, that is

$$p_{cz} = |\{\mathbf{x}_j \in \mathcal{L} \mid \mathbf{x}_j \text{ belongs to cluster } \xi_z\}|\,, \quad c, z = 1, \ldots, k. \tag{3.8}$$

Then the cluster ξ_z is assigned with a class label $\hat{y}_z$ based on:

$$\hat{y}_z = \underset{c}{\operatorname{argmax}} \frac{p_{cz}}{l_c}, \quad z = 1, \ldots, k\,. \tag{3.9}$$

With the class labels assigned to clusters, initial pseudo-label $\tilde{y}_j$ is generated by identifying the cluster to which $\mathbf{x}_j \in \mathcal{U}_{NN}$ belongs. Since we employed k-means++ to generate clusters in the first place, we assign a pseudo-label for $\mathbf{x}_j$ based on the assigned class label of the nearest cluster centroid. It is worth noting that the assigned pseudo-label is independent of both the current ensemble classifier and pseudo-labels of previously selected data points. This is purposely encoded into BSSL++ to mitigate the effect of noise accumulation. Let $\tilde{\mathcal{U}}_{NN}$ denote the set of pairs of data points in $\mathcal{U}_{NN}$ and their pseudo-labels; that is,

$$\tilde{\mathcal{U}}_{NN} = \{(\mathbf{x}_j, \tilde{y}_j) \mid \mathbf{x}_j \in \mathcal{U}_{NN}\}\,. \tag{3.10}$$

The set $\tilde{S}$ is then updated by adding $\tilde{\mathcal{U}}_{NN}$, and $\mathcal{U}_{NN}$ is removed from the unlabeled set $\mathcal{U}$ (step 5). Next, $\tilde{S}$ is used to train an AdaBoost-like ensemble classifier using a similar approach as described in Section 3.2 (step 6). At this point, we update all generated pseudo-labels, including the initial pseudo-labels generated at the current iteration and pseudo-labels generated in previous iterations using the ensemble classifier constructed so far, denoted H_{final}. The ensemble classifier is constructed by combining the previously trained ensemble classifiers using a weighted majority vote (as in Equation 3.5) (step 7). In the following iteration, the updated pseudo-labeled data are partially utilized to train the successive AdaBoost-like classifier. Algorithm 3 presents the step-by-step implementation of BSSL++.

Remark: In the first iteration of boosting, instead of initializing the sample weights to be equal for all s data points in set $\tilde{S}$, inspired by ASSEMBLE (Bennett et al., 2002), we encode the importance of the labeled data points by setting the weights of the labeled data to be higher than the unlabeled data. That is, in each iteration b, the initial sample weights $w_{1,b}(i), i = 1, \ldots, s$ are generated by

$$w_{1,b}(i) = \begin{cases} \gamma_l/l, & \text{if } \mathbf{x}_i \in \mathcal{L}\,, \\ (1-\gamma_l)/(s-l), & \text{otherwise}\,, \end{cases} \tag{3.11}$$

where γ_l is a tuning parameter that denotes the importance coefficient of labeled data.

Algorithm 3 BSSL++

Input:
an initial labeled set $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ of size l and an unlabeled set $\mathcal{U} = \{\mathbf{x}_i\}_{i=1}^{u}$ of size u
a base learning algorithm Ψ
a number of classes k
a k-means++ clustering algorithm Ξ^{++}
Parameter:
a number of base learners T
a distance metric d (default L_1)
a tuning parameter γ_l (default 0.9)
Output:
an ensemble of ensemble classifier H_{final}

$b \leftarrow 1$ {while loop counter}
$\tilde{S} \leftarrow \mathcal{L}$
Compute adjacency matrix $\mathbf{A}$ where
  $\mathbf{A}_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$ for each $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{L} \cup \mathcal{U}$
while $U \neq \emptyset$ **do**
  Call PLNS to obtain nearest unlabeled samples with and without pseudo-labels
  $\tilde{\mathcal{U}}_{NN}, \mathcal{U}_{NN} \leftarrow \text{PLNS}(\tilde{S}, \mathcal{U}, \Xi^{++}, \mathbf{A}, k, b)$
  Remove selected unlabeled data $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{U}_{NN}$
  Update training set $\tilde{S} \leftarrow \tilde{S} \cup \tilde{\mathcal{U}}_{NN}$
  Call ENSEMBLE to train AdaBoost-like classifier on $\tilde{S}$
  $H_{T_b,\kappa} \leftarrow \text{ENSEMBLE}(\tilde{S}, \Psi, k, b, T, \gamma_l)$
  Construct ensemble classifier $H_{\text{final}}(\mathbf{x})$ using (3.5)
  Generate new pseudo-labels $\hat{y}_i \leftarrow H_{\text{final}}(\mathbf{x}_i)$ if $\mathbf{x}_i \in \tilde{S} \setminus \mathcal{L}$
  Construct new pseudo-labeled set $\tilde{\mathcal{U}} \leftarrow \{(\mathbf{x}_i, \hat{y}_i) \mid \mathbf{x}_i \in \tilde{S} \setminus \mathcal{L}\}$
  Update pseudo-labels $\tilde{S} \leftarrow \mathcal{L} \cup \tilde{\mathcal{U}}$
  $b \leftarrow b + 1$
end while
return $H_{\text{final}}(\mathbf{x})$

3.4 Experimental settings

3.4.1 Baseline and benchmark methods

We compare BSSL++ with state-of-the-art boosting algorithms ASSEMBLE (ASMB), SemiBoost (SEMIB), and RegBoost (REGB). In addition, we collect the results of AdaBoost (ADAB) to obtain the baseline performance. That is to say, AdaBoost is trained using only labeled data. This provides insights about improvement in performance, if any, gained by utilizing unlabeled data via the SSL algorithms. On the other hand, results of ASMB, SEMIB, and REGB are considered benchmark performance. In addition, to compare with state-of-the-art SSL methods, we include the results of FixMatch (Sohn et al., 2020) with FT-Transformer (Feature Tokenizer + Transformer) (Gorishniy et al., 2021) network. The parameters of these models were fixed as suggested in (Sohn et al., 2020; Gorishniy et al., 2021), and for data augmentation, we used Gaussian noise with 0 means and std of 0.1 and 0.4 for “weak" and “strong" augmentations, respectively.

Algorithm 4 AdaBoost-like classifier (ENSEMBLE)

Input:
a labeled/pseudo-labeled set $\tilde{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{s}$ of size s
a base learning algorithm Ψ
a number of classes k
a ensemble counter b
Parameter:
a number of weak learners T
a tuning parameter γ_l (default 0.9)
Output:
an ensemble classifier $H_{t,b}$

1: Initialize sample weights $w_{1,b}(i)$ for $i = 1, \ldots, s$ using (3.11)
2: **for** $t \leftarrow 1$ **to** T **do**
3: Normalize $w_{t,b}$ to be a distribution
4: Use $w_{t,b}$ to randomly draw training subset of size $m/2$ from $\tilde{S}$, denoted $\tilde{S}^{tr}$
5: Use Ψ to train a classifier $h_{t,b}$ on $\tilde{S}^{tr}$
6: Compute the error $\epsilon_{t,b}$ of $h_{t,b}$ on $\tilde{S}$ using (3.1)
7: **if** $\epsilon_{t,b} > (k-1)/k$ **then**
8: Discard $h_{t,b}$, go to Step 20
9: **else**
10: Compute the normalized error as $\beta_{t,b} = \epsilon_{t,b}/(1-\epsilon_{t,b})$
11: **end if**
12: Construct majority vote classifier $H_{t,b}(\mathbf{x})$ using (3.2)
13: Compute the error $E_{t,b}$ of $H_{t,b}$ on $\tilde{S}$ using (3.3)
14: **if** $E_{t,b} > (k-1)/k$ **then**
15: Discard $H_{t,b}$, go to Step 20
16: **else**
17: Compute normalized error of $\mathcal{B}_{t,b} = E_{t,b}/(1-E_{t,b})$
18: **end if**
19: Update sample weights $w_{t+1,b}(i)$ using (3.4)
20: **end for**
21: **return** $H_{t,b}$

3.4.2 Datasets

To perform experiments, we collected nine benchmark datasets from OpenML and UCI dataset repositories. Table 3.1 shows the summary of datasets.

3.4.3 Supervised Learning Algorithms

Similar to Learn++ (Polikar et al., 2001), in BSSL++, we employ a one-hidden-layer perceptron (MLP) (Ivakhnenko, 1971) for the base learner. The number of neurons in the hidden layer is fixed to be $2p$ where p is the number of features in each dataset. The rationale behind selecting $2p$ comes from the universal approximation theorem, particularly, according to (Swingler, 1996), the size of the hidden-layer in a one-hidden-layer perceptron does not generally need to be larger than twice the feature size. The remaining hyperparameters of MLP were left at their default values provided by

Algorithm 5 Pseudo-label Nearest Samples (PLNS)

Input:
a labeled/pseudo-labeled set $\tilde{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$ of size m
an unlabeled set $\mathcal{U} = \{\mathbf{x}_i\}_{i=1}^{u}$ of size u
a k-means++ clustering algorithm Ξ^{++}
a distance matrix $\mathbf{A}$
a counter b
a number of classes k
Return:
set of selected unlabeled data points with and without "initial" pseudo-labels $\tilde{\mathcal{U}}_{NN}$ and $\mathcal{U}_{NN}$

if $b = 1$ **then**
  Initialize centroids, $o_1, \ldots, o_k$, by class-specific sample means in $\tilde{S}$
else
  Apply Ξ^{++} to $\tilde{S}$ to generate clusters $\xi_1, \ldots, \xi_k$ with centroids $o_1, \ldots, o_k$
  Compute l_c for $c = 1, \ldots, k$ using (3.7)
  Compute p_{cz} for $c, z = 1, \ldots, k$ using (3.8)
  Assign class label $\hat{y}_z$ to ξ_z based on weighted majority of labels of true labeled data points in ξ_z using $\hat{y}_z = \underset{c}{\operatorname{argmax}} \frac{p_{cz}}{l_c}, z = 1, \ldots, k$
end if
Let $\mathbf{x}^{(1)}(\mathbf{x}_i)$ denote the nearest neighbor of $\mathbf{x}_i \in \tilde{S}$ among $\mathbf{x}_j \in U$ according to $\mathbf{A}$
Take $\mathbf{x}^{(1)}(\mathbf{x}_i), i = 1, \ldots, m$ (i.e. take the nearest unlabeled data points)
$\mathcal{U}_{NN} = \cup_{i=1}^{m} \{\mathbf{x}^{(1)}(\mathbf{x}_i)\}$
$\tilde{y}_j = \underset{\hat{y}_z}{\operatorname{argmin}} ||\mathbf{x}_j - o_z||^2$ for $\mathbf{x}_j \in \mathcal{U}_{NN}$
Construct $\tilde{\mathcal{U}}_{NN} = \{(\mathbf{x}_j, \tilde{y}_j) \mid \mathbf{x}_j \in \mathcal{U}_{NN}\}$
return $\tilde{\mathcal{U}}_{NN}$ and $\mathcal{U}_{NN}$

Table 3.1: Summary of binary classification datasets.

Dataset	No. of features	No of. samples
Australian credit approval (`aus`)	14	690
Haberman's survival (`hms`)	3	306
Heard disease Cleveland (`hdc`)	13	303
Hepatitis (`hep`)	19	155
Pima Indians diabetes (`pid`)	8	768
Kr-vs-Kp (`kvk`)	36	3,196
Voting records (`vote`)	16	435
Mammographic mass (`mm`)	5	961
Breast cancer Wisconsin (`wdbc`)	30	569

`scikit-learn` library (version 1.3) (Pedregosa et al., 2011). Similar to (Bennett et al., 2002), the tuning parameter γ_l in (3.11) was set to 0.9.

3.4.4 Evaluation

In order to simulate scenarios with a large number of unlabeled data compared to labeled data, we first randomly partitioned each dataset into training and test sets with

Table 3.2: The average classification results in terms of test-set accuracy in the form of mean±std ($u : l = 90 : 10$).

		aus	hms	hdc	hep	kvk	mm	pid	vote	wdbc	avg.
NB	ADAB	77.2±4.1	63.6±4.6	74.8±6.5	80.0±6.8	82.4±1.4	80.3±3.8	69.7±3.2	90.6±4.6	93.9±2.3	79.2±4.1
	ASMB	76.5±4.6	66.7±7.8	75.5±5.4	80.0±5.2	81.1±2.4	79.5±3.8	70.3±5.8	89.0±4.6	93.2±2.1	79.1±4.6
	SEMIB	78.4±3.3	69.2±5.7	69.5±7.1	81.0±4.9	80.4±2.8	80.1±3.7	68.6±2.9	88.6±3.9	93.3±3.1	78.8±4.2
	REGB	80.4±4.5	70.0±4.4	80.2±3.9	80.7±5.3	82.6±1.6	80.5±3.2	**73.5±2.4**	91.0±3.9	94.2±1.8	81.5±3.4
C4.5	ADAB	81.2±2.7	62.0±6.3	76.5±6.1	77.4±11.8	84.1±1.4	75.8±2.9	71.3±3.2	93.1±2.4	91.9±2.2	79.3±4.3
	ASMB	77.3±5.8	65.4±6.9	70.5±5.9	71.6±14.1	81.6±2.2	79.0±4.9	68.9±5.0	91.5±3.1	88.7±5.7	77.2±6.0
	SEMIB	77.7±6.3	69.8±4.7	63.7±7.2	79.6±4.6	81.9±2.5	78.9±2.5	69.4±4.3	91.9±2.2	91.4±2.4	78.3±4.1
	REGB	**83.0±3.5**	68.8±5.2	77.3±7.3	80.0±5.8	84.1±1.4	80.3±3.6	72.3±3.6	**93.7±2.3**	93.0±1.3	81.4±3.8
MLP	ADAB	77.7±3.9	71.0±6.5	75.2±4.9	65.8±29.9	87.2±15.0	**82.1±2.3**	72.9±3.0	91.0±4.3	**95.4±2.5**	79.8±8.0
	ASMB	78.9±3.7	72.4±5.5	76.7±4.7	79.7±8.0	89.1±2.1	81.1±3.9	73.3±3.3	91.3±4.2	93.8±3.1	81.8±4.3
	SEMIB	68.5±2.8	**74.2±0.0**	66.2±4.9	**82.6±3.6**	60.0±3.1	80.5±3.9	65.0±0.0	90.3±2.1	70.8±3.4	73.1±2.6
	REGB	81.2±3.2	**74.2±0.0**	**80.3±5.2**	**82.6±5.6**	82.2±4.0	79.9±1.9	67.9±2.6	89.8±2.4	86.7±5.7	80.5±3.4
	BSSL++	82.2±2.7	**74.2±0.0**	79.5±4.2	81.0±7.4	**90.4±1.4**	81.5±3.7	73.2±3.4	91.3±2.7	**95.4±1.7**	**83.2±3.0**
FT-T	FixMatch	82.8±3.6	70.5±4.9	78.7±6.7	79.4±9.3	60.9±15.6	81.1±2.4	70.5±3.9	93.4±1.8	88.4±12.9	78.4±6.8

Table 3.3: The average classification results in terms of test-set accuracy in the form of mean±std ($u : l = 80 : 20$).

		aus	hms	hdc	hep	kvk	mm	pid	vote	wdbc	avg.
NB	ADAB	79.6±3.2	70.5±7.8	76.7±5.9	80.6±4.8	83.9±1.2	80.5±3.5	72.6±2.2	93.6±1.6	95.0±2.9	81.4±3.7
	ASMB	76.6±6.0	72.1±5.2	79.3±3.3	80.3±8.4	84.1±1.4	79.8±3.3	74.0±4.1	91.3±2.6	94.2±2.7	81.3±4.1
	SEMIB	78.7±2.6	70.7±4.4	74.0±7.3	81.6±6.6	84.0±1.2	80.3±3.6	72.2±3.0	89.9±2.8	94.3±2.1	80.6±3.7
	REGB	81.7±3.0	72.6±6.4	80.5±3.5	81.0±8.7	84.2±1.4	80.6±3.5	74.2±4.4	92.6±2.0	94.4±2.9	82.4±4.0
C4.5	ADAB	83.1±3.7	64.4±3.8	77.7±4.0	79.3±10.1	87.2±0.9	76.5±3.5	72.5±4.0	93.8±1.1	95.2±1.9	81.1±3.7
	ASMB	78.3±4.3	67.7±8.3	75.5±7.3	73.9±7.3	84.5±1.0	79.4±3.1	71.5±3.1	91.8±4.0	91.1±1.7	79.3±4.5
	SEMIB	82.2±5.1	70.2±4.6	71.1±6.7	80.0±5.9	85.2±1.2	80.7±1.3	72.5±3.9	93.4±2.7	92.1±2.9	80.8±3.8
	REGB	**84.5±3.4**	69.0±8.0	79.0±6.9	80.3±6.9	87.8±0.8	80.5±3.2	72.9±4.1	93.9±2.2	95.3±1.8	82.6±4.1
MLP	ADAB	80.7±3.7	**74.4±5.9**	73.6±7.8	81.0±5.5	83.2±21.3	82.4±1.8	74.6±3.7	93.1±2.7	**96.6±1.6**	82.2±6.0
	ASMB	81.4±3.5	74.0±3.8	80.7±3.1	81.3±5.5	93.7±1.3	81.1±3.4	**76.1±2.9**	92.5±4.0	95.3±1.3	84.0±3.2
	SEMIB	67.0±4.6	74.2±0.0	73.9±4.8	82.3±3.0	61.6±2.1	80.6±3.3	65.0±0.7	89.4±3.6	73.6±3.7	74.2±2.9
	REGB	82.2±3.7	74.2±0.0	78.7±4.1	82.9±5.6	81.7±4.0	80.6±2.4	70.5±2.1	90.7±2.7	89.9±3.9	81.3±3.2
	BSSL++	83.8±2.5	74.3±0.8	**82.3±5.5**	**84.4±5.7**	**94.3±1.4**	**82.7±3.8**	**76.1±1.7**	93.1±3.0	95.8±1.9	**85.2±2.9**
FT-T	FixMatch	79.9±8.2	71.0±5.2	78.2±9.1	80.3±6.5	52.2±0.0	81.7±3.2	73.7±3.7	**94.3±2.9**	92.5±9.9	78.2±5.4

an 80:20 ratio. Then, we further split the training set into unlabeled and labeled sets with relative size $u : l = 90 : 10$ and $u : l = 80 : 20$ ratios. All trained classifiers are evaluated on the test set. Additionally, all splits were performed using stratification. We repeat the experiments 20 times to obtain the average classification performance using different labeled, unlabeled, and test sets.

3.5 Results

Tables 3.2 and 3.3 show the average classification performance of BSSL++ obtained on benchmark datasets for the two cases of $u : l = 90 : 10$ and $80 : 20$, respectively. The results are shown in the form of average test-set accuracies ± standard deviations. Tables 3.2 and 3.3 also demonstrate the results of four boosting algorithms: ASMB,

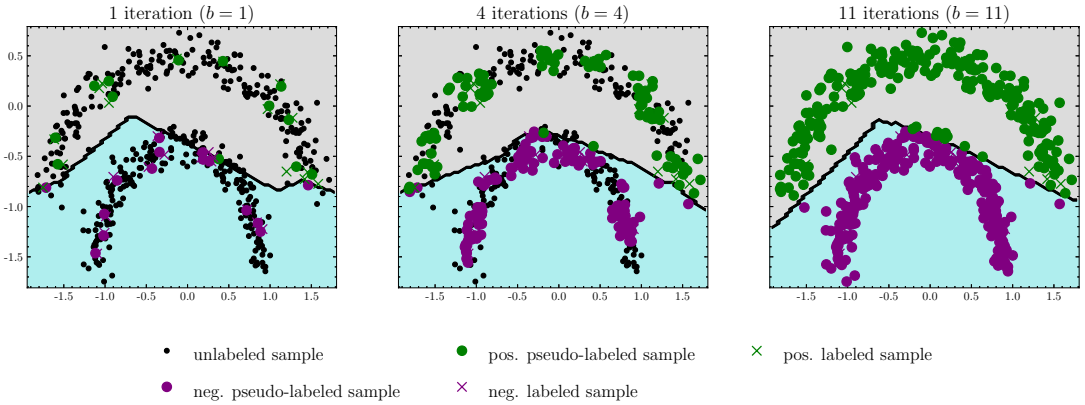

Figure 3.1: Decision boundaries of two half-moons dataset over iterations b produced by BSSL++.

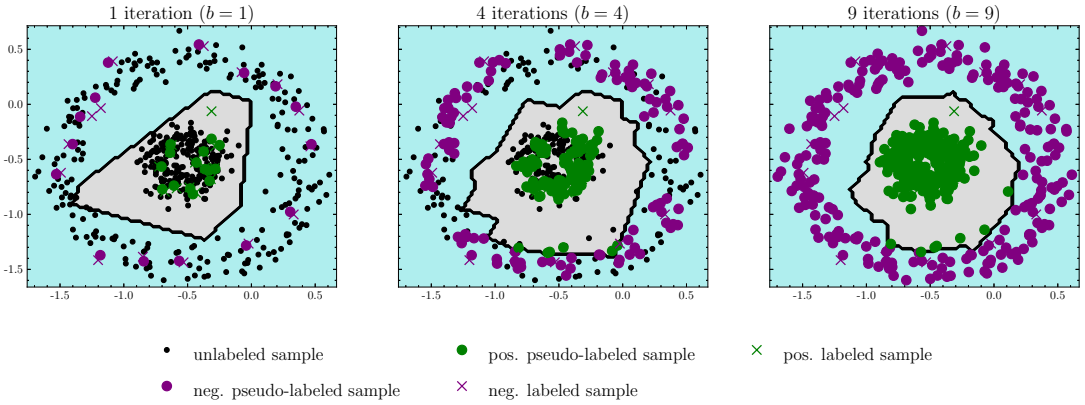

Figure 3.2: Decision boundaries of two circles dataset over iterations b produced by BSSL++.

SEMIB, REGB, and ADAB. Similar to (Lu et al., 2021; Triguero et al., 2015; Chen and Wang, 2011), Naive Bayes (NB), C4.5, and MLP are used as the base learning algorithms in these four boosting methods. As a result, we have 117 performance comparisons for each algorithm (13 algorithms $\times$ 9 datasets). The results show that in $\sim$84% and $\sim$91% of the comparisons, BSSL++ outperformed all other algorithms for $u : l = 90 : 10$ and $80 : 20$, respectively. At the same time, the runner-ups, REGB with NB and ASMB with MLP, showed better performance than other algorithms in $\sim$69% and $\sim$74% of the comparisons for $u : l = 90 : 10$ and $80 : 20$, respectively.

To summarize the results presented in Tables 3.2 and 3.3, on average (across all datasets and the 20 repetitions), BSSL++ achieved accuracy of 83.2$\pm$3.0% and 85.2$\pm$2.9% for $u : l = 90 : 10$ and $80 : 20$, respectively. This represents a considerable improvement compared to the runner-up (ASMB) with an average accuracy of 81.8$\pm$4.3% and 84.0$\pm$3.2% for $u : l = 90 : 10$ and $80 : 20$, respectively. As expected, the average performance of all algorithms improves with a larger size of labeled data (i.e., for $u : l = 80 : 20$ compared to $90 : 10$).

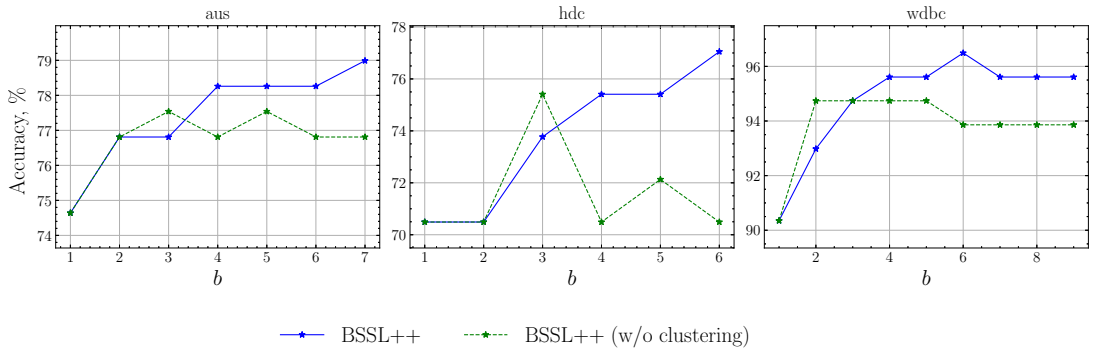

Figure 3.3: The test-set accuracy results over iterations b for `aus`, `hdc`, and `wdbc` datasets produced by BSSL++ and BSSL++ without clustering ($u : l = 90 : 10$).

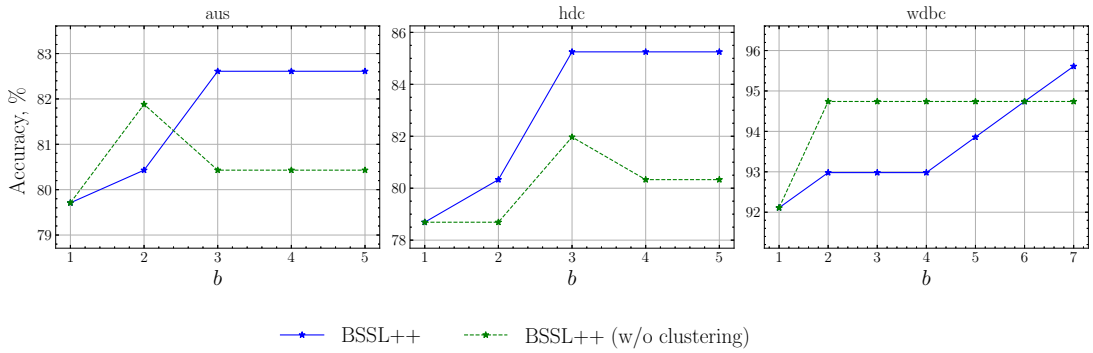

Figure 3.4: The test-set accuracy results over iterations b for `aus`, `hdc`, and `wdbc` datasets produced by BSSL++ and BSSL++ without clustering ($u : l = 80 : 20$).

3.6 Discussion

The behavior of BSSL++. To examine the behavior of BSSL++, we conducted additional experiments using two synthetic datasets known as *two half-moons* and *two circles*. The nonlinear ground-truth decision boundaries of these datasets are relatively difficult to handle using SSL algorithms. Particularly, the shape of these datasets corresponds to the shape of a moon and a circle. That being said, these datasets are commonly used to examine developed SSL algorithms (Chen and Wang, 2011; Mallapragada et al., 2009; Ren et al., 2020).

The experimental setup was as follows. Each dataset was generated with 500 samples where 470 of them are unlabeled and 30 of them are labeled samples (15 per class). Figures 3.1 and 3.2 depict the decision boundaries produced by BSSL++ at different boosting iterations. In each plot, a positive labeled, a negative labeled, a positive pseudo-labeled, a negative pseudo-labeled, and an unlabeled data point are represented with a green cross, a purple cross, a green full circle, a purple full circle, and a black full circle respectively. In particular, full-circled green and purple samples are those that are selected at each iteration and used in the training process. As seen in Figures 3.1 and 3.2, the decision boundaries correctly adjust to ground-truth boundaries after a few iterations.

Importance of the clustering-based procedure. As discussed in Section 3.3, the generation of "initial" pseudo-labels for a selected set of unlabeled data is essentially implemented using a clustering algorithm (Algorithm 5). This procedure is encoded into BSSL++ in order to guard against noise accumulation. In order to examine the importance of this clustering-based procedure, we conduct an ablation study by turning off the clustering after the first iteration and using the ensemble classifier constructed so far (i.e., H_{final}) to produce "initial" pseudo-labels in subsequent iterations. Therefore, we entirely rely on H_{final} for the generation of pseudo-labels. Figures 3.3 and 3.4 demonstrate the test-set accuracy results as a function of learning iterations obtained by BSSL++ with and without clustering for both cases of $u : l$. In particular, the results are obtained for `aus`, `hdc`, and `wdbc` datasets in one random split. In each plot, the solid and dashed curves represent BSSL++ with and without clustering, respectively. The results show that:

- the accumulation of noise over the iterations degrades the performance of BSSL++ without clustering. For example, in Figures 3.3 and 3.4, the performance of BSSL++ without clustering on `hdc` and `aus` datasets fluctuates over the iterations before declining to about the same level as in the first iteration, respectively;

- the decision boundary of BSSL++ without clustering may not improve over iterations. For example, for `wdbc` dataset in Figure 3.4, the test-set accuracy results of BSSL++ without clustering remained at the same level after the second iteration; and

- the performance of BSSL++ (with clustering) gradually improves over iterations. For example, in Figure 3.3, the test-set accuracy increase on `hdc` dataset starts from $\sim$70.4% and reaches $\sim$77%.

In summary, BSSL++ improves the generalization performance by learning from newly selected unlabeled data points at each iteration (this was also observed from results of BSSL++ and ADAB in Tables 3.2 and 3.3). Moreover, the generation of "initial" pseudo-labels using the clustering-based procedure can mitigate noise accumulation.

Pairwise distance. Recall that SemiBoost and RegBoost algorithms use pairwise affinity measures to direct the selection and utility of pseudo-labeled data. However, as the computation of affinity measures is based on the radial basis function, the scaling parameter of the function largely impacts the performance of the classifier and thus requires tuning (Zhu et al., 2003). In contrast, as discussed in Section 3.3, we use a pairwise distance metric (Manhattan distance), which does not require tuning, to find the nearest unlabeled data point to each labeled one (line 10, Algorithm 5).

The importance of unlabeled and labeled data points. As mentioned in Section 3.3, we compute the sample weights at the first iteration using (3.11), in which the parameter γ_l controls the initial weight distribution (also see line 1, Algorithm 4). As discussed in Section 3.4, in all experiments γ_l is set to be 0.9 similar to (Bennett et al., 2002). This increases the chance of labeled data being sampled in the first iteration. That is to say, the training set would primarily consist of labeled data points at the first iteration. Hence, the first base classifier would likely be trained using the set of data points that are mostly from labeled data. Nevertheless, another implication of (3.11) is attributed to the uniform distribution of weights for all unlabeled (and labeled) data; that is, all unlabeled data points have the same chance of being sampled. However, as pointed out in (Ren et al., 2020), not all unlabeled data are equal in SSL. Future research may focus on studying various techniques for the initialization of sample weights.

Time complexity. The time complexity of BSSL++ in terms of big $\mathcal{O}$ notation can be derived from the time complexities of the following parts: (i) for T boosting iterations, training one-hidden-layer perceptron requires $\mathcal{O}(ksTp^2e)$ (LeCun et al., 2012), where k is the number of classes, $s = u + l$ is the total number of unlabeled and labeled data points, p is the number of features, and e is the number of epochs; (ii) running the algorithm PLNS (Algorithm 5) is dominated by training time of k-means++ algorithm, which requires $\mathcal{O}(kspr)$ (Bachem et al., 2016), where r is the maximum number of iterations to solve k-means problem using Lloyd's algorithm (Lloyd, 1982); (iii) updating pseudo-labels after boosting (line 10, Algorithm 3) requires $\mathcal{O}(kTup^2 \log u)$. These procedures are repeated $\mathcal{O}(\log u)$ times, that is, until all unlabeled data are selected. As a result, the time complexity of BSSL++ is $\mathcal{O}(kps \log u(r + pTe + Tp \log u))$. This indicates that the proposed algorithm is linearly complex in terms of s, T, e, k, r, logarithmically complex in terms of u, and quadratically complex in terms of p. In comparison, the time complexity of RegBoost algorithm with Naive Bayes as a base classifier is $\mathcal{O}(ks \log s + kvTsp)$, where v is the number of nearest neighbors used in RegBoost.

Incremental learning. Despite being inspired by Learn++, the BSSL++ algorithm is not an "incremental" learning algorithm in its current form. As pointed out in (Polikar et al., 2001), the core mechanism of an incremental learning strategy corresponds to not requiring access to training data that was used before once a new batch of data arrived. In contrast, BSSL++ uses all labeled and past unlabeled data (via Algorithm 5) when a new portion of selected unlabeled data is provided to the algorithm. A promising research direction is adapting BSSL++ to learn from a newly arrived portion of unlabeled data in an incremental manner.

3.7 Summary

In this chapter, inspired by Learn++, we proposed a boosting algorithm for semi-supervised learning (BSSL++). In particular, BSSL++ is designed using a classification mechanism similar to Learn++, which is an (AdaBoost-like) ensemble of ensemble classifiers. During the learning of an ensemble classifier at each iteration the training set comprises labeled and selected pseudo-labeled unlabeled data. In order to guard against noise accumulation, the generation of pseudo-labels does not depend on previously pseudo-labeled data and the classifier constructed so fart. The guidance of selection and utility of unlabeled data was based on semi-supervised smoothness and cluster assumptions. The experimental results demonstrated that BSSL++ can outperform state-of-the-art semi-supervised boosting algorithms.

Chapter 4

BSSL++ with Similarity and Dissimilarity-based Manifold Regularized Adaptive Boosting Algorithm

In BSSL++ presented in Chapter 3, we did not take into account the third main SSL assumption, which is the manifold assumption. Hence, in this chapter, we aim to encode this assumption into BSSL++. In order to achieve this, we first propose a manifold regularized boosting algorithm that can be employed by BSSL++. As a result, this chapter covers a novel supervised boosting algorithm with manifold regularization and applicability of this algorithm with BSSL++.

4.1 Introduction

Previous works have shown that the use of knowledge about the underlying geometric structure of data can possibly help to improve the performance of the classifier (Zhu et al., 2018; Tang et al., 2019). In this regard, one effective approach for capturing the intrinsic structure of data relies on *manifold assumption*. Recall that manifold assumption states that data may lie in a low-dimensional manifold within higher-dimensional representation space (Chapelle et al., 2006; Bengio et al., 2013).

Across different manifold learning techniques, the graph Laplacian has been employed to learn a low-dimensional embedding that retains the local neighborhood connections in data (Watanabe et al., 2023; Maretic and Frossard, 2020; Trillos et al., 2021; Belkin et al., 2006). Particularly, the graph Laplacian uses the similarity information between data points to promote a smooth variation of model predictions over the data manifold. At the same time, to encode dissimilarity information, the analog of graph Laplacian, the mixed graph Laplacian, was introduced in (Goldberg

et al., 2007). In this regard, to encode the manifold assumption into BSSL++, we aim to replace the boosting ensemble classifier (AdaBoost-like) algorithm used in BSSL++ with a manifold regularized boosting algorithm.

AdaBoost, which is one of the powerful boosting techniques, builds an ensemble of base (weak) learners using a series of iteratively weighted training data (Freund and Schapire, 1997). In particular, at each iteration, the observation-specific weights are computed based on the performance of the ensemble classifier trained at the previous iteration. Specifically, the weights of misclassified data points are increased so that in the next iteration the learner focuses more on the incorrectly predicted data points in the previous iteration.

Given the success of regularization methods in many applications, soft margin AdaBoost, L_1-regularized AdaBoost, and entropy-based regularized AdaBoost algorithms have been developed as the extension of AdaBoost in (Rätsch et al., 2001), (Xi et al., 2009), and (Bereta, 2017), respectively. That being said, early work that used the graph Laplacian in AdaBoost corresponds to (Wang and Kégl, 2004). In particular, Wang and Kégl (Wang and Kégl, 2004) proposed boosting on manifolds algorithm, where each base learner is regularized using graph Laplacian.

Having said that, we aim to extend AdaBoost using manifold regularization based on label similarity and dissimilarity information between data points. To achieve this goal, we (i) propose a convex objective function for training AdaBoost while considering the underlying geometric structure of data, and (ii) use the mixed-graph analog of the graph Laplacian to regularize the smoothness on the data manifold.

4.2 Theory: graph Laplacian and mixed-graph Laplacian

A general framework of manifold regularization based on graph Laplacian has been formulated by Belkin *et al.* (Belkin et al., 2006). Let us consider a dataset $\{\mathbf{x}_i\}_{i=1}^n$ of size n, where $\mathbf{x}_i \in \mathbb{R}^p$ is a p-dimensional feature vector. Given the a symmetric weighted adjacency matrix $\mathbf{W}$, where $\mathbf{W}_{ij}$ is the similarity measure between $\mathbf{x}_i$ and $\mathbf{x}_j$ data points, and a discriminant function $f : \mathbb{R}^p \to \mathbb{R}$, the manifold regularization term is defined as (Belkin et al., 2006)

$$\frac{1}{2}\sum_{i}^{n}\sum_{j}^{n}\mathbf{W}_{ij}\Big(f(\mathbf{x}_i) - f(\mathbf{x}_j)\Big)^2 = \mathbf{f}^T\mathbf{L}\mathbf{f}, \tag{4.1}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the graph Laplacian, $\mathbf{D}$ denotes the degree matrix, which is a diagonal matrix with $\mathbf{D}_{ii} = \sum_j^n \mathbf{W}_{ij}$, and $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^T$. Given the highly similar $\mathbf{x}_i$ and $\mathbf{x}_j$ data points, minimization of (4.1) in terms of f enforces $f(\mathbf{x}_i) \approx f(\mathbf{x}_j)$. By leveraging this regularization, Laplacian Support Vector Machine

(LapSVM) and Laplacian Regularized Least Squares (LapRLS) algorithms have been proposed in (Belkin et al., 2006).

As pointed out in (Goldberg et al., 2007), to force $f(\mathbf{x}_i) \approx -f(\mathbf{x}_j)$ when $\mathbf{x}_i$ and $\mathbf{x}_j$ have different labels, $f(\mathbf{x}_i) - f(\mathbf{x}_j)$ in (4.1) can be replaced with $f(\mathbf{x}_i) + f(\mathbf{x}_j)$. In this regard, the following definition is instrumental in formulating the mixed-graph Laplacian-based manifold regularization term that encodes both label similarity and dissimilarity.

Definition 4.2.1. (Goldberg et al., 2007) A mixed undirected graph with n nodes is described by two matrices $\mathbf{S}$ and $\mathbf{W}$ with sizes $n \times n$. $\mathbf{S}$ represents type of the edge, in which $\mathbf{S}_{ij} = -1$ if i, j nodes have a dissimilarity edge, and $\mathbf{S}_{ij} = 1$, if i, j nodes have a similarity edge, and $\mathbf{W}$ describes the edge weights regardless of their types. □

The mixed-graph Laplacian-based manifold regularization term is then defined as (Goldberg et al., 2007)

$$\frac{1}{2}\sum_{i}^{n}\sum_{j}^{n}\mathbf{W}_{ij}\Big(f(\mathbf{x}_i) - \mathbf{S}_{ij}f(\mathbf{x}_j)\Big)^2 = \mathbf{f}^T\mathbf{M}\mathbf{f}, \tag{4.2}$$

where $\mathbf{W}_{ij}$ represents the strength of similarity or dissimilarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, $\mathbf{S}_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are from the same class (label similarity) and $\mathbf{S}_{ij} = -1$ otherwise (label dissimilarity), $\mathbf{M} = \mathbf{L} + (\mathbf{1} - \mathbf{S}) \odot \mathbf{W}$ is the mixed-graph Laplacian, $\mathbf{1}$ is all-one matrix, and $\odot$ is the element-wise product. The regularization term (4.2) considers both the label similarity and dissimilarity, compared to (4.1), which only uses label similarity knowledge between data points.

Following this intuition, in this chapter, we introduce a method that allows training AdaBoost while forcing the ensemble of t base learners $H_t(\mathbf{x})$ to produce similar outputs for observations that belong to the same class and also encouraging $H_t(\mathbf{x}_i)$ and $H_t(\mathbf{x}_j)$ to have different signs when $\mathbf{x}_i$ and $\mathbf{x}_j$ are from different classes.

4.3 Method: AdaBoost.SDM

In a binary classification problem, let $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ be the labeled training set of size l where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{-1, +1\}$ corresponding class label. In AdaBoost, the discriminant function, denoted $H_t(\mathbf{x}) : \mathbb{R}^p \to \mathbb{R}$, is a linear combination of t base classifiers. In particular, it can be expressed as (Freund and Schapire, 1997)

$$H_t(\mathbf{x}) = H_{t-1}(\mathbf{x}) + \alpha_t h_t(\mathbf{x}), \tag{4.3}$$

where $h_t(\mathbf{x}) : \mathbb{R}^p \to \{-1, +1\}$ is the base classifier trained at iteration t, $\alpha_t \in \mathbb{R}$ is the weight of the t^{th} base classifier. Also note that $H_0(\mathbf{x}) = 0$. Then, AdaBoost classifier is given by

$$\psi(\mathbf{x}) = \begin{cases} +1, & \text{if } H_t(\mathbf{x}) > 0, \\ -1, & \text{otherwise}, \end{cases} \tag{4.4}$$

4.3.1 Regularized cost function

AdaBoost constructs a sequence of base classifiers in a greedy fashion (Friedman et al., 2000). In particular, given $H_{t-1}(\mathbf{x})$, finding $H_t(\mathbf{x})$ is equivalent to finding $h_t(\mathbf{x})$ and α_t. That being said, we aim to optimize the regularized cost of training $H_t(\mathbf{x})$, or total error E, which is defined as

$$E = L + \lambda G, \tag{4.5}$$

where L is the empirical loss between $H_t(\mathbf{x})$ and class labels, G is a regularization term, and λ is a hyperparameter that determines the impact of G. We seek to minimize E by finding $h_t(\mathbf{x})$ and α_t that when combined with $H_{t-1}(\mathbf{x})$ results in the lowest E. In AdaBoost, the common empirical loss is the sum of exponential loss for each data point in the training set, which is given by (Hastie et al., 2009)

$$L = \sum_{i}^{l} e^{-2y_i H_t(\mathbf{x}_i)}. \tag{4.6}$$

Since we intend to use the manifold regularization term based on both label similarity and dissimilarity between data points, using (4.2), G can be expressed as (Goldberg et al., 2007)

$$G = \frac{1}{2}\sum_{i}^{l}\sum_{j}^{l} \mathbf{W}_{ij}\Big(H_t(\mathbf{x}_i) - \mathbf{S}_{ij}H_t(\mathbf{x}_j)\Big)^2 = \mathbf{h}_t^T \mathbf{M}\mathbf{h}_t, \tag{4.7}$$

where $\mathbf{h}_t = [H_t(\mathbf{x}_1), \ldots, H_t(\mathbf{x}_l)]^T$.

Inspired by (Mallapragada et al., 2009), we leverage the following approximation using the power-series expansion of $\cosh(\mathbf{a}) \approx 1 + \frac{\mathbf{a}^2}{2}$ to achieve a closed-form solution for α_t:

$$\begin{aligned} &\underset{\alpha_t, h_t}{\operatorname{argmin}} \sum_{i}^{l}\sum_{j}^{l} \mathbf{W}_{ij}\cosh\Big(H_t(\mathbf{x}_i) - \mathbf{S}_{ij}H_t(\mathbf{x}_j)\Big) \approx \\ &\underset{\alpha_t, h_t}{\operatorname{argmin}} \Big(\sum_{i}^{l}\sum_{j}^{l} \mathbf{W}_{ij} + \frac{1}{2}\sum_{i}^{l}\sum_{j}^{l} \mathbf{W}_{ij}\Big(H_t(\mathbf{x}_i) - \mathbf{S}_{ij}H_t(\mathbf{x}_j)\Big)^2\Big). \end{aligned} \tag{4.8}$$

The first term on the right-hand side of (4.8) is constant, hence, minimizing the left-hand side is (approximately) equivalent to minimizing the right-hand side. As a result, we use the left-hand side of the equation as the regularization term G. Then, E is given by

$$E = \sum_{i}^{l} e^{-2y_i H_t(\mathbf{x}_i)} + \lambda \sum_{i}^{l}\sum_{j}^{l} \mathbf{W}_{ij}\cosh\Big(H_t^{ij}\Big), \tag{4.9}$$

where $H_t^{ij} \triangleq H_t(\mathbf{x}_i) - \mathbf{S}_{ij}H_t(\mathbf{x}_j)$. Using $\cosh(H_t^{ij}) = (e^{H_t^{ij}} + e^{-H_t^{ij}})/2$ in (4.9) yields

$$E = \sum_{i}^{l} e^{-2y_i H_t(\mathbf{x}_i)} + \frac{\lambda}{2}\sum_{i}^{l}\sum_{j}^{l} \mathbf{W}_{ij}e^{H_t^{ij}} + \frac{\lambda}{2}\sum_{i}^{l}\sum_{j}^{l} \mathbf{W}_{ij}e^{-H_t^{ij}}. \tag{4.10}$$

Hence, the problem of finding the best α_t and $h_t(\mathbf{x})$ can be written as

$$\begin{aligned}\underset{\alpha_t,h_t}{\operatorname{argmin}} \sum_i^l & e^{-2y_i H_{t-1}(\mathbf{x}_i)} e^{-2y_i\alpha_t h_t(\mathbf{x}_i)} \\ &+ \frac{\lambda}{2}\sum_i^l\sum_j^l \mathbf{W}_{ij} e^{H_{t-1}^{ij}} e^{\alpha_t(h_t(\mathbf{x}_i)-\mathbf{S}_{ij}h_t(\mathbf{x}_j))} \\ &+ \frac{\lambda}{2}\sum_i^l\sum_j^l \mathbf{W}_{ij} e^{-H_{t-1}^{ij}} e^{\alpha_t(-h_t(\mathbf{x}_i)+\mathbf{S}_{ij}h_t(\mathbf{x}_j))},\end{aligned} \tag{4.11}$$

where $H_t(\mathbf{x})$ in (4.10) is expressed in terms of $H_{t-1}(\mathbf{x})$, α_t, and h_t using (4.3). However, the product of $e^{\pm\alpha_t h_t(\mathbf{x}_i)}$ and $e^{\mp\alpha_t h_t(\mathbf{x}_j)}$ in (4.11) makes the minimization challenging. To simplify the optimization, we first construct an upper bound of E, denoted as $\overline{E}_1$, using the following identities and then minimize the upper bound (Mallapragada et al., 2009):

$$e^{\alpha_t(h_t(\mathbf{x}_i)-\mathbf{S}_{ij}h_t(\mathbf{x}_j))} \leq \frac{1}{2}\left(e^{2\alpha_t h_t(\mathbf{x}_i)} + e^{-2\mathbf{S}_{ij}\alpha_t h_t(\mathbf{x}_j)}\right), \tag{4.12}$$

$$e^{\alpha_m(-h_t(\mathbf{x}_i)+\mathbf{S}_{ij}h_t(\mathbf{x}_j))} \leq \frac{1}{2}\left(e^{-2\alpha_t h_t(\mathbf{x}_i)} + e^{2\mathbf{S}_{ij}\alpha_t h_t(\mathbf{x}_j)}\right). \tag{4.13}$$

Using (4.12) and (4.13) yields

$$E \leq \overline{E}_1\,, \tag{4.14}$$

where

$$\begin{aligned}\overline{E}_1 = \sum_i^l & e^{-2y_i H_{t-1}(\mathbf{x}_i)} e^{-2y_i\alpha_t h_t(\mathbf{x}_i)} \\ &+ \frac{\lambda}{4}\sum_i^l\sum_j^l \mathbf{W}_{ij} e^{H_{t-1}^{ij}} e^{2\alpha_t h_t(\mathbf{x}_i)} \\ &+ \frac{\lambda}{4}\sum_i^l\sum_j^l \mathbf{W}_{ij} e^{H_{t-1}^{ij}} e^{-2\mathbf{S}_{ij}\alpha_t h_t(\mathbf{x}_j)} \\ &+ \frac{\lambda}{4}\sum_i^l\sum_j^l \mathbf{W}_{ij} e^{-H_{t-1}^{ij}} e^{-2\alpha_t h_t(\mathbf{x}_i)} \\ &+ \frac{\lambda}{4}\sum_i^l\sum_j^l \mathbf{W}_{ij} e^{-H_{t-1}^{ij}} e^{2\mathbf{S}_{ij}\alpha_t h_t(\mathbf{x}_j)}.\end{aligned} \tag{4.15}$$

We can rewrite $\overline{E}_1$ by encoding label similarity or dissimilarity by setting $\mathbf{S}_{ij} = 1$ for $y_i = y_j$, and $\mathbf{S}_{ij} = -1$ for $y_i = -y_j$ (details can be found in Appendix E):

$$\overline{E}_1 = \sum_i^l e^{-2\alpha_t h_t(\mathbf{x}_i)} r_i^t + e^{2\alpha_t h_t(\mathbf{x}_i)} q_i^t\,, \tag{4.16}$$

where

$$\begin{aligned}r_i^t = & e^{-2H_{t-1}(\mathbf{x}_i)}\delta(y_i,+1) \\ &+ \frac{\lambda}{2}\sum_j^l \mathbf{W}_{ij} e^{-H_{t-1}(\mathbf{x}_i)+H_{t-1}(\mathbf{x}_j)}\delta(y_i,y_j) \\ &+ \frac{\lambda}{2}\sum_j^l \mathbf{W}_{ij} e^{-H_{t-1}(\mathbf{x}_i)-H_{t-1}(\mathbf{x}_j)}\delta(y_i,-y_j),\end{aligned} \tag{4.17}$$

$$
\begin{aligned}
q_i^t =& e^{2H_{t-1}(\mathbf{x}_i)}\delta(y_i,-1)+ \\
&+\frac{\lambda}{2}\sum_j^l \mathbf{W}_{ij}e^{H_{t-1}(\mathbf{x}_i)-H_{t-1}(\mathbf{x}_j)}\delta(y_i,y_j) \\
&+\frac{\lambda}{2}\sum_j^l \mathbf{W}_{ij}e^{H_{t-1}(\mathbf{x}_i)+H_{t-1}(\mathbf{x}_j)}\delta(y_i,-y_j),
\end{aligned}
\tag{4.18}
$$

where $\delta(a,b)=1$ when $a=b$ and 0 otherwise.

4.3.2 Sample weights w^t

The weights associated with training data at iteration t are given by

$$
w_i^t = r_i^t\delta(y_i,+1)+q_i^t\delta(y_i,-1), \quad i=1,\ldots,l. \tag{4.19}
$$

We note that when $\lambda=0$, the sample weights work similarly to the original AdaBoost. In particular, the weights of correctly classified observations are decreased and the weights of misclassified ones are increased (see Appendix F).

4.3.3 Selection of h_t

Since $h_t(\mathbf{x}_i)$ is a function from $\mathbb{R}^p$ to $\{-1,+1\}$, we can divide $\overline{E}_1$ into two summations as

$$
\begin{aligned}
\overline{E}_1 =& \sum_i^l (e^{-2\alpha_t}r_i^t + e^{2\alpha_t}q_i^t)\delta(h_t(\mathbf{x}_i),+1) \\
&+\sum_i^l (e^{2\alpha_t}r_i^t + e^{-2\alpha_t}q_i^t)\delta(h_t(\mathbf{x}_i),-1).
\end{aligned}
\tag{4.20}
$$

Reorganizing (4.20) yields

$$
\overline{E}_1 = e^{-2\alpha_t}W_c + e^{2\alpha_t}W_e, \tag{4.21}
$$

where

$$
W_c = \sum_i^l \Big(r_i^t\delta(h_t(\mathbf{x}_i),+1)+q_i^t\delta(h_t(\mathbf{x}_i),-1)\Big), \tag{4.22}
$$

$$
W_e = \sum_i^l \Big(r_i^t\delta(h_t(\mathbf{x}_i),-1)+q_i^t\delta(h_t(\mathbf{x}_i),+1)\Big). \tag{4.23}
$$

Here, W_c and W_e describe the weighted cost of correctly and incorrectly classified data points, respectively, with some additional terms that represent manifold regularization.

In order to find the optimal h_t, we multiply both sides of (4.21) by $e^{2\alpha_t}$. In particular,

$$
e^{2\alpha_t}\overline{E}_1 = (W_c+W_e)+(e^{4\alpha_t}-1)W_e. \tag{4.24}
$$

At the current iteration, the first term on the right-hand side of (4.24) is constant. Hence, we select h_t that will minimize W_e.

4.3.4 Computation of α_t

To find the α_t that minimizes $\overline{E}_1$ with fixed h_t, we differentiate the $\overline{E}_1$ w.r.t. α_t and equate to zero,

$$-2e^{-2\alpha_t}W_c + 2e^{2\alpha_t}W_e = 0\,. \tag{4.25}$$

Then the optimal α_t is given by

$$\alpha_t = \frac{1}{4}\ln\Big(\frac{W_c}{W_e}\Big)\,. \tag{4.26}$$

Algorithm 6 outlines the step-by-step procedure of AdaBoost.SDM.

Algorithm 6 AdaBoost.SDM algorithm

Input:
a labeled set $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_i^l$ of size l
a base learning algorithm Ψ
Parameter:
a number of iterations T
a tuning parameter λ
a tuning parameter σ
Output:
an ensemble classifier ψ

1: Compute weighted adjacency matrix $\mathbf{W}$ where $\mathbf{W}_{ij} = e^{-||\mathbf{x}_i - \mathbf{x}_j||_2^2/2\sigma^2}$ for each $i, j \in l$
2: Initialize $C_0 = 0$
3: **for** $t = 1$ in T **do**
4: Compute r_i^t for each $i \in N$ using (4.17)
5: Compute q_i^t for each $i \in N$ using (4.18)
6: Compute weights w_i^t for each $i \in l$ using (4.19)
7: Normalize the weights to math a distribution
8: Call Ψ on $\mathcal{L}$ with provided w_i^t for each $i \in l$ to train a base classifier h_t
9: Obtain predictions on training set $\hat{y}_i \leftarrow h_t(\mathbf{x}_i)$ for each $i \in l$
10: Compute W_c and W_e using (4.22) and (4.23), respectively
11: Compute α_t using (4.26)
12: **if** $\alpha_t \leq 0$ **then**
13: **return** H_t
14: **end if**
15: $H_t(\mathbf{x}_i) = H_{t-1}(\mathbf{x}_i) + \alpha_t h_t(\mathbf{x}_i)$
16: **end for**
17: **return** $\psi(\mathbf{x}) = \text{sign}\Big(H_t(\mathbf{x})\Big)$

4.4 Method: BSSL++ with AdaBoost.SDM

In order to replace AdaBoost-like (Algorithm 4) algorithm in BSSL++ (line 8 in Algorithm 3) with AdaBoost.SDM we note the following changes:

Table 4.1: Summary of binary classification datasets.

Dataset	No. of features	No. of samples
Climate model simulation crashes (`climate`)	20	540
Castmetal (`metal`)	37	327
Breast cancer (`breast-c`)	9	277
Liver disorders (`liver`)	5	177
Liver patients (`ilpd`)	10	583
Heart disease (`heart-d`)	13	107
Marketing (`marketing`)	32	364
Seismic bumps (`seismic`)	7	140
Thoracic surgery (`thoracic-s`)	16	470
Pro football scores (`profb`)	9	672
Australian credit approval (`aus`)	14	690
KC2 software defect (`kc2`)	21	522
Primary tumor (`primary-t`)	17	339
Protein localization sites (`ecoli`)	7	220
Single proton emission computed tomography (`spect`)	22	267
Apnea (`apnea`)	3	475
Sensory evaluation (`sensory`)	11	576
Backache in pregnancy (`backache`)	31	180
Chatfield (`chat`)	12	235
German credit (`german`)	20	1,000

- since BSSL++ expects from AdaBoost.SDM a linear combination of base classifiers, we return H_t instead of ψ; and

- since AdaBoost.SDM is designed for binary classification tasks, we change the H_{final} used in BSSL++ (line 9 in Algorithm 3) to

$$H_{\text{final}}(\mathbf{x}) = \begin{cases} +1, & \text{if } \sum_{b=1}^{B} \sum_{t=1}^{T} \alpha_b^t H_{t,b}(\mathbf{x}) > 0, \\ -1, & \text{otherwise,} \end{cases} \tag{4.27}$$

  where b corresponds to the while loop counter used in BSSL++ (line 1 in Algorithm 3).

We refer BSSL++ with AdaBoost.SDM algorithm as BSSL++.SDM.

4.5 Experimental settings: AdaBoost.SDM

We compare the proposed AdaBoost.SDM with the classification performance of several benchmark approaches that use Laplacian regularization.

Table 4.2: Summary of hyperparameter spaces.

Model	Parameter	Search space
AdaBoost	ψ	{decision stump}
	T	$\{20\}$
AdaBoost.SDM	ψ	{decision stump}
	T	$\{20\}$
	ρ	$\{10, 30, 50, 70\}$
	λ	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$
BoM	λ	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$
	number of neighbors	$\{5, 7, 9, 11\}$
LapRLS	λ_k	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$
	λ_u	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$
	number of neighbors	$\{5, 7, 9, 11\}$
LapSVM	λ_k	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$
	λ_u	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$
	number of neighbors	$\{5, 7, 9, 11\}$

4.5.1 Baseline and benchmark methods

To assess the AdaBoost.SDM, we include the results of AdaBoost to show the baseline performance. In order to compare with state-of-the-art Laplacian regularization techniques, we obtain the results using LapRLS, LapSVM, and boosting on manifolds (BoM), which serve as benchmarks. Additionally, we include the results of TabNet, which is a deep neural network (DNN) model for tabular datasets (Arik and Pfister, 2021). These results allow us to compare our method with a recent model proposed for the family of DNNs.

Hyperparameters. The fixed and tuned hyperparameters of AdaBoost, AdaBoost.SDM, LapRLS, LapSVM, and BoM are presented in Table 4.2. We note that for AdaBoost.SDM, we tune the scale parameter σ used for computing weighted adjacency matrix $\mathbf{W}$. In particular, σ was searched to be ρ^{th} percentile of the distribution of $e^{-||\mathbf{x}_i - \mathbf{x}_j||_2^2/2}$ for all pairs of $\mathbf{x}_i$ and $\mathbf{x}_j$, where ρ varied from 10 to 70, with a step size of 20. At the same time, the parameters of TabNet were fixed as suggested in (Arik and Pfister, 2021). For hyperparameter tuning, we used a stratified 5-fold stratified cross-validation in a brute-force manner.

4.5.2 Datasets

To perform experiments, we collected 20 real datasets from the OpenML dataset repository. Table 4.1 presents the summary of these binary classification datasets.

Table 4.3: The average performance in terms of test-set accuracy of the algorithms across 20 datasets.

dataset	AdaBoost	LapRLS	LapSVM	BoM	TabNet	AdaBoost.SDM
climate	90.2±1.4	91.0±0.9	**91.3±0.1**	90.0±1.0	89.5±1.9	91.2±0.6
metal	84.3±2.4	86.0±2.5	**86.3±2.2**	85.1±1.4	86.2±1.2	85.9±2.0
breast-c	72.6±4.1	71.5±4.1	68.4±13.2	72.6±3.8	71.7±6.0	**74.9±3.7**
liver	89.3±2.9	83.6±5.5	**91.0±1.7**	89.6±3.5	90.8±2.2	90.3±2.7
ilpd	69.7±3.2	68.4±2.7	70.1±2.3	70.0±2.9	70.3±1.7	**70.5±1.6**
heart-d	52.4±9.2	49.7±7.2	50.5±6.4	53.2±9.9	52.7±8.4	**54.1±8.0**
marketing	64.9±2.0	59.5±4.4	**68.2±0.0**	66.0±2.1	58.9±3.7	67.4±1.9
seismic	95.8±4.3	96.7±3.1	95.8±3.8	96.1±3.5	**97.6±2.0**	96.7±3.0
thoracic-s	84.4±1.4	81.1±1.3	**85.1±0.2**	84.4±1.4	83.8±1.9	85.0±0.4
profb	65.9±2.4	63.7±3.2	67.0±1.3	65.8±1.9	**67.9±3.7**	66.7±2.2
aus	85.1±1.8	85.7±1.6	**86.0±1.7**	85.0±1.7	84.9±2.3	85.5±1.9
kc2	**82.7±2.1**	81.0±2.3	**82.7±1.6**	**82.7±2.4**	82.2±1.5	**82.7±2.5**
primary-t	84.6±2.5	82.2±3.2	82.9±2.2	84.7±2.4	83.5±3.1	**85.3±2.4**
ecoli	98.7±1.5	96.6±2.6	98.1±1.5	98.5±1.7	98.6±1.1	**98.9±1.4**
spect	**84.2±2.5**	82.1±3.6	81.2±2.5	83.3±3.3	77.8±5.1	**84.2±3.0**
apnea	90.3±2.2	92.5±1.3	91.4±8.6	88.9±2.2	**94.8±1.7**	89.7±1.9
sensory	63.1±2.8	61.5±3.1	61.3±2.7	63.1±2.8	**68.2±3.3**	63.3±2.6
backache	83.2±4.0	84.1±2.8	**85.9±1.7**	83.3±3.1	79.6±4.9	85.1±2.2
chat	87.7±2.7	87.7±3.9	87.7±3.5	87.9±2.9	87.7±3.2	**88.2±3.7**
german	**73.3±1.8**	71.7±1.6	72.4±1.7	73.0±2.0	71.6±2.9	**73.3±1.6**
avg.	80.1±2.9	78.8±3.0	80.2±2.9	80.2±2.8	79.9±3.1	**80.9±2.5**

4.5.3 Evaluation

Each dataset is randomly partitioned into training and test sets with a splitting ratio of 70:30 in a stratified manner. We evaluate the AdaBoost.SDM on classification performance using the held-out test set. We repeat the experiments 20 times to obtain the average classification performance using different labeled, unlabeled, and test sets.

4.6 Results: AdaBoost.SDM

Table 4.3 demonstrates the average performance in terms of test-set accuracy results for all datasets using AdaBoost, LapRLS, LapSVM, BoM, TabNet, and AdaBoost.SDM algorithms. In the last row of Table 4.3, the average performance of each method across all datasets is presented. The results show that:

- AdaBoost.SDM outperforms the baseline in 16 datasets. For example, for `breast-c` and `marketing` datasets, using AdaBoost.SDM the test-set accuracy improved by $\sim$2.3% and $\sim$2.5% compared to the baseline AdaBoost, respectively;

- AdaBoost.SDM shows better results then all Laplacian regularized algorithms in nine datasets; and
- AdaBoost.SDM outperforms TabNet in 14 datastes.

Although LapSVM demonstrated better performance in seven datasets, AdaBoost.SDM appears more efficient taking into account the higher cost of training LapSVM (Section 4.9 presents time complexities of training AdaBoost.SDM and LapSVM).

In summary, on average (across all datasets and all splits), AdaBoost.SDM achieved 80.9±2.5% accuracy. These results confirm that the proposed AdaBoost.SDM algorithm is highly competitive with the compared algorithms.

4.7 Experimental settings: BSSL++.SDM

We assess BSSL++.SDM using the same datasets used to test BSSL++ (see Section 3.4.2 in Chapter 3). In addition, to be able to compare the result of BSSL++ with BSSL++.SDM we employed the same data-splitting strategy. In particular, we divided each dataset into training and test sets using a splitting ratio of 80:20, where the training set was further split into unlabeled and labeled sets with the same ratios of $u : l = 90 : 10$ and $80 : 20$. Similar to BSSL++, we used MLP as a base classifier with the same parameters described in Section 3.4.3 of Chapter 3. At the same time, we tuned the parameters of AdaBoost.SDM in BSSL++.SDM using a stratified 5-fold stratified cross-validation. In particular, we used the same hyperparameter space specified in Table 4.2.

4.8 Results: BSSL++.SDM

Tables 4.4 and 4.5 demonstrate the average classification performance of BSSL++.SDM for cases of $u : l = 90 : 10$ and $80 : 20$, respectively. The results show that usage of AdaBoost.SDM in BSSL++ can lead to better performance compared to BSSL++ (in the original form). Particularly, BSSL++.SDM outperformed BSSL++ in seven and five datasets for $u : l = 90 : 10$ and $80 : 20$ cases, respectively. In addition, the results also demonstrate that, out of 117 performance comparisons for each algorithm (13 algorithms $\times$ 9 datasets), BSSL++.SDM outperformed all other algorithm in $\sim$90% and $\sim$91% of the comparisons for $u : l = 90 : 10$ and $80 : 20$, respectively.

4.9 Discussion

BSSL++ with AdaBoost.SDM. Recall that BSSL++ in its original form was designed based on *semi-supervised smoothness* and *cluster assumptions*. At the same time,

Table 4.4: The average classification results in terms of test-set accuracy in the form of mean±std ($u : l = 10 : 90$).

		aus	hms	hdc	hep	kvk	mm	pid	vote	wdbc	avg.
NB	ADAB	77.2±4.1	63.6±4.6	74.8±6.5	80.0±6.8	82.4±1.4	80.3±3.8	69.7±3.2	90.6±4.6	93.9±2.3	79.2±4.1
	ASMB	76.5±4.6	66.7±7.8	75.5±5.4	80.0±5.2	81.1±2.4	79.5±3.8	70.3±5.8	89.0±4.6	93.2±2.1	79.1±4.6
	SEMIB	78.4±3.3	69.2±5.7	69.5±7.1	81.0±4.9	80.4±2.8	80.1±3.7	68.6±2.9	88.6±3.9	93.3±3.1	78.8±4.2
	REGB	80.4±4.5	70.0±4.4	80.2±3.9	80.7±5.3	82.6±1.6	80.5±3.2	73.5±2.4	91.0±3.9	94.2±1.8	81.5±3.4
C4.5	ADAB	81.2±2.7	62.0±6.3	76.5±6.1	77.4±11.8	84.1±1.4	75.8±2.9	71.3±3.2	93.1±2.4	91.9±2.2	79.3±4.3
	ASMB	77.3±5.8	65.4±6.9	70.5±5.9	71.6±14.1	81.6±2.2	79.0±4.9	68.9±5.0	91.5±3.1	88.7±5.7	77.2±6.0
	SEMIB	77.7±6.3	69.8±4.7	63.7±7.2	79.6±4.6	81.9±2.5	78.9±2.5	69.4±4.3	91.9±2.2	91.4±2.4	78.3±4.1
	REGB	**83.0±3.5**	68.8±5.2	77.3±7.3	80.0±5.8	84.1±1.4	80.3±3.6	72.3±3.6	93.7±2.3	93.0±1.3	81.4±3.8
MLP	ADAB	77.7±3.9	71.0±6.5	75.2±4.9	65.8±29.9	87.2±15	**82.1±2.3**	72.9±3.0	91.0±4.3	**95.4±2.5**	79.8±8.0
	ASMB	78.9±3.7	72.4±5.5	76.7±4.7	79.7±8.0	89.1±2.1	81.1±3.9	73.3±3.3	91.3±4.2	93.8±3.1	81.8±4.3
	SEMIB	68.5±2.8	**74.2±0.0**	66.2±4.9	82.6±3.6	60.0±3.1	80.5±3.9	65.0±0.0	90.3±2.1	70.8±3.4	73.1±2.6
	REGB	81.2±3.2	**74.2±0.0**	**80.3±5.2**	82.6±5.6	82.2±4.0	79.9±1.9	67.9±2.6	89.8±2.4	86.7±5.7	80.5±3.4
	BSSL++	82.2±2.7	**74.2±0.0**	79.5±4.2	81.0±7.4	90.4±1.4	81.5±3.7	73.2±3.4	91.3±2.7	**95.4±1.7**	83.2±3.0
	BSSL++.SDM	82.9±2.8	73.0±4.4	80.1±5.0	**82.7±6.1**	**90.7±1.7**	81.6±3.7	**73.6±3.7**	91.8±2.6	95.1±1.7	**83.5±3.1**
FT-T	FixMatch	82.8±3.6	70.5±4.9	78.7±6.7	79.4±9.3	60.9±15.6	81.1±2.4	70.5±3.9	**93.4±1.8**	88.4±12.9	78.4±6.8

Table 4.5: The average classification results in terms of test-set accuracy in the form of mean±std ($u : l = 80 : 20$).

		aus	hms	hdc	hep	kvk	mm	pid	vote	wdbc	avg.
NB	ADAB	79.6±3.2	70.5±7.8	76.7±5.9	80.6±4.8	83.9±1.2	80.5±3.5	72.6±2.2	93.6±1.6	95.0±2.9	81.4±3.7
	ASMB	76.6±6.0	72.1±5.2	79.3±3.3	80.3±8.4	84.1±1.4	79.8±3.3	74.0±4.1	91.3±2.6	94.2±2.7	81.3±4.1
	SEMIB	78.7±2.6	70.7±4.4	74.0±7.3	81.6±6.6	84.0±1.2	80.3±3.6	72.2±3.0	89.9±2.8	94.3±2.1	80.6±3.7
	REGB	81.7±3.0	72.6±6.4	80.5±3.5	81.0±8.7	84.2±1.4	80.6±3.5	74.2±4.4	92.6±2.0	94.4±2.9	82.4±4.0
C4.5	ADAB	83.1±3.7	64.4±3.8	77.7±4.0	79.3±10.1	87.2±0.9	76.5±3.5	72.5±4.0	93.8±1.1	95.2±1.9	81.1±3.7
	ASMB	78.3±4.3	67.7±8.3	75.5±7.3	73.9±7.3	84.5±1.0	79.4±3.1	71.5±3.1	91.8±4.0	91.1±1.7	79.3±4.5
	SEMIB	82.2±5.1	70.2±4.6	71.1±6.7	80.0±5.9	85.2±1.2	80.7±1.3	72.5±3.9	93.4±2.7	92.1±2.9	80.8±3.8
	REGB	**84.5±3.4**	69.0±8.0	79.0±6.9	80.3±6.9	87.8±0.8	80.5±3.2	72.9±4.1	93.9±2.2	95.3±1.8	82.6±4.1
MLP	ADAB	80.7±3.7	74.4±5.9	73.6±7.8	81.0±5.5	83.2±21.3	82.4±1.8	74.6±3.7	93.1±2.7	**96.6±1.6**	82.2±6.0
	ASMB	81.4±3.5	74.0±3.8	80.7±3.1	81.3±5.5	93.7±1.3	81.1±3.4	**76.1±2.9**	92.5±4.0	95.3±1.3	84.0±3.2
	SEMIB	67.0±4.6	74.2±0.0	73.9±4.8	82.3±3.0	61.6±2.1	80.6±3.3	65.0±0.7	89.4±3.6	73.6±3.7	74.2±2.9
	REGB	82.2±3.7	74.2±0.0	78.7±4.1	82.9±5.6	81.7±4.0	80.6±2.4	70.5±2.1	90.7±2.7	89.9±3.9	81.3±3.2
	BSSL++	83.8±2.5	74.3±0.8	82.3±5.5	84.4±5.7	**94.3±1.4**	**82.7±3.8**	**76.1±1.7**	93.1±3.0	95.8±1.9	**85.2±2.9**
	BSSL++.SDM	84.3±2.8	**74.5±1.4**	**83.0±5.2**	**84.5±5.5**	93.3±1.6	82.4±3.6	**76.1±2.3**	93.3±2.4	95.6±2.1	**85.2±3.0**
FT-T	FixMatch	79.9±8.2	71±5.2	78.2±9.1	80.3±6.5	52.2±0	81.7±3.2	73.7±3.7	**94.3±2.9**	92.5±9.9	78.2±5.4

AdaBoost.SDM, which at its core uses mixed-graph Laplacian, is based on *manifold assumption*. That being said, to encode the manifold assumption into BSSL++, we used AdaBoost.SDM to train an ensemble classifier instead of the AdaBoost-like classifier proposed by (Polikar et al., 2001). The results from Tables 4.4 and 4.5 suggest that encoding manifold assumption via similarity and dissimilarity-based manifold regularized AdaBoost can possibly capture the intrinsic structure of the data manifold and improve the performance of BSSL++.

Time complexity of AdaBoost.SDM. The complexity of AdaBoost with decision stump is $\mathcal{O}(Tlp)$, where T is the maximum number of base learners (i.e., the maximum number of boosting iterations), l is the size of the training set, and p is the number of features. Since the AdaBoost.SDM is the extension of AdaBoost, its time complexity is

$\mathcal{O}(Tlp + l^2p + Tl^2)$, where $\mathcal{O}(l^2p)$ is needed to compute the matrix $\mathbf{W}$, and $\mathcal{O}(Tl^2)$ is needed to compute (4.17) and (4.18). For comparison, the time complexity of LapSVM is $\mathcal{O}(l^2p + l^3)$ (Belkin et al., 2006). In practice, we generally have $T << l$, therefore AdaBoost.SDM tends to be more efficient compared to LapSVM. To show the runtime differences, we collected the average training time (across 20 repetitions) of both algorithms using `german` dataset with a fixed hyperparameter set. For the runtime experiments, we used a Windows 10 workstation with an Intel(R) Xeon(R) W-2145 CPU (3.7 GHz). The results show that AdaBoost.SDM required 0.44±0.01 seconds to train the classifier, while LapSVM required 95.72±8.95 seconds (i.e., around 200 times slower).

Time complexity of BSSL++.SDM. To derive the time complexity of BSSL++.SDM, recall that the time complexity of BSSL++ in its original form is $\mathcal{O}(kps \log u(r + pTe + Tp \log u))$ where k is the number of classes, $s = u + l$ is the total number of unlabeled and labeled data points, r is the maximum number of iterations to solve k-means problem using Lloyd's algorithm (Lloyd, 1982), and e is the number of epochs (see Section 3.6 in Chapter 3). Now, replacing the AdaBoost-like classifier in BSSL++ with AdaBoost.SDM results in time complexity of $\mathcal{O}(ks \log u(pr + p^2Te + ps + Ts + Tp^2 \log u))$.

Table 4.6: Summary of multi-class classification datasets.

no.	dataset	no. of samples	no. of features	no. of classes
1	`glass`	214	9	6
2	`solar-flare`	315	12	5
3	`cleveland`	303	7	5
4	`thyroid`	215	5	3
5	`splice`	3,190	60	3

Table 4.7: The average performance in terms of test-set accuracy of the algorithms across 5 datasets.

	AdaBoost	LapRLS	LapSVM	BoM	TabNet	AdaBoost.SDM
`glass`	68.9±6.7	70.2±5.3	69.8±5.7	69.8±5.7	67.2±5.1	**70.3±6.3**
`solar-flare`	71.2±3.9	66.4±3.1	66.7±4.4	70.4±3.4	**74.6±2.6**	71.9±3.9
`cleveland`	58.1±3.0	52.7±4.4	56.0±1.4	57.7±4.4	55.2±3.2	**58.4±2.9**
`thyroid`	95.7±3.0	**96.2±3.2**	95.2±3.0	**96.2±2.1**	90.0±4.0	**96.2±2.6**
`splice`	87.0±1.4	85.5±0.9	74.4±9.2	92.1±1.3	**94.5±1.0**	92.6±0.9
avg.	76.2±3.6	74.2±3.4	72.4±4.7	77.2±3.4	76.3±3.2	**77.9±3.3**

One-versus-rest multi-class experiments. To assess the performance of AdaBoost.SDM in multi-class settings, we studied extending the method to multi-class

setting using a one-versus-rest (OvR) classification strategy. In this approach, a separate binary classifier is trained for each class against all others, and final predictions are made by selecting the class with the highest confidence score. We evaluated the multi-class version of AdaBoost.SDM on five benchmark datasets collected from OpenML dataset repository (see Table 4.6). The experimental setup was similar to Section 4.5, except we repeated experiments ten times using different splits of labeled, unlabeled, and test sets. Table 4.7 shows the average test-set accuracy results of all methods across five datasets. The results indicate that AdaBoost.SDM can achieve competitive performance in multi-class scenarios.

Limitations. The limitation of the proposed AdaBoost.SDM (and BSSL++.SDM) is its restriction to binary classification problems. While we demonstrated that the method can be extended to multi-class problems using a one-versus-rest (OvR) strategy and achieves promising results across several datasets, this approach does not provide a principled solution to the multi-class setting. As discussed in (Bishop, 2006), combining multiple binary classifiers to construct a multi-class decision function can result in ambiguous decision regions. To address this limitation, AdaBoost.SDM can be extended to support multi-class classification using a similar approach that was employed in AdaBoost Stagewise Additive Modeling (SAMME) (Hastie et al., 2009) algorithm. On the other hand, one can extend the manifold regularization used in our work to a multi-class setting using the same approach as in (Goldberg et al., 2007). In particular, these approaches extend to the multi-class setting using the following class label encoding: for a k-class classification task, the class labels can be encoded to k-dimensional vector with all entries as $-1/(k-1)$ except for the entry to which class belongs (e.g., k^{th} class), which is set to 1. We leave these investigations for future studies.

4.10 Summary

In this chapter, we proposed AdaBoost.SDM, which is the similarity and dissimilarity-based manifold regularized AdaBoost algorithm. In particular, the regularization is based on label similarity and dissimilarity knowledge between data points. The experimental results demonstrate that the proposed algorithm can effectively exploit manifold regularization to achieve promising results. At the same time, experiments with BSSL++ using AdaBoost.SDM reveal the applicability of manifold assumption into BSSL++. The results show that by taking into account the manifold assumption, we can improve the performance of BSSL++ in its original form.

Chapter 5

Efficient Active Learning using Recursive Estimation of Error Reduction

This chapter describes another approach for using unlabeled data for small labeled sample classification tasks by constructing efficient labeled training data while reducing labeling costs.

5.1 Introduction

Given a *labeling budget* of $\tilde{b}$ data points, active learning (AL) aims to actively build a labeled training dataset by querying the most beneficial data points for training a classifier from an available pool of unlabeled data points. Thus, AL helps to reduce the cost of producing the labels and redundancy by focusing more on labeling the most "informative" observations. In this regard, different proposed AL algorithms are based on how they define and identify informative data points.

There have been proposed AL algorithms that are mostly based on either *diversity sampling* (Gao et al., 2020; Shui et al., 2020) or *uncertainty sampling* (Houlsby et al., 2011; Kirsch et al., 2019). In diversity sampling, the approach focuses on querying the data points in diverse regions of the data distribution. On the other hand, uncertainty sampling ensures the selection of data points that a classifier trained using labeled data is least certain about. This can be obtained by predicting the labels for unlabeled data points by the classifier. However, both approaches consider querying heterogeneous data points. As a result, these methods do not take into account the effect of an unlabeled data point on the performance of the classifier once its label is obtained from the annotator and added to the training set (Aggarwal et al., 2014).

In contrast to the above-mentioned methods, the AL method, which is referred to as *estimated error reduction (EER)*, considers reducing the generalization error (Roy

and McCallum, 2001; Moskovitch et al., 2007). Particularly, the aim of the EER is to identify the $\tilde{b}$ "most informative" unlabeled data points in a one-step-look-ahead manner (Aggarwal et al., 2014; Tharwat and Schenck, 2023). That is to say, in each step, the most informative data point is the one that reduces the estimate of the error rate the most if employed in training, along with all available labeled data.

EER can be formalized as follows. Consider a classification task with a labeled training set containing l labeled data points and their labels, $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ where $\mathbf{x}_i \in \mathbb{R}^p$ represents a p-dimensional feature vector, y_i denotes the class label associated with $\mathbf{x}_i$. In classification, $\mathcal{L}$ is used to construct a classifier $\psi(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathcal{Y}$, where $\mathcal{Y} = \{0, 1, \ldots, k-1\}$ in which k is the number of classes. Consider plug-in classification rules, where a classifier is constructed by estimating the unknown posterior probability of class Y given an input $\mathbf{X} = \mathbf{x}$. Let $P(Y|\mathbf{x})$ denote the (unknown) true posterior probability, and $\hat{P}_{\mathcal{L}}(Y|\mathbf{x})$ denote its corresponding estimate. Hence, the error of the classifier with given some loss function L, denoted $\varepsilon_{\mathcal{L}}^{L}$, is given by (Roy and McCallum, 2001):

$$\varepsilon_{\mathcal{L}}^{L} = \int L\Big(P(Y|\mathbf{x}), \hat{P}_{\mathcal{L}}(Y|\mathbf{x})\Big) p(\mathbf{x}) d\mathbf{x}\,, \tag{5.1}$$

where $p(\mathbf{x})$ is the marginal probability density function of $\mathbf{X}$ at $\mathbf{x}$. The probability of misclassification, that is the error of the classifier, using conventional 0/1 loss, denoted $\varepsilon_{\mathcal{L}}$, can be written as

$$\varepsilon_{\mathcal{L}} = P\Big(\psi(\mathbf{X}) \neq Y\Big) = \int \Big(1 - \max_{y \in \mathcal{Y}} \hat{P}_{\mathcal{L}}(Y = y|\mathbf{x})\Big) p(\mathbf{x}) d\mathbf{x}\,. \tag{5.2}$$

Given a set of u unlabeled data points $\mathcal{U} = \{\mathbf{x}_i^U\}_{i=1}^{u}$, EER successively finds the unlabeled data point $\mathbf{x}_O \triangleq \mathbf{x}_j^U \in \mathcal{U}$ that corresponds to the lowest weighted average error across all candidate labels for $\mathbf{x}_j^U$, if $\mathbf{x}_j^U$ and its label is included to the labeled training set. In other words, considering classification error rate (5.2), in the first iteration $\mathbf{x}_O$ can be found as

$$\mathbf{x}_O = \underset{\mathbf{x}_j^U \in \mathcal{U}}{\operatorname{argmin}} \sum_{y^\star \in \mathcal{Y}} \hat{P}_{\mathcal{L}}(Y = y^\star|\mathbf{x}_j^U) \varepsilon_{\mathcal{L} \cup \{(\mathbf{x}_j^U, y^\star)\}}\,. \tag{5.3}$$

Nevertheless, to determine $\mathbf{x}_O$ in (5.3), $p(\mathbf{x})$ in (5.2) is substituted with an empirical density function $P(\mathbf{X} = \mathbf{x}_j^U) = \frac{1}{u}, \forall j$. This puts an equal mass at the observed samples and yields to an estimate of $\varepsilon_{\mathcal{L} \cup \{(\mathbf{x}_j^U, y^\star)\}}$, denoted $\hat{\varepsilon}_{\mathcal{L} \cup \{(\mathbf{x}_j^U, y^\star)\}}$, which is given by (Roy and McCallum, 2001)

$$\hat{\varepsilon}_{\mathcal{L} \cup \{(\mathbf{x}_j^U, y^\star)\}} = \frac{1}{u} \sum_{\mathbf{x}_k^U \in \mathcal{U}} \left(1 - \max_{y \in \mathcal{Y}} \hat{P}_{\mathcal{L} \cup \{(\mathbf{x}_j^U, y^\star)\}}(Y = y|\mathbf{x}_k^U)\right). \tag{5.4}$$

As soon as $\mathbf{x}_O$ is identified, the data labeling expert will assign the true label, and $\mathbf{x}_O$ and its label will be added to the set of labeled data points.

From an algorithmic point of view, EER methods generally use (5.3) and (5.4) in the following fashion: (i) a classifier is trained using a labeled training set; (ii) the training set is updated by adding a candidate unlabeled data point with a candidate class label; (iii) the updated training set is used to retrain the classifier; (iv) the retrained classifier is evaluated by computing (5.4); (v) steps (ii)-(iv) are repeated over all possible class labels—this represented by the summation in (5.3); (vi) steps (ii)-(v) are repeated for all unlabeled observations from the pool; and (vii) the $\tilde{b}$ unlabeled data points that with the least weighted average error are queried for annotation. As it has been pointed out in earlier works, this repetitive "retraining-evaluation" procedure requires substantially high computational resources (Aggarwal et al., 2014; Tharwat and Schenck, 2023). Particularly, it runs the learning algorithm for every unlabeled observation and every potential class label, which yields in training and evaluating $u \times k$ classifiers.

Earlier attempts have been made to mitigate the computational limitations of EER by avoiding retraining the classifier from scratch. For instance, Zhang *et al.* (Zhang et al., 2017) proposed an approximation approach for updating Logistic Regression model and employed an incremental variant of the Support Vector Machine. Roy and McCallum (Roy and McCallum, 2001) employed Fast Naive Bayes to increase the efficiency of retraining. However, computational requirements still hinder the use of EER-based for large unlabeled data. Hence, an efficient mechanism for estimating the effect of labeling the particular unlabeled data point on the performance of the classifier is still required.

In this chapter, we focus on reducing the computational requirements of the EER method. Specifically, we intend to substitute the repetitive retraining-evaluation procedure with a re-evaluation. Having said that, to achieve this objective, the following assumptions are made:

Assumption 5.1.1. *The method is focused on the probability of misclassification given in (5.2) for the binary classification task, where observations are randomly sampled from two populations* Π_0 *and* Π_1 *with prior probability* α_0 *and* α_1*, respectively.*

Assumption 5.1.2. *The posterior weights* $\hat{P}_{\mathcal{L}}(Y = y^{\star}|\mathbf{x}_j^U)$ *utilized in (5.3) are substituted with class prior probabilities.*

Assumption 5.1.3. *Regularized discriminant analysis (RLDA) classifier (Duda et al., 2001) is used for screening the pool of unlabeled data points.*

As a result, we propose a computationally efficient EER-based AL method that minimizes an accurate closed-form error estimation of RLDA (AL-RLDA). In particular, the method uses a re-evaluation procedure, which can be viewed as integrating the retraining procedure entirely into the evaluation step. Moreover, we derive recursive update rules to efficiently compute the re-evaluation procedure for screening unlabeled data points.

5.2 Theory: Estimated Error Reduction with Prior Probabilities

Based on Assumption 5.1.2, we propose the following objective function for EER, formulated based on prior probabilities, which we refer to as EER with prior probabilities (EER-P)

$$\mathbf{x}_O = \underset{\mathbf{x}_j^U \in \mathcal{U}}{\operatorname{argmin}} \sum_{y^\star \in \mathcal{Y}} \alpha_{y^\star} \varepsilon_{\mathcal{L} \cup \{(\mathbf{x}_j^U, y^\star)\}} \,, \tag{5.5}$$

where $\alpha_{y^\star}$ is the class prior probability. As pointed out by Roy and McCallum (Roy and McCallum, 2001), using the objective function with posterior probabilities (5.3), the selection of data points is directed if data points reinforce the learner's existing belief over unlabeled set. That being said, the proposed EER-P is designed to maintain the balance between the existing belief of the current classifier and the problem's nature captured in the class prior probability of data points in $\mathcal{U}$. This is beneficial in two ways:

1. EER-P reduces dependence on the current classifier. That is, the method that selects the most informative data points and the final classifier does not need to be the same. In other words, we can employ RLDA for "screening" and selection of data points for annotation, however the final classifier can be trained utilizing a different learning algorithm (discussion about this point is given in Section 5.7); and

2. EER-P allows to use of a re-evaluation procedure instead of retraining-evaluation that is employed in the conventional EER. To elaborate, let us consider the objective function of conventional EER given in (5.3). Despite the provided an efficient estimator of $\varepsilon_{\mathcal{L} \cup \{(\mathbf{x}_j^U, y^\star)\}}$, dependence of (5.3) on the classifier itself via $\hat{P}_{\mathcal{L}}(Y = y^\star | \mathbf{x}_j^U)$ does not allow avoiding retraining-evaluation procedure. In contrast, in novel objective function EER-P, the proportion of data points from class $y^\star$ can be used to consistently estimate $\alpha_{y^\star}$ under random sampling assumption (Braga-Neto et al., 2014).

5.3 Theory: RLDA Classifier and Its Error Estimator

Let $\mathcal{L}_k = \{\mathbf{x}_i\}_{i=1}^{l_k}$ denote a labeled set of size l_k from class k, where $k = 0, 1$, and $l_0 + l_1 = l$. For the binary classification task, the objective function (5.5), which describes the identification of the unlabeled data point with the least weighted average error rate, can be written as

$$\mathbf{x}_O = \underset{\mathbf{x}_j^U \in \mathcal{U}}{\operatorname{argmin}} \, \alpha_0 \, \varepsilon^{\alpha_0, \alpha_1}_{\mathcal{L}_0 \cup \{\mathbf{x}_j^U\}, \mathcal{L}_1} + \alpha_1 \, \varepsilon^{\alpha_0, \alpha_1}_{\mathcal{L}_0, \mathcal{L}_1 \cup \{\mathbf{x}_j^U\}} \,, \tag{5.6}$$

where

$$\varepsilon^{\alpha_0,\alpha_1}_{\mathcal{L}_0,\mathcal{L}_1} = \alpha_0\, \varepsilon^{\alpha_0,\alpha_1}_{0,\mathcal{L}_0,\mathcal{L}_1} + \alpha_1\, \varepsilon^{\alpha_0,\alpha_1}_{1,\mathcal{L}_0,\mathcal{L}_1}, \tag{5.7}$$

and where $\varepsilon^{\alpha_0,\alpha_1}_{0,\mathcal{L}_0,\mathcal{L}_1}$ and $\varepsilon^{\alpha_0,\alpha_1}_{1,\mathcal{L}_0,\mathcal{L}_1}$ denote the class-specific error rate. Specifically, for $i = 0, 1$, $\varepsilon^{\alpha_0,\alpha_1}_{i,\mathcal{L}_0,\mathcal{L}_1}$, which is a function of α_0 and α_1, is the probability of misclassifying a data point from class i given as

$$\varepsilon^{\alpha_0,\alpha_1}_{i,\mathcal{L}_0,\mathcal{L}_1} = P\Big(\psi(\mathbf{X}) \neq Y \mid \mathbf{x} \in \Pi_i, \mathcal{L}_0, \mathcal{L}_1\Big). \tag{5.8}$$

RLDA dichotomizer, denoted $\psi^{\text{RLDA}}(\mathbf{x})$, is given by (Di Pillo, 1976; Friedman, 1989; Zollanvari and Dougherty, 2015)

$$\psi^{\text{RLDA}}(\mathbf{x}) = \begin{cases} 1\,, & \text{if } W^{\text{RLDA}}(\mathbf{x}) \leq 0 \\ 0\,, & \text{if } W^{\text{RLDA}}(\mathbf{x}) > 0 \end{cases}, \tag{5.9}$$

where

$$W^{\text{RLDA}}(\mathbf{x}) = \gamma \left(\mathbf{x} - \frac{\bar{\mathbf{x}}_{\mathcal{L}_0} + \bar{\mathbf{x}}_{\mathcal{L}_1}}{2}\right)^T \mathbf{H}_{\mathcal{L}_0,\mathcal{L}_1}(\bar{\mathbf{x}}_{\mathcal{L}_0} - \bar{\mathbf{x}}_{\mathcal{L}_1}) - \log\frac{\alpha_1}{\alpha_0}\,, \tag{5.10}$$

and where $\gamma > 0$,

$$\bar{\mathbf{x}}_{\mathcal{L}_i} = \frac{1}{l_i} \sum_{\mathbf{x}_l \in \mathcal{L}_i} \mathbf{x}_l, \tag{5.11}$$

$$\mathbf{H}_{\mathcal{L}_0,\mathcal{L}_1} = (\mathbf{I}_p + \gamma \hat{\mathbf{\Sigma}}_{\mathcal{L}_0,\mathcal{L}_1})^{-1}, \tag{5.12}$$

$$\hat{\mathbf{\Sigma}}_{\mathcal{L}_0,\mathcal{L}_1} = \frac{(l_0 - 1)\hat{\mathbf{\Sigma}}_{\mathcal{L}_0} + (l_1 - 1)\hat{\mathbf{\Sigma}}_{\mathcal{L}_1}}{l_0 + l_1 - 2}, \tag{5.13}$$

$$\hat{\mathbf{\Sigma}}_{\mathcal{L}_i} = \frac{1}{l_i - 1} \sum_{\mathbf{x}_l \in \mathcal{L}_i} (\mathbf{x}_l - \bar{\mathbf{x}}_{\mathcal{L}_i})(\mathbf{x}_l - \bar{\mathbf{x}}_{\mathcal{L}_i})^T\,, \tag{5.14}$$

and $\mathbf{I}_p$ denotes the identity matrix with $p \times p$ size . Note that we need to compute the class-specific error rates $\varepsilon^{\alpha_0,\alpha_1}_{i,\mathcal{L}_0,\mathcal{L}_1}$ in (5.8) to identify $\mathbf{x}_O$ from (5.6). Since $\varepsilon^{\alpha_0,\alpha_1}_{i,\mathcal{L}_0,\mathcal{L}_1}$ is a function of unknown feature-label distributions, it needs to be estimated. However, instead of relying on resampling-based error estimation rules such as bootstrap or cross-validation, which would lead to greater computational requirements of the EER algorithm, we leverage a closed-form error estimator of RLDA proposed in (Zollanvari and Dougherty, 2015). Particularly, Zollanvari *et al.* (Zollanvari and Dougherty, 2015) developed the generalized consistent estimator of $\varepsilon^{\alpha_0,\alpha_1}_{i,\mathcal{L}_0,\mathcal{L}_1}$ of RLDA by considering the following sequence of Gaussian discrimination problems relative to RLDA:

$$\Xi_p = \{\Pi_0, \Pi_1, \boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \mathbf{\Sigma}, l_0, l_1, W^{\text{RLDA}}, \varepsilon^{\alpha_0,\alpha_1}_{i,\mathcal{L}_0,\mathcal{L}_1}\}\,, \quad p = 1, \ldots, \tag{5.15}$$

where all parameters are functions of p (which is omitted for ease the notation), and Ξ_p is restricted by the following conditions:

(I) Π_i follows a multivariate Gaussian distribution $N(\boldsymbol{\mu}_i, \mathbf{\Sigma})$.

(II) $l_0 \to \infty, l_1 \to \infty, p \to \infty$, and the following limits exist: $\frac{p}{l_0} \to J_0 > 0$, $\frac{p}{l_1} \to J_1 > 0$, and $\frac{p}{l_0+l_1} \to J < \infty$.

(III) All eigenvalues of $\mathbf{\Sigma}$ are located in a segment $[c_1, c_2]$, where $c_1 > 0$ and c_2 does not depend on p.

(IV) $|\boldsymbol{\mu}_i|$ is bounded over all p, that is, there exists Q such that $|\boldsymbol{\mu}_i| < Q$ for $i = 0, 1$ and $p = 1, 2, \ldots$.

Under the stated conditions, the following theorem holds.

Theorem 5.3.1. *Under conditions I-IV (Section III.A in (Zollanvari and Dougherty, 2015)):*

$$\varepsilon_{\mathcal{L}_0,\mathcal{L}_1}^{\alpha_0,\alpha_1} - \hat{\varepsilon}_{\mathcal{L}_0,\mathcal{L}_1}^{\alpha_0,\alpha_1} \overset{a.s.}{\to} 0, \tag{5.16}$$

where

$$\hat{\varepsilon}_{\mathcal{L}_0,\mathcal{L}_1}^{\alpha_0,\alpha_1} = \alpha_0\, \hat{\varepsilon}_{0,\mathcal{L}_0,\mathcal{L}_1}^{\alpha_0,\alpha_1} + \alpha_1\, \hat{\varepsilon}_{1,\mathcal{L}_0,\mathcal{L}_1}^{\alpha_0,\alpha_1}, \tag{5.17}$$

and

$$\hat{\varepsilon}_{i,\mathcal{L}_0,\mathcal{L}_1}^{\alpha_0,\alpha_1} = \Phi\left(\frac{(-1)^{i+1}\hat{G}_{i,\mathcal{L}_0,\mathcal{L}_1}^{\alpha_0,\alpha_1}}{\sqrt{\hat{D}_{\mathcal{L}_0,\mathcal{L}_1}}}\right), \tag{5.18}$$

$$\hat{G}_{i,\mathcal{L}_0,\mathcal{L}_1}^{\alpha_0,\alpha_1} = G_{\bar{\mathbf{x}}_{\mathcal{L}_i},\mathcal{L}_0,\mathcal{L}_1}^{\alpha_0,\alpha_1} + (-1)^{i+1}\frac{(l_0+l_1-2)\hat{\delta}_{\mathcal{L}_0,\mathcal{L}_1}}{l_i} - \frac{1}{\gamma}\left(\log\frac{\alpha_1}{\alpha_0}\right), \tag{5.19}$$

$$\hat{D}_{\mathcal{L}_0,\mathcal{L}_1} = (1+\gamma\hat{\delta}_{\mathcal{L}_0,\mathcal{L}_1})^2 D_{\hat{\mathbf{\Sigma}}_{\mathcal{L}_0,\mathcal{L}_1},\mathcal{L}_0,\mathcal{L}_1}, \tag{5.20}$$

$$\hat{\delta}_{\mathcal{L}_0,\mathcal{L}_1} = \frac{\frac{p}{l_0+l_1-2} - \frac{\mathrm{tr}[\mathbf{H}_{\mathcal{L}_0,\mathcal{L}_1}]}{l_0+l_1-2}}{\gamma\left(1 - \frac{p}{l_0+l_1-2} + \frac{\mathrm{tr}[\mathbf{H}_{\mathcal{L}_0,\mathcal{L}_1}]}{l_0+l_1-2}\right)}, \tag{5.21}$$

$$G_{\bar{\mathbf{x}}_{\mathcal{L}_i},\mathcal{L}_0,\mathcal{L}_1} = \left(\bar{\mathbf{x}}_{\mathcal{L}_i} - \frac{\bar{\mathbf{x}}_{\mathcal{L}_0} + \bar{\mathbf{x}}_{\mathcal{L}_1}}{2}\right)^T \mathbf{H}_{\mathcal{L}_0,\mathcal{L}_1}\left(\bar{\mathbf{x}}_{\mathcal{L}_0} - \bar{\mathbf{x}}_{\mathcal{L}_1}\right), \tag{5.22}$$

$$D_{\hat{\mathbf{\Sigma}}_{\mathcal{L}_0,\mathcal{L}_1},\mathcal{L}_0,\mathcal{L}_1} = (\bar{\mathbf{x}}_{\mathcal{L}_0} - \bar{\mathbf{x}}_{\mathcal{L}_1})^T \mathbf{H}_{\mathcal{L}_0,\mathcal{L}_1}\hat{\mathbf{\Sigma}}_{\mathcal{L}_0,\mathcal{L}_1}\mathbf{H}_{\mathcal{L}_0,\mathcal{L}_1}(\bar{\mathbf{x}}_{\mathcal{L}_0} - \bar{\mathbf{x}}_{\mathcal{L}_1}), \tag{5.23}$$

and where $\overset{a.s.}{\to}$ denotes almost sure convergence, $\Phi(\cdot)$ denotes the cumulative distribution function of a standard normal random variable, and $\mathrm{tr}[\cdot]$ *is the trace operator.* ■

Theorem 5.3.1 describes that the estimator $\hat{\varepsilon}_{i,\mathcal{L}_0,\mathcal{L}_1}^{\alpha_0,\alpha_1}$ converges almost surely (under conditions I-IV) to $\varepsilon_{i,\mathcal{L}_0,\mathcal{L}_1}^{\alpha_0,\alpha_1}$. Despite in condition I, the data is assumed to be drawn from Gaussian distribution, the estimator has demonstrated promising results in estimating the true error of RLDA on real data as well (Zollanvari and Dougherty, 2015; Bakir et al., 2016).

5.4 Method: AL-RLDA

The proposed AL-RLDA method successively finds $\mathbf{x}_O \in \mathcal{U}$ that corresponds to the least estimate of the EER-P for the RLDA classifier. Particularly, in the first iteration

$\mathbf{x}_O$ can be found using

$$\mathbf{x}_O = \underset{\mathbf{x}_j^U \in \mathcal{U}}{\operatorname{argmin}}\ \hat{\alpha}_0 \hat{\varepsilon}^{\hat{\alpha}_0,\hat{\alpha}_1}_{\mathcal{L}_0 \cup \{\mathbf{x}_j^U\},\, \mathcal{L}_1} + \hat{\alpha}_1 \hat{\varepsilon}^{\hat{\alpha}_0,\hat{\alpha}_1}_{\mathcal{L}_0,\, \mathcal{L}_1 \cup \{\mathbf{x}_j^U\}}\,, \tag{5.24}$$

where $\hat{\alpha}_0 = 1 - \hat{\alpha}_1 = \frac{l_0}{l_0 + l_1}$ and for two arbitrary sets $\mathcal{A}$ and $\mathcal{B}$, $\hat{\varepsilon}^{\alpha_0,\alpha_1}_{\mathcal{A},\mathcal{B}}$ is computed using (5.17). Note that $\hat{\alpha}_0$ is a consistent estimator of α_0 under the assumption of random sampling (Braga-Neto et al., 2014).

The use of $\hat{\varepsilon}^{\alpha_0,\alpha_1}_{i,\mathcal{L}_0,\mathcal{L}_1}$ presented in (5.18) for computation of the class-specific error rate estimates $\hat{\varepsilon}^{\hat{\alpha}_0,\hat{\alpha}_1}_{\mathcal{L}_0 \cup \{\mathbf{x}_j^U\},\, \mathcal{L}_1}$ and $\hat{\varepsilon}^{\hat{\alpha}_0,\hat{\alpha}_1}_{\mathcal{L}_0,\, \mathcal{L}_1 \cup \{\mathbf{x}_j^U\}}$ that are used in (5.24) already allows more efficient computation compared to resampling-based rules due to its closed-form expression. Nevertheless, to further reduce the computational requirements, the following lemma provides the derivation of recursive update rules for the core statistics that are required in (5.18) through (5.19)-(5.23). Hence, we avoid the recalculating these statistics from scratch for every newly presented unlabeled data point $\mathbf{x}_j^U$.

Lemma 5.4.1. *Suppose that the sample means $\bar{\mathbf{x}}_{\mathcal{L}_0}$ and $\bar{\mathbf{x}}_{\mathcal{L}_1}$, the pooled sample covariance matrix $\hat{\mathbf{\Sigma}}_{\mathcal{L}_0,\mathcal{L}_1}$, and*

$$\mathbf{K} = (\mathbf{I}_p + \kappa\gamma\hat{\mathbf{\Sigma}}_{\mathcal{L}_0,\mathcal{L}_1})^{-1}\,, \quad \kappa = \frac{l_0 + l_1 - 2}{l_0 + l_1 - 1}\,, \tag{5.25}$$

are available and computed using $\mathcal{L}_0$ and $\mathcal{L}_1$. To compute $\hat{\varepsilon}^{\hat{\alpha}_0,\hat{\alpha}_1}_{\mathcal{L}_0 \cup \{\mathbf{x}_j^U\},\, \mathcal{L}_1}$ for every $\mathbf{x}_j^U \in \mathcal{U}$ that is required in (5.24), we can update all necessary statistics in (5.18)-(5.23) using the following updates rules:

$$\bar{\mathbf{x}}_{\mathcal{L}_0 \cup \{\mathbf{x}_j^U\}} = \bar{\mathbf{x}}_{\mathcal{L}_0} + \frac{\mathbf{d}_{l_0+1}}{l_0 + 1}, \tag{5.26}$$

$$\hat{\mathbf{\Sigma}}_{\mathcal{L}_0 \cup \{\mathbf{x}_j^U\},\mathcal{L}_1} = \kappa\hat{\mathbf{\Sigma}}_{\mathcal{L}_0,\mathcal{L}_1} + \beta_{l_0}\,\mathbf{D}_{l_0+1}, \tag{5.27}$$

$$\mathbf{H}_{\mathcal{L}_0 \cup \{\mathbf{x}_j^U\},\mathcal{L}_1} = \mathbf{K} \times \Big(\mathbf{I}_p - \frac{\gamma\beta_{l_0}\,\mathbf{D}_{l_0+1}\mathbf{K}}{1 + \gamma\beta_{l_0}\,\mathbf{d}_{l_0+1}^T\mathbf{K}\,\mathbf{d}_{l_0+1}}\Big), \tag{5.28}$$

$$\beta_{l_i} = \frac{l_i}{(l_0 + l_1 - 1)(l_i + 1)}, \tag{5.29}$$

$$\mathbf{d}_{l_i+1} = \mathbf{x}_j^U - \bar{\mathbf{x}}_{\mathcal{L}_i}, \tag{5.30}$$

$$\mathbf{D}_{l_i+1} = \mathbf{d}_{l_i+1}\mathbf{d}_{l_i+1}^T, \tag{5.31}$$

where $\bar{\mathbf{x}}_{\mathcal{L}_1 \cup \{\mathbf{x}_j^U\}}$, $\hat{\mathbf{\Sigma}}_{\mathcal{L}_0,\mathcal{L}_1 \cup \{\mathbf{x}_j^U\}}$, and $\mathbf{H}_{\mathcal{L}_0,\mathcal{L}_1 \cup \{\mathbf{x}_j^U\}}$, which are needed to compute $\hat{\varepsilon}^{\hat{\alpha}_0,\hat{\alpha}_1}_{\mathcal{L}_0,\, \mathcal{L}_1 \cup \{\mathbf{x}_j^U\}}$, can be obtained by interchanging the roles of l_0 and l_1 in (5.26), (5.27), and (5.28), respectively.

Proof. See Appendix G. ■

Algorithm 7 outlines the step-by-step implementation of the proposed method.

Algorithm 7 AL-RLDA

Input:
a labeled set $\mathcal{L}_k = \{\mathbf{x}_i\}_{i=1}^{l_k}$ from class $k = 0, 1$
an unlabeled set $\mathcal{U} = \{\mathbf{x}_j^U\}_{j=1}^{u}$ of size u
Parameters:
an estimate of prior probabilities $\hat{\alpha}_0$ and $\hat{\alpha}_1$
a regularization parameter γ
an annotation budget size $\tilde{b}$
Return:
a set of selected unlabeled data points $\hat{\mathcal{U}}$

Calculate sample means $\bar{\mathbf{x}}_{\mathcal{L}_0}$ and $\bar{\mathbf{x}}_{\mathcal{L}_1}$ from (5.11)
Calculate pooled sample covariance matrix $\hat{\mathbf{\Sigma}}_{\mathcal{L}_0,\mathcal{L}_1}$ from (5.13)
Calculate $\mathbf{K}$ from (5.25)
for $j = 1, \ldots, |\mathcal{U}|$ **do**
  for $k = 0, 1$ **do**
    $\mathcal{S}_0 \leftarrow \mathcal{L}_0$, $\mathcal{S}_1 \leftarrow \mathcal{L}_1$, $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{\mathbf{x}_j^U\}$
    Update $\bar{\mathbf{x}}_{\mathcal{S}_k}$ using (5.26)
    Update $\hat{\mathbf{\Sigma}}_{\mathcal{S}_0,\mathcal{S}_1}$ using (5.27)
    Update $\mathbf{H}_{\mathcal{S}_0,\mathcal{S}_1}$ using (5.28)
    Calculate $\hat{\varepsilon}^{\hat{\alpha}_0,\hat{\alpha}_1}_{\mathcal{S}_0,\mathcal{S}_1}$ using (5.17)
  end for
  Calculate the EER-P objective function in (5.24):
  $\epsilon_j \triangleq \hat{\alpha}_0 \hat{\varepsilon}^{\hat{\alpha}_0,\hat{\alpha}_1}_{\mathcal{L}_0 \cup \{\mathbf{x}_j^U\},\mathcal{L}_1} + \hat{\alpha}_1 \hat{\varepsilon}^{\hat{\alpha}_0,\hat{\alpha}_1}_{\mathcal{L}_0,\mathcal{L}_1 \cup \{\mathbf{x}_j^U\}}$
end for
Sort ϵ_j and take indices of the $\tilde{b}$ least ϵ_j
$j_1, \ldots, j_{\tilde{b}} \leftarrow \epsilon_{j_1} \leq \epsilon_{j_2} \leq \ldots \leq \epsilon_{j_u}$
$\hat{\mathcal{U}} = \{\mathbf{x}_j^U \mid j \in j_1, \ldots, j_{\tilde{b}}\}$
return $\hat{\mathcal{U}}$

5.5 Experimental settings

5.5.1 Baseline and benchmark AL methods

To assess the AL-RLDA, we include results of using the following AL baseline methods: (i) random sampling, which selects unlabeled data points in a uniform manner from the available pool; (ii) uncertainty sampling, which selects data points with the lowest scores assigned by a classifier; and (iii) diversity sampling, which selects data points based on the lowest and highest scores produced by a classifier. We also obtain the results of the state-of-the-art methods: (i) Probability Coverage (ProbCover) (Yehuda et al., 2022); and (ii) Bayesian Active Learning by Disagreement (BALD) (Houlsby et al., 2011) algorithms. Specifically, ProbCover is a graph-based algorithm that selects data points based on maximizing coverage of unlabeled data within a predefined radius from any labeled data point. In addition, since ProbCover properly works only with given good embeddings, we construct the embedding using Self-Supervised Contrastive

Table 5.1: Summary of binary classification datasets.

Dataset	No. of features	No. of samples	l_0/l_1
CPU activity (`cpu`)	21	8,192	2.3
KDDCup upselling (`ups`)	45	5,032	1.0
Home equity line of credit (`heloc`)	22	10,000	1.0
Higgs bosons (`higgs`)	24	2,000	1.0
Compass analysis (`compass`)	17	2,000	1.0
KDDCup IPUMS census (`ipums`)	20	2,000	1.0
Bank customers (`bank`)	32	8,192	2.2
Airlines delay (`airlines`)	7	2,000	1.2
Eye movements (`emv`)	20	7,608	1.0
AutoML challenge jannis (`jannis`)	54	2,000	1.0

Learning using Random Feature Corruption (SCARF) (Bahri et al., 2022). On the other hand, BALD selects data points that maximize the mutual information between model posterior and predictions.

5.5.2 Datasets

To perform experiments, we collected 10 datasets from the OpenML dataset repository. In Table 5.1, the summary of binary datasets is presented. In addition, Table 5.1 includes the class imbalance ratio information, l_0/l_1, where l_0 is the number of samples from majority class, l_1 is the number of samples from minority class.

5.5.3 Supervised Learning Algorithms

As discussed before, AL-RLDA is an efficient screening method for identifying unlabeled data points for annotation. However, the final classifier trained with all combined labeled data (labeled and queried with AL) is not required to be the RLDA. Therefore, we compare all considered AL methods by obtaining the average test-set accuracy of five classifiers: particularly, three linear classifiers, namely, RLDA, Logistic Regression (LRR) with L_2 regularization (Anderson and Richardson, 1979), and Linear Support Vector Machine (LSVM) (Vapnik, 1995), and two non-linear classifiers such as one-hidden-layer perceptron (MLP) (Ivakhnenko, 1971) with a fixed $\frac{p}{2}$ neurons, and Random Forest (RF) (Breiman, 2001). The hyperparameter space of each classifier is shown in Table 5.2. The remaining hyperparameters were left at their default values provided by `scikit-learn` library (as of version 1.3.0) (Pedregosa et al., 2011) or fixed *a priori* (e.g., the number of neurons in the MLP).

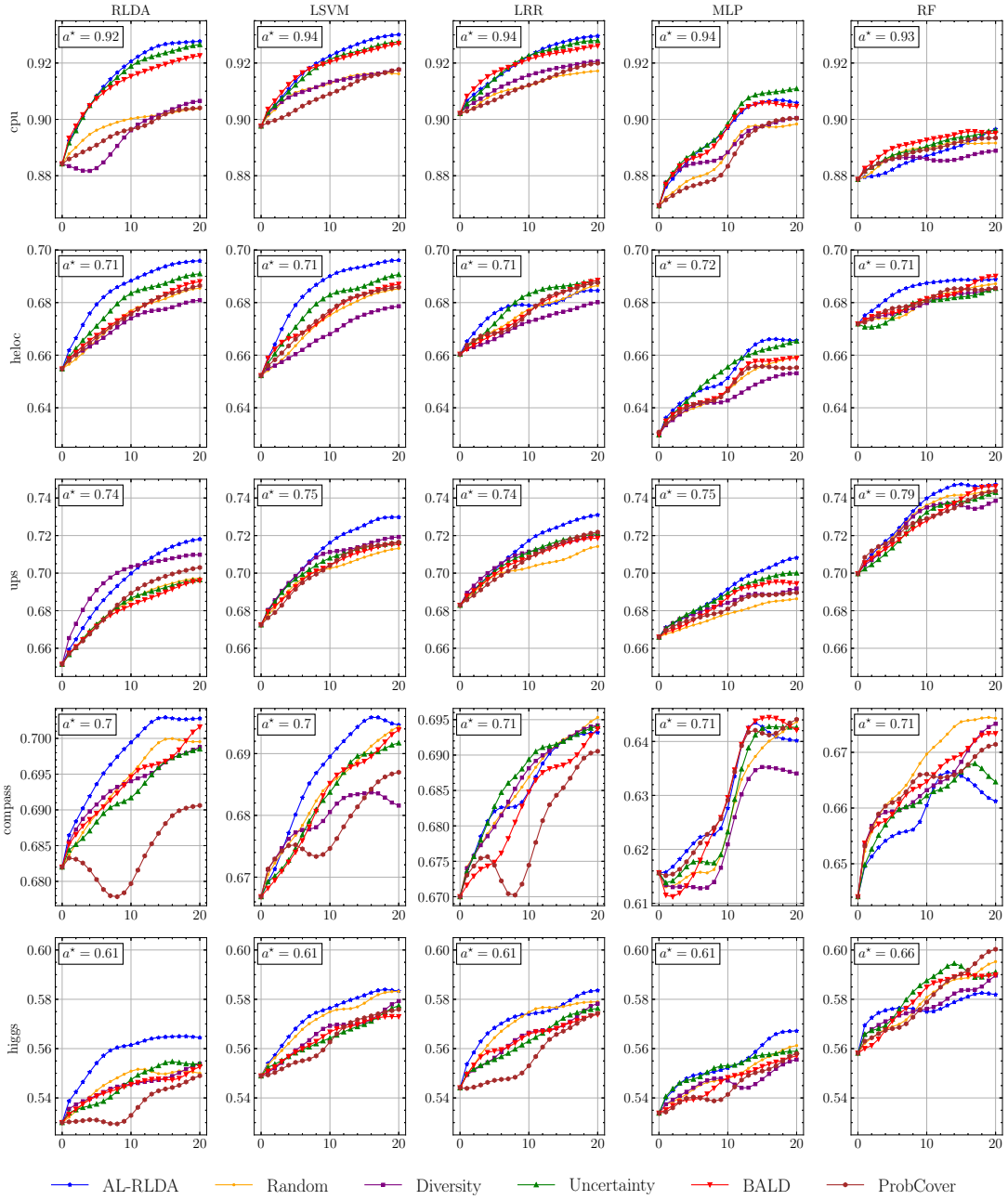

Figure 5.1: The average performance in terms of test-set accuracy results of all considered AL methods across 20 AL cycles for `cpu`, `heloc`, `ups`, `compass`, and `higgs` datasets.

5.5.4 Evaluation

To perform experiments and evaluate the proposed method, we randomly divided each dataset into training and test sets using a splitting ratio of $70:30$. The training subset was further split into labeled and unlabeled sets, where labeled set $\mathcal{L}$ comprised 100 randomly selected data points, and the rest of the data points constructed unlabeled set $\mathcal{U}$. All splits were performed in a stratified manner.

To simulate active learning settings, we conduct experiments with 20 AL cycles. In particular, given a labeling budget size of $\tilde{b}$ data points, in each AL cycle we: (i) run an

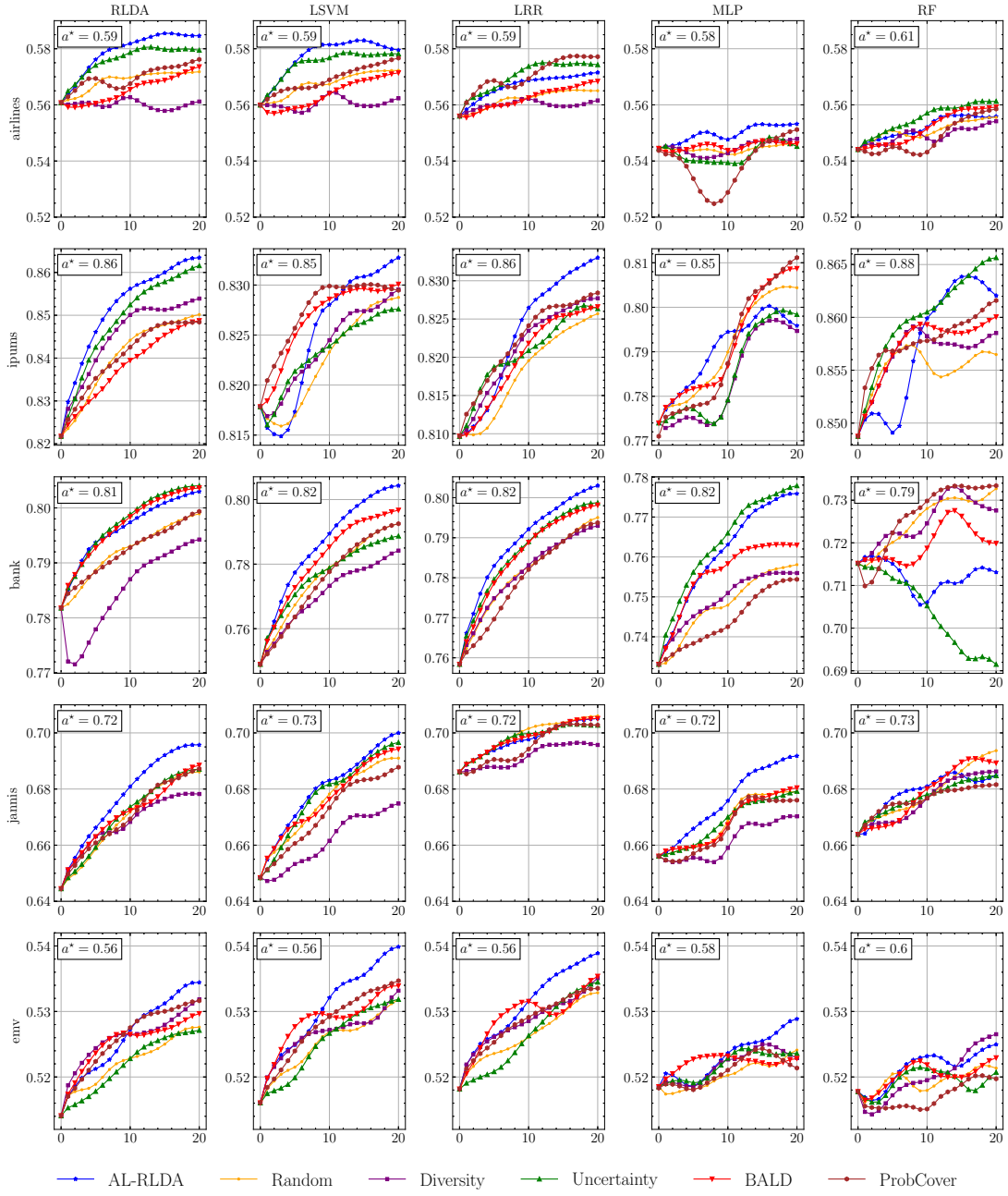

Figure 5.2: The average performance in terms of test-set accuracy results of all considered AL methods across 20 AL cycles for `airlines`, `ipums`, `bank`, `jannis`, and `emv` datasets.

AL algorithm to query the $\tilde{b}$ "most informative" data points from $\mathcal{U}$; (ii) obtain labels for selected data points; (iii) update the labeled set by adding the selected data points along with their acquired labels; (iv) use the updated labeled set to train the classifier; and (v) evaluate the classifier using the test set. As a result, we have 20 AL cycles and 10 datasets, thus yielding 200 cases for comparison (10 datasets $\times$ 20 cycles).

We repeat the experiments twenty times using different labeled, unlabeled, and test sets.

Table 5.2: Summary of hyperparameter spaces.

Model	Parameter	Search space
LSVM	regularization parameter	{ 10, 5, 1, 0.5, 0.1}
	penalty	{ L_1, L_2 }
	loss	{ squared hinge, hinge }
LRR	regularization parameter	{ 100, 10, 1, 0.1, 0.01 }
MLP	solver	{ SGD, Adam, L-BFGS, }
	activation	{ relu, logistic sigmoid }
RF	number of estimators	{ 1, 5, 10, 20 }
	maximum depth	{ 2, 5, 10 }
	maximum features	{ square root, $\log_2$ }
RLDA	γ	{ 100, 10, 1, 0.1 }

5.6 Results

In Figures 5.1 and 5.2 the average (over twenty repetitions) test-set accuracy of each classifier (RLDA, LSVM, LRR, MLP, and RF) across 20 AL cycles with budget size of $\tilde{b} = 10$ are shown. In each AL cycle, the performance results of all considered AL algorithms are shown. In each plot, AL cycle 0 shows the average accuracy on the test set of the classifier trained using merely initial labeled set. In each plot, AL-RLDA, uncertainty sampling, diversity sampling, random sampling, BALD, and ProbCover are depicted in the blue, purple, green, orange, red, and brown curves, respectively. In addition, we also show the average accuracy on the test set of the classifier trained utilizing all data points and true labels of labeled and unlabeled sets, denoted as $a^\star$. For the ease of visualization, errors bars are not presented in Figures 5.1 and 5.2 (see Appendix H for figures with error bars).

As can be seen from Figures 5.1 and 5.2, the results demonstrate that in a number of cases AL-RLDA uniformly performs better than all other AL algorithms. Particularly, AL-RLDA shows better results than *all* other AL methods in 156, 162, 97, 106, and 66 cases for RLDA, LSVM, LRR, MLP, and RF classifiers, respectively. These results confirm the efficacy of the proposed algorithm. Nevertheless, the results also reveal that the data points queried using AL-RLDA fit more effectively with linear classifiers compared to highly non-linear classifier such as RF. As an example, AL-RLDA outperformed all other AL algorithms in 66 cases when using RF, while with LSVM, AL-RLDA showed the highest performance in 162 cases. That being said, with RF, none of the other algorithms achieved better performance in more than 66 out of 200 cases. Particularly, the runner-up algorithm, which corresponds to uncertainty sampling, outperformed other methods in only 44 out of 200 cases.

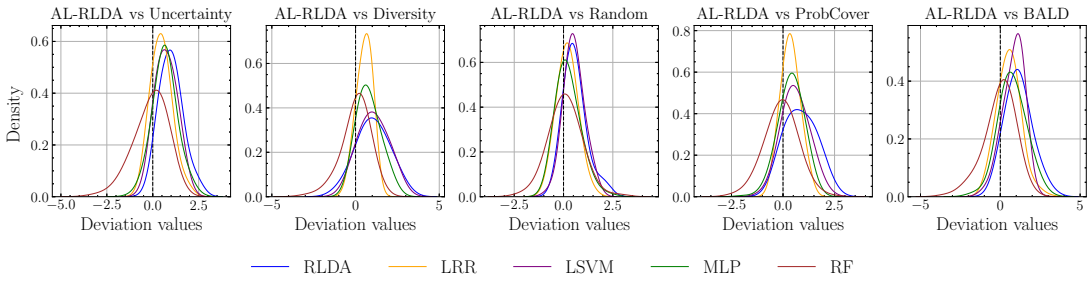

Figure 5.3: Empirical distributions illustrating the performance deviations of all classifiers trained on data points selected by AL-RLDA or other AL algorithms.

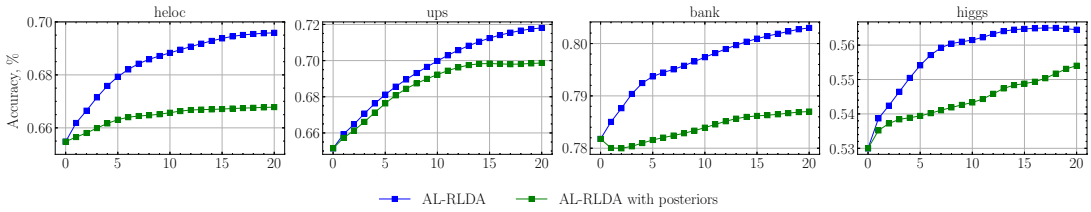

Figure 5.4: The average performance in terms of test-set accuracy results across 20 AL cycles for `heloc`, `ups`, `bank`, and `higgs` datasets produced by AL-RLDA and "AL-RLDA with posteriors".

5.7 Discussion

AL-RLDA classification performance. To provide a summary of the experimental results, we compute the distribution of average test-set accuracy differences (i.e., empirical deviation distribution) between each classifier trained on data points selected by AL-RDLA and other AL algorithms. In particular, Figure 5.3 shows the empirical deviation distributions for all considered classifiers. In each plot, a positive shift in these distributions suggests that AL-RLDA is a superior algorithm compared with another AL algorithm for the same classifier. Figure 5.3 reveals that the majority mass of deviation distributions are on the positive side for all classifiers.

EER with prior probabilities. As outlined in Section 5.1, we proposed the objective function with class prior probabilities for EER, in contrast to using a common objective function (5.3) with posterior probabilities. Naturally, if good estimates of posteriors were available, using them in (5.3) would have been advantageous. However, having good estimates of posteriors generally leads to the use of AL futile in the first place. On the other hand, using the EER-P objective function allows to balance between the existing belief of the learner, which in this case is captured by the error rate of the RLDA classifier, and the nature of the problem, which is captured by the class prior probabilities. In order to investigate the effect of using prior probabilities instead of posterior estimates, we conducted addition numerical experiments with the same AL-RLDA algorithm except with the use of posteriors to weight error estimates, to which we refer as "AL-RLDA with posteriors." The experimental setup was similar

to Section 5.5. Figure 5.4 demonstrates the average test-set accuracy for AL-RLDA and AL-RLDA with posteriors across four datasets. The results demonstrate that AL-RLDA with EER-P shows better performance in comparison with AL-RLDA with EER objective function. However, it is worthwhile to study the proposed EER-P and the common EER objective functions for other estimated error reduction approaches.

Impact of class imbalance. To examine the performance of AL-RLDA under class imbalance, we performed an additional experiment with simulated imbalanced datasets. Particularly, we used the largest available `heloc` dataset and obtained class imbalance ratios of 1, 2, 4, and 8 by randomly downsampling one class. The experimental setup was similar to Section 5.5, except we show the balanced accuracy results, as this metric is more appropriate for evaluating performance on imbalanced datasets (Brodersen et al., 2010). Figure 5.5 shows the average test-set balanced accuracy of RLDA achieved under four class imbalance scenarios. Also, in the plots, $\hat{a}^\star$ displays the average balanced accuracy on test set of the classifier trained utilizing data points and true labels of labeled and unlabeled sets. The results demonstrate that AL-RLDA shows better results than all other AL methods for all considered cases except $l_0/l_1 = 8$, where in the first three cycles, ProbCover demonstrated better results.

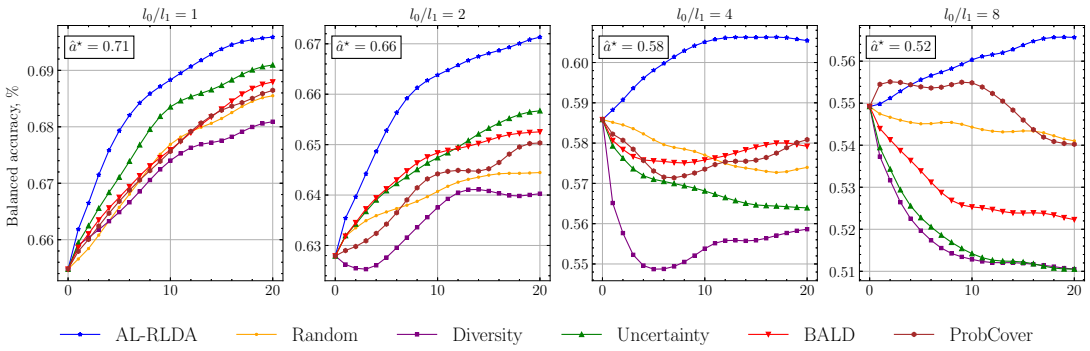

Figure 5.5: The average classification results in terms of test-set balanced accuracy using RLDA across 20 AL cycles for `heloc` dataset.

Less annotation budget. An alternative approach to assess the performance of a specific AL algorithm is to compare how much annotation budget was required to achieve the same level of performance as other AL methods. For instance, we can see from Figures 5.1 and 5.2 that for both `jannis` and `heloc` datasets, the AL-RLDA reached ∼68% test-set accuracy after ten cycles of AL (corresponds to a total of 100 obtained labels), respectively. At the same time, ProbCover and uncertainty sampling methods needed four and five more AL cycles (corresponds to 150 and 140 total labels) to achieve the same level of performance, respectively. For `higgs` dataset, AL-RLDA reached ∼55.5% test-set accuracy in the fifth cycle of AL (50 total labels), whereas the

Table 5.3: Summary of real-world case datasets.

Dataset	No. of features	No. of samples	l_0/l_1
DNA	180	1,532	1.0
SAVEE	128	120	1.0
EMODB	128	131	1.1

runner-up algorithm, uncertainty sampling, required 10 cycles of AL (150 total labels) to yield the same accuracy.

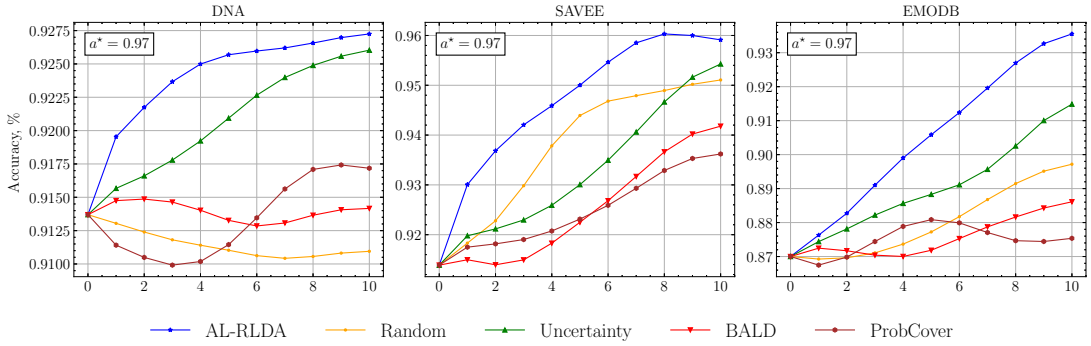

Figure 5.6: The average classification results in terms of test-set accuracy using RLDA across 10 AL cycles.

Real-world cases. To evaluate the practical applicability of the proposed AL-RLDA method, we conducted additional experiments on diverse real-world datasets from the domains of genomics and speech emotion recognition. For the genomics task, we used the DNA (Noordewier et al., 1990) dataset, which involves classifying nucleotide sequences to detect biologically meaningful patterns. Specifically, the task focuses on distinguishing exon/intron (EI) and intron/exon (IE) boundaries. Each DNA sequence consists of nucleotides (A, C, G, T), which were encoded using three binary indicator vectors per nucleotide. In the speech domain, we evaluated AL-RLDA on the SAVEE (Haq and Jackson, 2010) dataset using the happy and sad emotion classes, and on the EMODB (Burkhardt et al., 2005) dataset using anxiety and sad classes. For each audio sample, we computed the log Mel-spectrogram (Mukhamediya et al., 2023) and then averaged across time frames to obtain a fixed-length 128-dimensional feature vector. The summary of datasets is shown in Table 5.3. These tasks reflect real-world scenarios where labeling is time-consuming and often requires domain experts.

The experimental setup was similar to Section 5.5, except we repeated experiments ten times using different splits of labeled, unlabeled, and test sets. In addition, for each dataset, EMODB, SAVEE, and DNA, the labeled set was initialized with 10, 20, and 50 randomly selected samples, respectively. Figure 5.6 shows the average test-set accuracy results of RLDA for three datasets with budget size of $\tilde{b} = 1$ (see Appendix I for figure

with error bars). The results demonstrate that AL-RLDA uniformly outperforms all other methods across all AL cycles.

Limitations. We note the following limitations of our method. As mentioned in Section 5.1, in its current form, AL-RLDA is designed for binary classification tasks. This limitation arises from the use of a closed-form error estimator of RLDA, which is originally developed for use in binary classification settings. To extend AL-RLDA for multi-class classification tasks the following approaches can be considered. The first approach is to undertake the development of a counterpart to Theorem 5.3.1. Then, the developed closed-form error estimator of a multi-class RLDA classifier can be used to extend the AL-RLDA method to support multi-class classification. Another approach is to use the existing error estimator of binary RLDA classifier and modify the objective function (5.6) for one-versus-one binary tasks. We leave these research directions to future studies.

Another limitation corresponds to the estimation of class prior probabilities. As discussed in Section 5.4, we estimate the class prior probabilities using $\hat{\alpha}_i = \frac{l_i}{l_0+l_1}, i = 0, 1$. We note that this is a legitimate estimator of prior probabilities only under the assumption of random sampling (Braga-Neto et al., 2014). For separate sampling cases, where samples are drawn independently from each class, we cannot use the class proportions in the labeled data to estimate α_i (Braga-Neto et al., 2014; Shahrokh Esfahani and Dougherty, 2013). In this case, we can obtain reliable estimates of class prior probabilities by collecting population statistics separately from the labeled data.

AL-RLDA as a sampling method. As discussed in Section 5.2, once the AL-RLDA selected samples to label, the updated labeled set can be used to train a different classifier besides RLDA. The results in Figures 5.1 and 5.2 reveal that the AL-RLDA can outperform other AL methods when used with linear classifiers. However, when employing a non-linear classifier, the performance of AL-RLDA can deteriorate. This is not surprising since the AL-RLDA attempts to minimize the error of RLDA, which is a linear classifier. Thus, the EER-based optimal selection strategy should be classifier rule-dependent. Nonetheless, future work may focus on deriving the closed-form expressions for various classifiers.

Scalability to ultra-high-dimensional settings. AL-RLDA is inherently suitable for ultra-high-dimensional settings due to its closed-form error estimator and the use of RLDA. In comparison with traditional EER-based approaches which require retraining the model for each unlabeled candidate, AL-RLDA computes the estimated error recursively without retraining. Moreover, RLDA can effectively handle the small-sample and high-dimensional regime by regularizing the within-class scatter matrix,

which prevents numerical instability when the number of features exceeds the number of labeled instances. These characteristics make AL-RLDA scalable and efficient, particularly in domains with thousands of features, such as genomics or medical signal analysis.

Time complexity. Since the most time-consuming operation in RLDA is the matrix inversion, we derive the time complexity of AL-RLDA in terms of matrix inversion. As can be seen from Algorithm 7, AL-RLDA requires a calculation of matrix inversion once, which takes $\mathcal{O}(p^3)$. An alternative and naive approach to estimating the consistent error is to use cross-validation or bootstrapping estimation rules. For comparison, the time complexity of K-Fold CV is $\mathcal{O}(Kup^3)$ to compute matrix inversions, where u is the number of unlabeled samples.

5.8 Summary

In this chapter, we proposed an efficient estimated error reduction-based active learning method for binary classification tasks, named AL-RLDA. At its core, AL-RLDA employs an accurate closed-form error estimator of RLDA. That is to say, AL-RLDA identifies unlabeled data points with the least weighted average error estimate of the RLDA classifier. At the same time, AL-RLDA uses a new objective function that is based on class prior probabilities in contrast to the conventional objective function with posterior probabilities. The efficiency of AL-RLDA is achieved by replacing the retraining-evaluation procedure with the re-evaluation stage, which can be performed recursively for each new candidate data point. The experimental results showed that AL-RLDA can achieve better performance compared to baseline and state-of-the-art active learning methods.

BLANK

Chapter 6

Conclusion

In this thesis, two main directions have been explored that aim to efficiently leverage a large number of unlabeled data points for a small labeled sample classification task.

6.1 Semi-supervised learning

The first direction focused on semi-supervised learning (SSL) in which unlabeled data can be employed with available labeled data to potentially improve the performance of the classifier. In particular, we conducted a study of wrapper SSL methods and introduced two novel SSL algorithms.

In Chapter 2, we introduced a new form of self-training algorithm, named SRPM-ST, which sequentially retrains and pseudo-labels mini-batches of unlabeled data. Experimental results show that SRPM-ST can, on average, outperform a pervasive form of self-training. In particular, the results suggest that there exists the optimal mini-batch size that can lead to the lowest error rate. In practice, this mini-batch size can be considered a hyperparameter, therefore we also studied the applicability of a cross-validation-based search strategy adapted for self-training to estimate the optimal mini-batch size. The results demonstrate that cross-validation can be used as a reliable mini-batch size estimator. Another interesting point is attributed to the observation that improvement is shown even though unlabeled data in SRPM is divided into mini-batches randomly and used in an arbitrary order.

A key limiation of self-training methods is that they are prone to noise accumulation. Since the quality of generated pseudo-labels depends on the classifier trained on the previously pseudo-labeled and labeled data, incorrectly assigned pseudo-labels may be reinforced in subsequent iterations, thus leading to performance degradation. Therefore, an additional mechanism is required to avoid reinforcement of incorrectly pseudo-labeled data in the training process. Nonetheless, the results indicate that the generation of pseudo-labeled unlabeled data in mini-batches can be more resilient to noise accumulation compared to iterative self-training.

In Chapter 3, we introduced a novel boosting algorithm for SSL based on *semi-supervised smoothness* and *cluster assumptions*. In particular, the selection and utility of unlabeled data is performed in the form of "batches". That is, BSSL++ iteratively selects a portion of unlabeled data based on the guidance of smoothness assumption and generates pseudo-labels relying on cluster assumption. Experimental results show that BSSL++ can outperform state-of-the-art semi-supervised boosting algorithms. Particularly, the results also show that the clustering-based procedure used in BSSL++ reduces dependence on the classifier and thus guards against noise accumulation. In addition, we observe that the generalization performance of BSSL++ improves by learning from a newly selected portion of unlabeled data at each iteration (this is also seen from experimental results on synthetic data).

In Chapter 4, we introduced a novel adaptive boosting algorithm with manifold regularization based on label similarity and dissimilarity between data points, namely AdaBoost.SDM. In particular, AdaBoost.SDM leverages the mixed-graph Laplacian to use the information about the underlying structure of the data. Additionally, to encode the *manifold assumption* into BSSL++, we replace AdaBoost-like ensemble used in BSSL++ with AdaBoost.SDM (we referred to this algorithm as BSSL++.SDM). As a result, BSSL++.SDM encodes the manifold assumption alongside already presented semi-supervised smoothness and cluster assumptions. Experimental results show that BSSL++.SDM can improve the performance of BSSL++.

6.2 Active learning

The second direction focused on active learning (AL) to construct a labeled dataset from a pool of unlabeled data points by selecting the most beneficial ones in terms of leading to better predictive performance. In particular, we conducted a study of estimated error reduction-based AL methods and introduced an efficient AL algorithm.

In Chapter 5, we introduced an efficient AL algorithm that reduces an accurate closed-form error estimation of RLDA, named AL-RLDA. In particular, the efficiency of AL-RLDA is attributed to the factor that, at its core, it uses the proposed objective function based on class prior probabilities (EER-P). Specifically, EER-P allows to use re-evaluation procedure that can be performed recursively instead of conventional repetitive retraining-evaluation. Experimental results show that AL-RLDA can outperform state-of-the-art AL algorithms. It is essential to note that the results also show that AL-RLDA requires less annotation budget to achieve the same performance level as other AL algorithms. Another interesting point is that AL-RLDA can be used as solely a sampling method. In particular, data points that were selected using AL-RLDA can be used to train a different classifier besides RLDA. The results also indicate that AL-RLDA can be successfully used with linear classifiers.

Table 6.1: Summary of proposed methods and key results.

Method	Key idea	Key results
SRPM-ST	Sequential mini-batch pseudo-labeling	Up to 33% improvement in accuracy vs. Full-batch ST
BSSL++	Learn++-inspired semi-supervised boosting using SSL smoothness and cluster assumptions	Adaptively learns from selected and pseudo-labeled data
AdaBoost.SDM BSSL++.SDM	Manifold regularized adaptive boosting	Encodes manifold assumption effectively
AL-RLDA	EER-P objective function Recursive estimation of error	Faster than EER and label-efficient

Table 6.1 summarizes the core contributions and outcomes of the methods proposed in this thesis. Each method addresses a key challenge in learning with limited labeled data, specifically, by improving pseudo-labeling (SRPM-ST), enhancing ensemble strategies (BSSL++), incorporating structure-aware classification (AdaBoost.SDM and BSSL++.SDM), and enabling efficient data labeling (AL-RLDA). These methods demonstrate significant improvements in both performance and efficiency, supporting effective learning in real-world settings with limited annotations.

6.3 Societal impact

SSL and AL methods have significant societal implications, particularly in domains where data annotation is expensive, time-consuming, or requires specialized expertise. For instance, in healthcare, labeling medical images or clinical data often demands highly trained professionals such as radiologists or pathologists. In this context, SSL reduces the dependence on large annotated datasets by leveraging abundant unlabeled data, while AL focuses on selecting the most informative samples for labeling, thereby minimizing expert effort. These methods can substantially reduce annotation costs and accelerate model development.

6.4 Future works

Future work may consider the following directions:

- SRPM-ST can be enhanced with a method to identify the most successful or even optimal order of mini-batches of unlabeled data that is used to retrain the classifier sequentially;
- SRPM-ST can be potentially used with a subset selection strategy, particularly, a subset of pseudo-labeled unlabeled data can be selected utilizing either soft or hard thresholds;

- BSSL++ can be extended for incremental learning strategy, in which the ensemble classifier is trained only on newly arrived batches of unlabeled or labeled data without having access to the original training data; and
- AdaBoost.SDM (and BSSL++.SDM) and AL-RLDA can be extended to multi-class classification settings.

Another promising direction for future research lies in applying the proposed semi-supervised and active learning frameworks to real-world domains with limited labeled data. For example, in healthcare, our methods could support diagnostic systems that learn effectively from a few expert-annotated samples while leveraging large volumes of unlabeled medical images. Similarly, in genomics, where labeling often involves costly laboratory experiments, the ability to prioritize the most informative genetic sequences for annotation could significantly reduce experimental burden. As a result, integrating the proposed AL and SSL techniques in such applications enables greater data efficiency and broadens the accessibility of machine learning solutions.

Appendices

BLANK

Appendix A

Dataset Repository and Source Code

A.1 Dataset Repository

The links to the dataset repositories are as follows:

- https://openml.org/
- https://archive.ics.uci.edu/

A.2 Source Code

All source codes are available at:

- https://github.com/Azamat-Mukhamediya/source-codes

BLANK

Appendix B

Derivation of (2.14) and (2.15)

$$
\begin{aligned}
&p_1^+ = P\Big(x > \hat{\theta}_1^{\text{SRPM}} \mid x \in \pi^+\Big) = \\
&P\Big(x > \frac{1}{2}(1-\alpha_0^- + \alpha_0^+)\mu^+ + \frac{1}{2}(1-\alpha_0^+ + \alpha_0^-)\mu^- \mid x \in \pi^+\Big) = \\
&P\Big(z > \frac{\frac{1}{2}(1-\alpha_0^- + \alpha_0^+)\mu^+ + \frac{1}{2}(1-\alpha_0^+ + \alpha_0^-)\mu^- - \mu^+}{\sigma}\Big) = \\
&P\Big(z > \frac{\frac{1}{2}(1-\alpha_0^- + \alpha_0^+)\mu^+ + \frac{1}{2}(1-\alpha_0^+ + \alpha_0^-)\mu^- - \mu^+}{\sigma}\Big) = \\
&P\Big(z > -\frac{\mu^+ - \mu^-}{2\sigma}(1-\alpha_0^+ + \alpha_0^-)\Big) = \\
&\Phi\Big(\frac{\delta}{2}(1-\alpha_0^+ + \alpha_0^-)\Big)
\end{aligned}
\tag{B.1}
$$

where $\delta = \frac{\mu^+ - \mu^-}{\sigma}$. Same approach can be applied to derive (2.15).

BLANK

Appendix C

K-fold CV

Algorithm 8 ST K-fold CV

Input:
an initial labeled set $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ of size l
an unlabeled set $\mathcal{U} = \{\mathbf{x}_i\}_{i=1}^{u}$ of size u
Parameters:
a classification algorithm Ψ
a pre-specified set of I candidate mini-batch sizes, denoted M_i, $i = 1, \ldots, I$, to evaluate
Return:
estimated optimal mini-batch size M_i

for $i = 1, \ldots, I$ **do**
  Split $\mathcal{L}$ into K folds $\mathcal{L}_k = \{(\mathbf{x}_i, y_i)\}$, $i = (k-1)F + 1, \ldots, kF$, $k = 1, \ldots, K$, where $F = N/K$
  Initialize the estimate of the error rate: $\epsilon \leftarrow 0$
  for $k = 1, \ldots, K$ **do**
    Held-out fold $\mathcal{L}_k$ from $\mathcal{L}$: $\overline{\mathcal{L}} = \mathcal{L} \setminus \mathcal{L}_k$
    Call SRPM-ST algorithm: $\psi_k \leftarrow$ SRPM-ST($\overline{\mathcal{L}}, \mathcal{U}, m_i$)
    $\epsilon_k \leftarrow$ Evaluate classifier ψ_k on $\mathcal{L}_k$
    $\epsilon \leftarrow \epsilon + \epsilon_k$
  end for
  $\epsilon_i \leftarrow \frac{\epsilon}{K}$
end for
$i = \underset{i}{\operatorname{argmin}}\ \epsilon_i$
return M_i

BLANK

Appendix D

Figures 2.2 and 2.3 with error bars

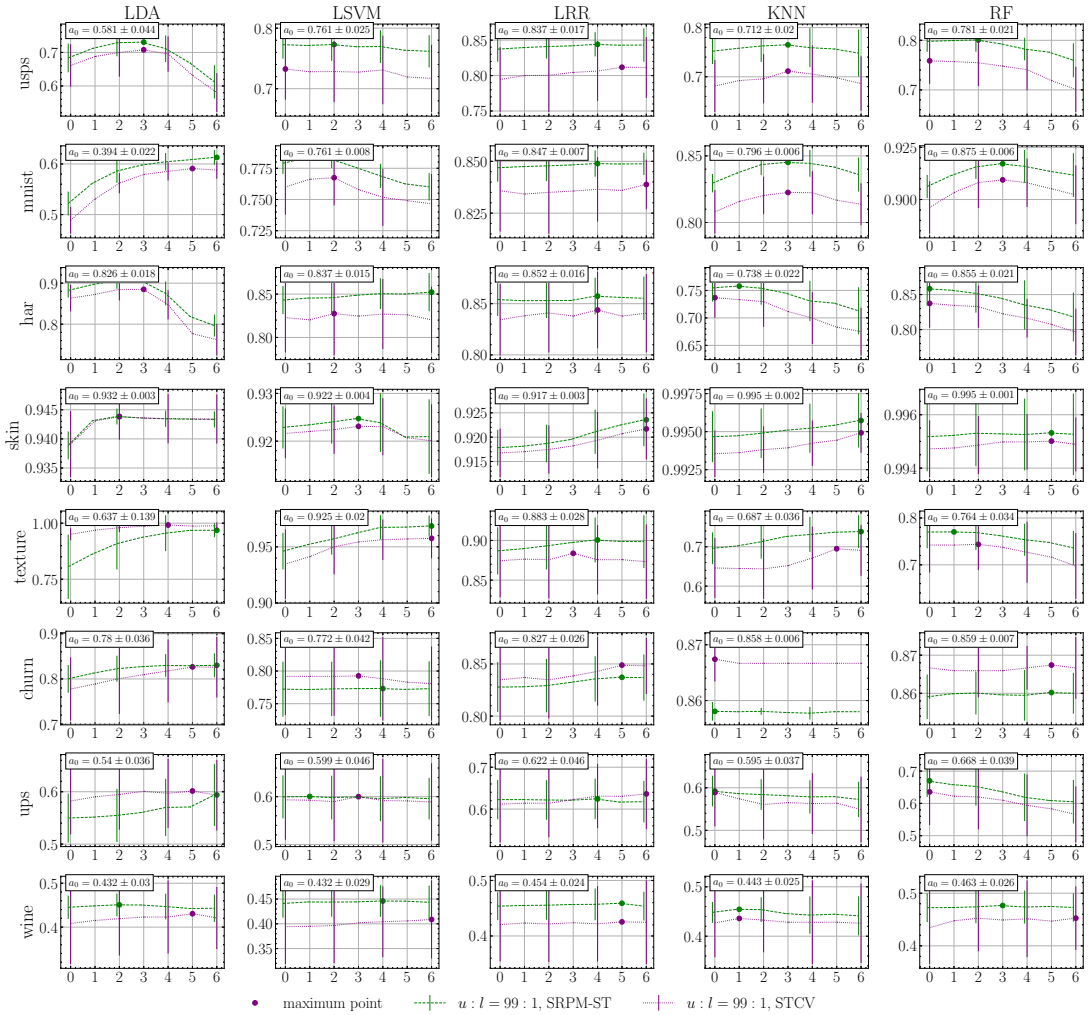

Figure D.1: The average performance in terms of test-set accuracy with error bars across different mini-batch sizes for all datasets and all considered classifiers using $u : l = 99 : 1$.

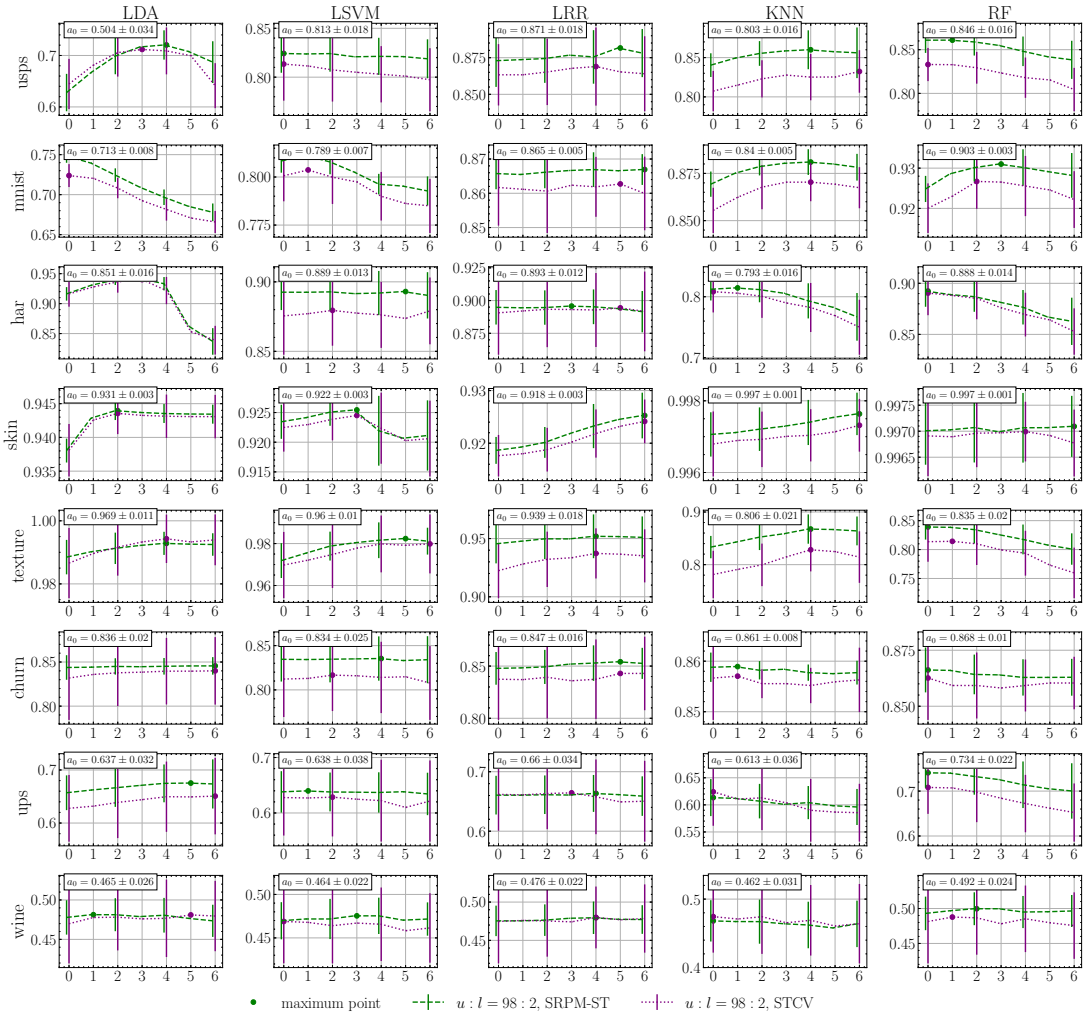

Figure D.2: The average performance in terms of test-set accuracy with error bars across different mini-batch sizes for all datasets and all considered classifiers using $u : l = 98 : 2$.

Appendix E

Proof of (4.16)

To derive the (4.16), we rewrite (4.15) by interchanging the indices i and j in the third and fourth lines of (4.15) as

$$\begin{aligned}\overline{E}_1 =& \sum_{i}^{l} e^{-2y_i H_{t-1}(\mathbf{x}_i)} e^{-2y_i \alpha_t h_t(\mathbf{x}_i)} \\ &+ \frac{\lambda}{4} \sum_{i}^{l} \sum_{j}^{l} \mathbf{W}_{ij} e^{H_{t-1}^{ij}} e^{2\alpha_t h_t(\mathbf{x}_i)} \\ &+ \frac{\lambda}{4} \sum_{j}^{l} \sum_{i}^{l} \mathbf{W}_{ji} e^{H_{t-1}^{ji}} e^{-2\mathbf{S}_{ji}\alpha_t h_t(\mathbf{x}_i)} \\ &+ \frac{\lambda}{4} \sum_{i}^{l} \sum_{j}^{l} \mathbf{W}_{ij} e^{-H_{t-1}^{ij}} e^{-2\alpha_t h_t(\mathbf{x}_i)} \\ &+ \frac{\lambda}{4} \sum_{j}^{l} \sum_{i}^{l} \mathbf{W}_{ji} e^{-H_{t-1}^{ji}} e^{2\mathbf{S}_{ji}\alpha_t h_t(\mathbf{x}_i)} .\end{aligned} \tag{E.1}$$

Then, replacing H_{t-1}^{ij} with $H_{t-1}(\mathbf{x}_i) - \mathbf{S}_{ij} H_{t-1}(\mathbf{x}_j)$ yields

$$\begin{aligned}\overline{E}_1 =& \sum_{i}^{l} e^{-2y_i H_{m-1}(\mathbf{x}_i)} e^{-2y_i \alpha_t h_t(\mathbf{x}_i)} \\ &+ \frac{\lambda}{4} \sum_{i}^{l} \sum_{j}^{l} \mathbf{W}_{ij} e^{H_{t-1}(\mathbf{x}_i) - \mathbf{S}_{ij} H_{t-1}(\mathbf{x}_j)} e^{2\alpha_t h_t(\mathbf{x}_i)} \\ &+ \frac{\lambda}{4} \sum_{j}^{l} \sum_{i}^{l} \mathbf{W}_{ji} e^{H_{t-1}(\mathbf{x}_j) - \mathbf{S}_{ji} H_{t-1}(\mathbf{x}_i)} e^{-2\mathbf{S}_{ji}\alpha_t h_t(\mathbf{x}_i)} \\ &+ \frac{\lambda}{4} \sum_{i}^{l} \sum_{j}^{l} \mathbf{W}_{ij} e^{-H_{t-1}(\mathbf{x}_i) + \mathbf{S}_{ij} H_{t-1}(\mathbf{x}_j)} e^{-2\alpha_t h_t(\mathbf{x}_i)} \\ &+ \frac{\lambda}{4} \sum_{j}^{l} \sum_{i}^{l} \mathbf{W}_{ji} e^{-H_{t-1}(\mathbf{x}_j) + \mathbf{S}_{ji} H_{t-1}(\mathbf{x}_i)} e^{2\mathbf{S}_{ji}\alpha_t h_t(\mathbf{x}_i)} .\end{aligned} \tag{E.2}$$

Setting $\mathbf{S}_{ij} = 1$ for $y_i = y_j$, and $\mathbf{S}_{ij} = -1$ for $y_i = -y_j$ gives

$$
\begin{aligned}
\overline{E}_1 = & \sum_{i}^{l} e^{-2y_i H_{t-1}(\mathbf{x}_i)} e^{-2y_i \alpha_t h_t(\mathbf{x}_i)} \\
& + \frac{\lambda}{4} \sum_{i}^{l} \sum_{j}^{l} \mathbf{W}_{ij} e^{H_{t-1}(\mathbf{x}_i) - H_{t-1}(\mathbf{x}_j)} e^{2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, y_j) \\
& + \frac{\lambda}{4} \sum_{i}^{l} \sum_{j}^{l} \mathbf{W}_{ij} e^{H_{t-1}(\mathbf{x}_i) + H_{t-1}(\mathbf{x}_j)} e^{2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, -y_j) \\
& + \frac{\lambda}{4} \sum_{j}^{l} \sum_{i}^{l} \mathbf{W}_{ji} e^{H_{t-1}(\mathbf{x}_j) - H_{t-1}(\mathbf{x}_i)} e^{-2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, y_j) \\
& + \frac{\lambda}{4} \sum_{j}^{l} \sum_{i}^{l} \mathbf{W}_{ji} e^{H_{t-1}(\mathbf{x}_j) + H_{t-1}(\mathbf{x}_i)} e^{2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, -y_j) \\
& + \frac{\lambda}{4} \sum_{i}^{l} \sum_{j}^{l} \mathbf{W}_{ij} e^{-H_{t-1}(\mathbf{x}_i) + H_{t-1}(\mathbf{x}_j)} e^{-2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, y_j) \\
& + \frac{\lambda}{4} \sum_{i}^{l} \sum_{j}^{l} \mathbf{W}_{ij} e^{-H_{t-1}(\mathbf{x}_i) - H_{t-1}(\mathbf{x}_j)} e^{-2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, -y_j) \\
& + \frac{\lambda}{4} \sum_{j}^{l} \sum_{i}^{l} \mathbf{W}_{ji} e^{-H_{t-1}(\mathbf{x}_j) + H_{t-1}(\mathbf{x}_i)} e^{2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, y_j) \\
& + \frac{\lambda}{4} \sum_{j}^{l} \sum_{i}^{l} \mathbf{W}_{ji} e^{-H_{t-1}(\mathbf{x}_j) - H_{t-1}(\mathbf{x}_i)} e^{-2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, -y_j).
\end{aligned}
\tag{E.3}
$$

Since $\mathbf{W}$ is symmetric, we can sum common terms and split the first line into two summations using $y_i \in \{-1, +1\}$ as

$$
\begin{aligned}
\overline{E}_1 = & \sum_{i}^{l} e^{-2H_{t-1}(\mathbf{x}_i)} e^{-2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, +1) \\
& \sum_{i}^{l} e^{2H_{t-1}(\mathbf{x}_i)} e^{2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, -1) \\
& + \frac{\lambda}{2} \sum_{i}^{l} \sum_{j}^{l} \mathbf{W}_{ij} e^{H_{t-1}(\mathbf{x}_i) - H_{t-1}(\mathbf{x}_j)} e^{2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, y_j) \\
& + \frac{\lambda}{2} \sum_{i}^{l} \sum_{j}^{l} \mathbf{W}_{ij} e^{H_{t-1}(\mathbf{x}_i) + H_{t-1}(\mathbf{x}_j)} e^{2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, -y_j) \\
& + \frac{\lambda}{2} \sum_{i}^{l} \sum_{j}^{l} \mathbf{W}_{ij} e^{-H_{t-1}(\mathbf{x}_i) + H_{t-1}(\mathbf{x}_j)} e^{-2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, y_j) \\
& + \frac{\lambda}{2} \sum_{i}^{l} \sum_{j}^{l} \mathbf{W}_{ij} e^{-H_{t-1}(\mathbf{x}_i) - H_{t-1}(\mathbf{x}_j)} e^{-2\alpha_t h_t(\mathbf{x}_i)} \delta(y_i, -y_j) \,.
\end{aligned}
\tag{E.4}
$$

Factoring out $e^{-2\alpha_t h_t(\mathbf{x}_i)}$ and $e^{2\alpha_t h_t(\mathbf{x}_i)}$ results in (4.16).

Appendix F

Weights assigned to observations

To show that when $\lambda = 0$ the weights associated with observations work similarly to original AdaBoost, we can rewrite Eq. (4.19) as

$$w_i^t = e^{-2H_{t-1}(\mathbf{x}_i)}\delta(y_i, +1) + e^{2H_{t-1}(\mathbf{x}_i)}\delta(y_i, -1), \quad i = 1, \ldots, N. \tag{F.1}$$

The weights at the next iteration, $m + 1$, can be written as

$$\begin{aligned} w_i^{t+1} &= e^{-2H_t(\mathbf{x}_i)}\delta(y_i, +1) + e^{2H_t(\mathbf{x}_i)}\delta(y_i, -1) \\ &= e^{-2H_{t-1}(\mathbf{x}_i)}e^{-2\alpha_t h_t(\mathbf{x}_i)}\delta(y_i, +1) + e^{2H_{t-1}(\mathbf{x}_i)}e^{2\alpha_t h_t(\mathbf{x}_i)}\delta(y_i, -1) \\ &= w_i^t e^{-2\alpha_t h_t(\mathbf{x}_i)}\delta(y_i, +1) + w_i^t e^{2\alpha_t h_t(\mathbf{x}_i)}\delta(y_i, -1)\,. \end{aligned} \tag{F.2}$$

Suppose $y_i = 1$. Misclassifying $\mathbf{x}_i$ leads to $w_i^{t+1} = w_i^t e^{2\alpha_t}$ (the weight is increased). In case of correct classification, $w_i^{t+1} = w_i^t e^{-2\alpha_t}$ (the weight is decreased).

BLANK

Appendix G

Proof of Lemma 5.4.1

To prove (5.26), we can write $\bar{\mathbf{x}}_{\mathcal{L}_0\cup\{\mathbf{x}_j^U\}}$ as

$$\begin{aligned}\bar{\mathbf{x}}_{\mathcal{L}_0\cup\{\mathbf{x}_j^U\}} &= \frac{1}{l_0+1}\Big(\sum_{\mathbf{x}_l\in\mathcal{L}_0}\mathbf{x}_l + \mathbf{x}_j^U\Big)\\ &= \frac{\sum_{\mathbf{x}_l\in\mathcal{L}_0}\mathbf{x}_l}{l_0+1} + \frac{\mathbf{x}_j^U}{l_0+1}\\ &= \frac{l_0\bar{\mathbf{x}}_{\mathcal{L}_0}}{l_0+1} + \frac{\mathbf{x}_j^U}{l_0+1} - \frac{\bar{\mathbf{x}}_{\mathcal{L}_0}}{l_0+1} + \frac{\bar{\mathbf{x}}_{\mathcal{L}_0}}{l_0+1}\\ &= \bar{\mathbf{x}}_{\mathcal{L}_0} + \frac{\mathbf{x}_j^U - \bar{\mathbf{x}}_{\mathcal{L}_0}}{l_0+1}.\end{aligned} \tag{G.1}$$

Replacing $\mathbf{x}_j^U - \bar{\mathbf{x}}_{\mathcal{L}_0}$ with $\mathbf{d}_{l_0+1}$ yields (5.26).

To prove (5.27), $\hat{\mathbf{\Sigma}}_{\mathcal{L}_0\cup\{\mathbf{x}_j^U\},\mathcal{L}_1}$ can be written as

$$\hat{\mathbf{\Sigma}}_{\mathcal{L}_0\cup\{\mathbf{x}_j^U\},\mathcal{L}_1} = \frac{n_0\hat{\mathbf{\Sigma}}_{\mathcal{L}_0\cup\{\mathbf{x}_j^U\}} + (l_1-1)\hat{\mathbf{\Sigma}}_{\mathcal{L}_1}}{l_0+l_1-1}\,. \tag{G.2}$$

Then

$$\begin{aligned}\hat{\mathbf{\Sigma}}_{\mathcal{L}_0\cup\{\mathbf{x}_j^U\}} &= \frac{1}{l_0}\sum_{\mathbf{x}_l\in\mathcal{L}_0\cup\{\mathbf{x}_j^U\}}(\mathbf{x}_l - \bar{\mathbf{x}}_{\mathcal{L}_0\cup\{\mathbf{x}_j^U\}})(\mathbf{x}_l - \bar{\mathbf{x}}_{\mathcal{L}_0\cup\{\mathbf{x}_j^U\}})^T\\ &= \frac{1}{l_0}\sum_{\mathbf{x}_l\in\mathcal{L}_0}\Big(\mathbf{x}_l - \bar{\mathbf{x}}_{\mathcal{L}_0} - \frac{\mathbf{d}_{l_0+1}}{l_0+1}\Big)\Big(\mathbf{x}_l - \bar{\mathbf{x}}_{\mathcal{L}_0} - \frac{\mathbf{d}_{l_0+1}}{l_0+1}\Big)^T\\ &+ \frac{1}{n_0}\Big(\mathbf{x}_j^U - \bar{\mathbf{x}}_{\mathcal{L}_0} - \frac{\mathbf{d}_{l_0+1}}{l_0+1}\Big)\Big(\mathbf{x}_j^U - \bar{\mathbf{x}}_{\mathcal{L}_0} - \frac{\mathbf{d}_{l_0+1}}{l_0+1}\Big)^T\\ &= \frac{l_0-1}{l_0}\hat{\mathbf{\Sigma}}_{\mathcal{L}_0} + \frac{\mathbf{D}_{l_0+1}}{l_0+1}\,.\end{aligned} \tag{G.3}$$

Replacing (G.3) in (G.2) gives (5.27). Given an invertible matrix $\mathbf{A}\in\mathbb{R}^{p\times p}$ and vectors $\mathbf{u},\mathbf{v}\in\mathbb{R}^p$, the Sherman–Morrison identity states that (Sherman and Morrison, 1950)

$$(\mathbf{A}+\mathbf{u}\mathbf{v}^T)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{A}^{-1}}{1+\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}}. \tag{G.4}$$

To prove (5.28), $\mathbf{H}_{\mathcal{L}_0\cup\{\mathbf{x}_j^U\},\mathcal{L}_1}$ can be written as

$$\mathbf{H}_{\mathcal{L}_0\cup\{\mathbf{x}_j^U\},\mathcal{L}_1} = (\mathbf{I}_p + \gamma\hat{\mathbf{\Sigma}}_{\mathcal{L}_0\cup\{\mathbf{x}_j^U\},\mathcal{L}_1})^{-1}. \tag{G.5}$$

Replacing (5.27) in (G.5) gives

$$\mathbf{H}_{\mathcal{L}_0\cup\{\mathbf{x}_j^U\},\mathcal{L}_1} = (\mathbf{I}_p + \kappa\gamma\hat{\mathbf{\Sigma}}_{\mathcal{L}_0,\mathcal{L}_1} + \gamma\beta_{l_0}\mathbf{D}_{l_0+1})^{-1}. \tag{G.6}$$

Using (G.4) in (G.6) leads to (5.28).

Appendix H

Figures 5.1 and 5.2 with error bars

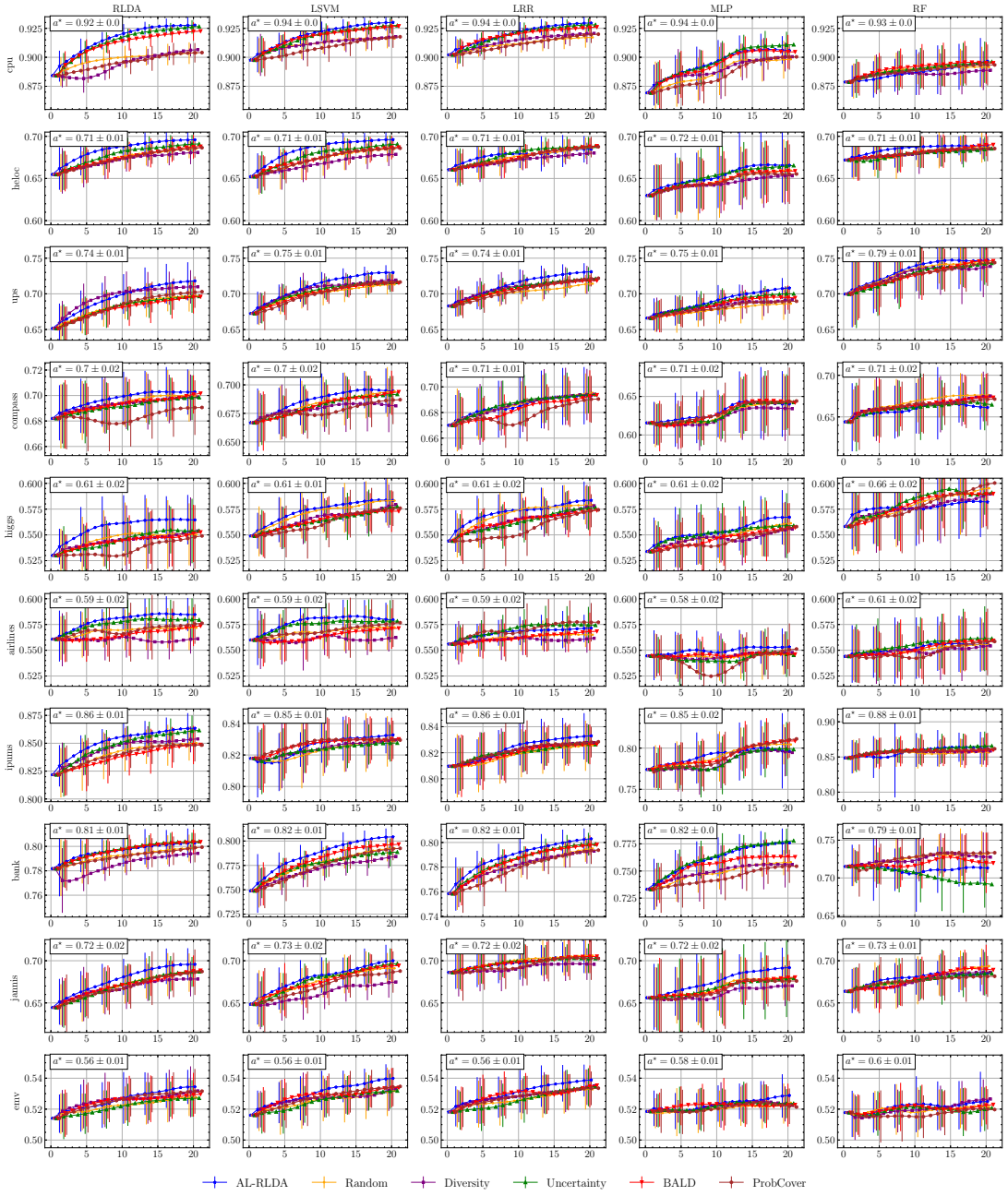

Figure H.1: The average performance in terms of test-set accuracy results with error bars of all considered AL methods across 20 AL cycles.

Appendix I

Figures 5.6 with error bars

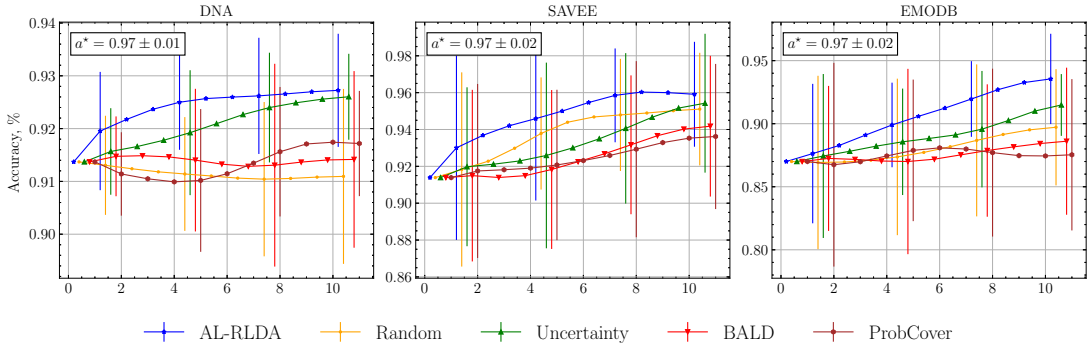

Figure I.1: The average performance in terms of test-set accuracy results with error bars of all considered AL methods across 10 AL cycles.

BLANK

References

Charu C Aggarwal, Xiangnan Kong, Quanquan Gu, Jiawei Han, and S Yu Philip. Active learning: A survey. In *Data classification*, pages 599–634. Chapman and Hall/CRC, 2014.

Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Lies Hadjadj, Emilie Devijver, and Yury Maximov. Self-training: A survey. *Neurocomputing*, 616:128904, 2025.

JA Anderson and SC Richardson. Logistic discrimination and bias correction in maximum likelihood estimation. *Technometrics*, 21(1):71–78, 1979.

Theodore W Anderson. Classification by multivariate analysis. *Psychometrika*, 16(1): 31–50, 1951.

Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6679–6687, 2021.

David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, page 1027–1035, 2007.

Olivier Bachem, Mario Lucic, S. Hamed Hassani, and Andreas Krause. Approximate k-means++ in sublinear time. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), 2016.

Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. Scarf: Self-supervised contrastive learning using random feature corruption. In *International Conference on Learning Representations*, 2022.

Daniyar Bakir, Alex Pappachen James, and Amin Zollanvari. An efficient method to estimate the optimum regularization parameter in rlda. *Bioinformatics*, 32(22): 3461–3468, 2016.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(11), 2006.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

Kristin Bennett and Ayhan Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information processing systems*, volume 11, 1998.

Kristin P. Bennett, Ayhan Demiriz, and Richard Maclin. Exploiting unlabeled data in ensemble methods. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 289–296, 2002.

Michał Bereta. Entropy-based regularization of adaboost. *Computer Assisted Methods in Engineering and Science*, 24(2):89–100, 2017.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer New York, NY, 2006.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100, 1998.

Ulisses Braga-Neto. *Fundamentals of pattern recognition and machine learning*. Springer, 2020.

Ulisses M Braga-Neto, Amin Zollanvari, and Edward R Dougherty. Cross-validation under separate sampling: strong bias and how to correct it. *Bioinformatics*, 30(23): 3349–3355, 2014.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th International Conference on Pattern Recognition*, pages 3121–3124, 2010.

F. Burkhardt, A. Paeschke, M. Rolfes, W. Sendlmeier, and B. Weiss. A database of german emotional speech. In *in Proceedings of Interspeech, Lissabon*, pages 1517–1520, 2005.

Davide Cacciarelli, Murat Kulahci, and John Sølve Tyssedal. Stream-based active learning with linear models. *Knowledge-Based Systems*, 254:109664, 2022.

Guilherme Camargo, Pedro H Bugatti, and Priscila TM Saito. Active semi-supervised learning for biological data classification. *PLoS One*, 15(8):e0237428, 2020.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Introduction to Semi-Supervised Learning*, pages 1–12. MIT Press, 2006.

Ke Chen and Shihai Wang. Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):129–143, 2011.

Ayhan Demiriz, Kristin P Bennett, and Mark J Embrechts. Semi-supervised clustering using genetic algorithms. In *Proceedings of Artificial Neural Networks in Engineering*, 1999.

Pasquale J Di Pillo. The application of bias to discriminant analysis. *Communications in Statistics - Theory and Methods*, 5(9):843–854, 1976.

Thomas Drugman, Janne Pylkkönen, and Reinhard Kneser. Active and semi-supervised learning in asr: Benefits on the acoustic and language models. In *Proc. Interspeech 2016*, pages 2318–2322, 2016.

Florence d'Alché-Buc, Yves Grandvalet, and Christophe Ambroise. Semi-supervised marginboost. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

Ye Du, Yujun Shen, Haochen Wang, Jingjing Fei, Wei Li, Liwei Wu, Rui Zhao, Zehua Fu, and Qingjie Liu. Learning from future: A novel self-training framework for semantic segmentation. In *Advances in Neural Information Processing Systems*, pages 4749–4761, 2022.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, New York, 2 edition, 2001. ISBN 978-0-471-05669-0.

Evelyn Fix and J. L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238–247, 1989.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, page 148–156, 1996.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

Jerome H Friedman. Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175, 1989.

Mingfei Gao, Zizhao Zhang, Guo Yu, Sercan Ö Arık, Larry S Davis, and Tomas Pfister. Consistency-based semi-supervised active learning: Towards minimizing labeling cost. In *European Conference on Computer Vision*, pages 510–526, 2020.

Andrew Goldberg, Xiaojin Zhu, Aarti Singh, Zhiting Xu, and Robert Nowak. Multi-manifold semi-supervised learning. In *Artificial intelligence and statistics*, pages 169–176. PMLR, 2009.

Andrew B. Goldberg, Xiaojin Zhu, and Stephen Wright. Dissimilarity in graph-based semi-supervised classification. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2, pages 155–162, 2007.

Andrew Brian Goldberg and Xiaojin Zhu. *New directions in semi-supervised learning*. PhD thesis, University of Wisconsin–Madison, 2010.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems*, volume 34, pages 18932–18943, 2021.

Zhenghui Gu, Zhuliang Yu, Zhifang Shen, and Yuanqing Li. An online semi-supervised brain–computer interface. *IEEE Transactions on Biomedical Engineering*, 60(9): 2614–2623, 2013.

Wen Guo, Liangliang Cao, Tony X. Han, Shuicheng Yan, and Changsheng Xu. Max-confidence boosting with uncertainty for visual tracking. *IEEE Transactions on Image Processing*, 24(5):1650–1659, 2015.

S. Haq and P.J.B. Jackson. *Machine Audition: Principles, Algorithms and Systems*, chapter Multimodal Emotion Recognition, pages 398–423. IGI Global, Hershey PA, Aug. 2010.

Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

Jiangpeng He, Runyu Mao, Zeman Shao, and Fengqing Zhu. Incremental learning in online scenario. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13926–13935, 2020.

Tomer Hertz, Aharon Bar-Hillel, and Daphna Weinshall. Boosting margin based distance functions for clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning*, page 50, 2004.

Steven CH Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289, 2021.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Alexey Grigorevich Ivakhnenko. Polynomial theory of complex systems. *IEEE transactions on Systems, Man, and Cybernetics*, SMC-1(4):364–378, 1971.

Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Bartosz Krawczyk and Michał Woźniak. Online query by committee for active learning from drifting data streams. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2120–2127, 2017. DOI: 10.1109/IJCNN.2017.7966111.

Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. *Efficient BackProp*, pages 9–48. Springer Berlin Heidelberg, 2012.

Seungyeol Lee, Taeho Kim, and Jae-Pil Heo. Cross-loss pseudo labeling for semi-supervised segmentation. *IEEE Access*, 11:96761–96772, 2023.

S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

Huijuan Lu, Huiyun Gao, Minchao Ye, and Xiuhui Wang. A hybrid ensemble algorithm combining adaboost and genetic algorithm for cancer classification with gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(3):863–870, 2021.

Pavan Kumar Mallapragada, Rong Jin, Anil K. Jain, and Yi Liu. Semiboost: Boosting for semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):2000–2014, 2009.

Hermina Petric Maretic and Pascal Frossard. Graph laplacian mixture model. *IEEE Transactions on Signal and Information Processing over Networks*, 6:261–270, 2020.

Robert Moskovitch, Nir Nissim, Dima Stopel, Clint Feher, Roman Englert, and Yuval Elovici. Improving the detection of unknown computer worms activity using active learning. In *Annual Conference on Artificial Intelligence*, pages 489–493, 2007.

Azamat Mukhamediya, Siamac Fazli, and Amin Zollanvari. On the effect of log-mel spectrogram parameter tuning for deep learning-based speech emotion recognition. *IEEE Access*, 11:61950–61957, 2023.

Subhabrata Mukherjee and Ahmed Awadallah. Uncertainty-aware self-training for few-shot text classification. In *Advances in Neural Information Processing Systems*, volume 33, pages 21199–21212, 2020.

Michiel Noordewier, Geoffrey Towell, and Jude Shavlik. Training knowledge-based neural networks to recognize genes in dna sequences. *Advances in neural information processing systems*, 3, 1990.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Robi Polikar, Lalita Upda, Satish S. Upda, and Vasant Honavar. Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 31(4):497–508, 2001.

Gunnar Rätsch, Takashi Onoda, and K-R Müller. Soft margins for adaboost. *Machine learning*, 42:287–320, 2001.

Zhongzheng Ren, Raymond Yeh, and Alexander Schwing. Not all unlabeled data are equal: Learning to weight data in semi-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21786–21797, 2020.

Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 441–448, 2001.

Raphael Schumann and Ines Rehbein. Active learning via membership query synthesis for semi-supervised sentence classification. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 472–481, 2019.

H. Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Mohammad Shahrokh Esfahani and Edward R Dougherty. Effect of separate sampling on classification accuracy. *Bioinformatics*, 30(2):242–250, 2013.

Jack Sherman and Winifred J Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

Changjian Shui, Fan Zhou, Christian Gagné, and Boyu Wang. Deep active learning: Unified and principled method for query and training. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 1308–1318, 2020.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems*, volume 33, pages 596–608, 2020.

Hang Su, Zhaozheng Yin, Seungil Huh, Takeo Kanade, and Jun Zhu. Interactive cell segmentation based on active and semi-supervised learning. *IEEE Transactions on Medical Imaging*, 35(3):762–777, 2015.

Kevin Swingler. *Applying neural networks : a practical guide*. Morgan Kaufmann, San Francisco, 1996.

Chang Tang, Meiru Bian, Xinwang Liu, Miaomiao Li, Hua Zhou, Pichao Wang, and Hailin Yin. Unsupervised feature selection via latent representation learning and manifold regularization. *Neural Networks*, 117:163–178, 2019.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*, 2017.

Alaa Tharwat and Wolfram Schenck. A survey on active learning: state-of-the-art, practical challenges and research directions. *Mathematics*, 11(4):820, 2023.

Katrin Tomanek and Udo Hahn. Semi-supervised active learning for sequence labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1039–1047, 2009.

Isaac Triguero, Salvador García, and Francisco Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems*, 42(2):245–284, 2015.

Nicolás García Trillos, Franca Hoffmann, and Bamdad Hosseini. Geometric structure of graph laplacian embeddings. *Journal of Machine Learning Research*, 22(63):1–55, 2021.

Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine learning*, 109(2):373–440, 2020.

Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995. ISBN 0-387-94559-8.

He Wang, Peiyin Chen, Meng Zhang, Jianbo Zhang, Xinlin Sun, Mengyu Li, Xiong Yang, and Zhongke Gao. Eeg-based motor imagery recognition framework via multisubject dynamic transfer and iterative self-training. *IEEE Transactions on Neural Networks and Learning Systems*, 2023a.

Ligen Wang and Balázs Kégl. Boosting on manifolds: adaptive regularization of base classifiers. *Advances in Neural Information Processing Systems*, 17, 2004.

Zicheng Wang, Zhen Zhao, Xiaoxia Xing, Dong Xu, Xiangyu Kong, and Luping Zhou. Conflict-based cross-view consistency for semi-supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19585–19595, 2023b.

Koshi Watanabe, Keisuke Maeda, Takahiro Ogawa, and Miki Haseyama. Learning graph laplacian from intrinsic patterns via gaussian process. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

Si Wu, Hau-San Wong, and Shufeng Wang. Variant semiboost for improving human detection in application scenes. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(7):1595–1608, 2018.

Yongxin Xi, Zhen Xiang, Peter Ramadge, and Robert Schapire. Speed and sparsity of regularized boosting. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, pages 615–622, 2009.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(9): 8934–8954, 2022.

Yuzhe Yang and Zhi Xu. Rethinking the value of labels for improving class-imbalanced learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 19290–19301, 2020.

Ofer Yehuda, Avihu Dekel, Guy Hacohen, and Daphna Weinshall. Active learning through a covering lens. In *Advances in Neural Information Processing Systems*, volume 35, pages 22354–22367, 2022.

Dong Yu, Balakrishnan Varadarajan, Li Deng, and Alex Acero. Active learning and semi-supervised learning for speech recognition: A unified framework using the global entropy reduction maximization criterion. *Computer Speech & Language*, 24 (3):433–444, 2010.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*, 2016.

Xueying Zhan, Huan Liu, Qing Li, and Antoni B Chan. A comparative survey: Benchmarking for pool-based active learning. In *International Joint Conferences on Artificial Intelligence*, pages 4679–4686, 2021.

Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. In *Advances in Neural Information Processing Systems*, pages 18408–18419, 2021.

Wenqiao Zhang, Lei Zhu, James Hallinan, Shengyu Zhang, Andrew Makmur, Qingpeng Cai, and Beng Chin Ooi. Boostmis: Boosting medical image semi-supervised learning with adaptive pseudo labeling and informative active annotation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20666–20676, 2022.

Yexun Zhang, Yanfeng Wang, Wenbin Cai, Siyuan Zhou, and Ya Zhang. From theory to practice: Efficient active cost-sensitive classification with expected error reduction. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 153–161, 2017.

Zixing Zhang, Fabien Ringeval, Bin Dong, Eduardo Coutinho, Erik Marchi, and Björn Schüller. Enhanced semi-supervised learning for multimodal emotion recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5185–5189, 2016.

Yue Zhao, Ciwen Xu, and Yongcun Cao. Research on query-by-committee method of active learning and application. In *International Conference on Advanced Data Mining and Applications*, pages 985–991, 2006.

Arman Zharmagambetov and Miguel Á Carreira-Perpiñán. Semi-supervised learning with decision trees: Graph laplacian tree alternating optimization. In *Advances in Neural Information Processing Systems*, volume 35, pages 2392–2405, 2022.

Shusen Zhou, Qingcai Chen, and Xiaolong Wang. Active deep learning method for semi-supervised sentiment classification. *Neurocomputing*, 120:536–546, 2013.

Wei Zhu, Qiang Qiu, Jiaji Huang, Robert Calderbank, Guillermo Sapiro, and Ingrid Daubechies. Ldmnet: Low dimensional manifold regularized neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2743–2751, 2018.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, page 912–919, 2003.

Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

Amin Zollanvari and Edward R Dougherty. Generalized consistent error estimator of linear discriminant analysis. *IEEE Transactions on Signal Processing*, 63(11): 2804–2814, 2015.